Elsevier Editorial System(tm) for Future Generation Computer Systems
Manuscript Draft

Manuscript Number:

Title: Ensuring the Performance and Scalability of Peer-To-Peer Distributed Virtual Environments

Article Type: SS: P2P and Grid Systems - Antonopoulos

Keywords: Distributed applications; Performance evaluation

Corresponding Author: Professor Juan Manuel Orduña Huertas, Ph.D.

Corresponding Author's Institution: Universidad de Valencia

First Author: Silvia Rueda, Ph. D

Order of Authors: Silvia Rueda, Ph. D; Pedro Morillo, Ph.D; Juan Manuel Orduña Huertas, Ph.D.; José Duato, Ph.D

Abstract: Large scale distributed virtual environments (DVEs) have become a major trend in distributed applications, mainly due to the enormous popularity of multi-player online games in the entertainment industry. Peer-to-peer (P2P) architectures have been proposed as an efficient and truly scalable solution for this kind of systems. However, in order to design efficient P2P DVEs these systems must be characterized, measuring the impact of different client behaviors on system performance.
This paper presents the experimental characterization of P2P DVEs. The results show that the saturation of a given client exclusively has an effect on the surrounding clients in the virtual world, having no noticeable effect at all on the rest of clients. Nevertheless, the interactions among clients that can take place in this type of systems can lead to the temporal saturation of an unbounded number of clients, thus limiting the performance of P2P DVEs. In this paper, we also discuss and propose a technique for avoiding the saturation of the client computers in P2P DVEs. The

evaluation results show that the performance and the scalability of P2P DVEs are significantly improved. These results can be used as the basis for an efficient design of P2P DVEs.

Ensuring the Performance and Scalability of Peer-To-Peer Distributed Virtual Environments

P.Morillo, S. Rueda, J. M.Orduña[a], J.Duato[b]

[a]*Departamento de Informática - Universidad de Valencia - Spain*
[b]*DISCA - Universidad Politécnica de Valencia - SPAIN*

Abstract

Large scale distributed virtual environments (DVEs) have become a major trend in distributed applications, mainly due to the enormous popularity of multi-player online games in the entertainment industry. Peer-to-peer (P2P) architectures have been proposed as an efficient and truly scalable solution for this kind of systems. However, in order to design efficient P2P DVEs these systems must be characterized, measuring the impact of different client behaviors on system performance.

This paper presents the experimental characterization of P2P DVEs. The results show that the saturation of a given client exclusively has an effect on the surrounding clients in the virtual world, having no noticeable effect at all on the rest of clients. Nevertheless, the interactions among clients that can take place in this type of systems can lead to the temporal saturation of an unbounded number of clients, thus limiting the performance of P2P DVEs. In this paper, we also discuss and propose a technique for avoiding the saturation of the client computers in P2P DVEs. The evaluation results show that the performance and the scalability of P2P DVEs are significantly improved. These results can be used as the basis for an efficient design of P2P DVEs.

Key words: Distributed Applications, Performance Evaluation

Email addresses: `Silvia.Rueda@uv.es`, `Pedro.Morillo@uv.es`, `Juan.Orduna@uv.es` (P.Morillo, S. Rueda, J. M.Orduña), `jduato@gap.upv.es` (J.Duato)

1. Introduction

The current expansion of massively multi-player online games (MMOGs) has promoted the growth of large scale distributed virtual environments (DVEs). These highly interactive systems simulate a 3-D virtual world where multiple users share the same scenario. Each user is represented in the shared virtual environment by an entity called *avatar*, whose state is controlled by the user through a client computer. The system renders the images of the virtual world that each user would see if he was located at that point in the virtual environment. Thousands and even hundreds of thousands of client computers can be simultaneously connected to the DVE [1]. The client computers can be connected through different networks, and even through Internet [2, 3, 4, 5, 6].

In these systems, architectures based on networked servers have been the major standard for DVEs during the last years [7, 8, 9, 10, 11, 12]. In these architectures, the control of the simulation relies on several interconnected servers. Client computers are assigned to one of the servers in the system. When a client computer modifies the state (usually the position, but it can also modify the appearance or other statefull information) of an avatar, it also sends an updating message to its server, which in turn must propagate this message to other servers and clients. The main reasons for the prevalence of networked-server architectures over peer-to-peer architectures have been the control of the simulation and the awareness problem. Effectively, if there are only a few servers that are controlled by the application owner, the chance of non-desired client behaviors is greatly reduced. On other hand, the awareness problem consists of ensuring that each avatar (for the sake of shortness, in the rest of the paper we will use the term avatar to denote the client computer controlling that avatar) is aware of all the avatars in its neighborhood [13]. Usually, the Area Of Interest (AOI) [14] of an avatar is considered as the neighborhood for that avatar. In networked-server architectures, the awareness of each avatar can be easily provided by the servers, since they periodically synchronize among them [15, 16, 17] and therefore they know the approximate location of each avatar in the scene at each instant. Taking into account the AOI size, the servers can compute which other avatars are the neighbors of each avatar. Nevertheless, the inter-server dependences shown by networked-server architectures prevent these schemes from quickly adapting to the workload generated by users behavior [8], particularly for the case of MMOGs, where up to hundreds of thousands of clients can be simultaneously connected to the system.

Peer-to-peer (P2P) architectures were proposed in the nineties for DVEs [18, 19, 20]. In classic peer-to-peer architectures, each client computer is also a system server, and the control of the simulation is distributed among all the client computers. In hybrid peer-to-peer architectures, only some of the client computers act as system servers. Peer-to-peer architectures seem to be the most flexible scheme in order to provide good scalability for large scale DVEs [21], and several online games based on P2P architectures have been designed last years [22, 23, 24, 25, 26, 27]. Several awareness methods have been proposed for solving the awareness problem [28, 29, 30, 31, 32], at the cost of imposing both additional communications between different avatars and additional computations (performed by the client computers controlling these avatars). In this scenario, the design of truly efficient P2P DVEs can allow these architectures to become the major trend in a near future. In order to achieve such goals, the joint impact that the user behavior, the awareness method and other computations can have on the real system performance must be actually measured. Particularly, it must be guaranteed that these architectures are less prone to saturation than the networked-server architectures under different user behavior.

Based on that motivation, in this paper we propose the characterization of peer-to-peer DVEs by performing real-system measurements. The characterization results show that, unlike in networked-server architectures, the system flexibility remains unchanged with the number of client computers connected to the system. That is, the peer-to-peer scheme easily adapts to the workload generated by users behavior, since every client can also act as a server. Also, the results show that the saturation of a given client exclusively has an effect on the surrounding clients in the virtual world, having no effects at all on the rest of avatars. Finally, the characterization results show that the use of a peer-to-peer scheme does not prevent client computers from reaching saturation. That is, under certain behaviors an unbounded number of clients could reach saturation, seriously affecting the scalability and/or the performance of P2P DVEs. In order to solve this problem, in this paper we also propose a technique for avoiding client saturation. Unlike another existing techniques proposed for MMOGs [33, 34], our proposal is focused on the state of the client computer, rather than on the application requirements, and it exclusively drops obsolete messages containing location updates. As a result, it does not require the analysis of *obsolescence* relationships [34] that depend on the application, and it can be adjusted to fulfill any given time constraint. The performance evaluation results show that the benefits achieved by preventing client computers from reaching saturation are higher than the drawbacks of loosing information about the current state of other client computers. Thus, the selected method avoids client saturation on DVEs based on P2P architectures while maintaining the awareness rate (the percentage of correct detections of new neighbors) close to 100%, regardless of the movement pattern and the initial distribution of avatars. Therefore, the proposed technique can ensure the performance and the scalability of P2P DVEs.

The rest of the paper is organized as follows: Section 2 details the proposed characterization setup that allows to experimentally study the behavior of peer-to-peer DVEs. Next, Section 3 presents the characterization results. Section 4 describes the proposed method for avoiding the saturation of the client computers, and Section 5 shows the performance evaluation of the saturation avoidance technique. Finally, Section 6 outlines some concluding remarks.

2. Characterization Setup

We propose the characterization of P2P DVEs by simulation. The evaluation methodology used is based on the main standards for modeling collaborative virtual environments, such as FIPA [35], DIS [36] and HLA [37]. Since DVEs are inherently based on networks, the metrics used for evaluating the performance of these systems include the two main metrics used for evaluating network performance. These metrics are latency and throughput [38]. Nevertheless, in order to avoid clock skewing we have selected throughput and response time (defined as the round-trip delay for the messages sent by each client) as the performance metrics to be characterized.

Concretely, we have used a distributed simulator modeling a DVE based on a peer-to-peer architecture. The simulator is written in C++ and it is composed of two different applications, one modeling the clients and the other one modeling the central *Loader*. The Loader is the entity in charge of the initialization of new avatars when they join the DVE [39], and it is also denoted as Bootstrap server [28].

Each instance of the client application (as well as the central loader application) can be executed either on the same machine (thus composing a centralized simulator that does not actually communicate clients through the network) or on a different machine, resulting in a truly distributed simulator.

All clients must initially join the system through the central loader. Both applications use different threads for managing the different connections they must establish. Such connections are performed by means of sockets.

Each client has a main thread for managing the actions required by the user and different threads for communicating with its neighbor clients. For each neighbor, two threads are executed, one for listening and one for sending messages. Similarly, the central loader has two threads for communicating with each client in the system and also a main thread. It must be noticed that once a client has joined the system, that client does not need to communicate with the central loader any more. Since the goal of this characterization is to study how the system evolves as clients interact with the environment, rather than analyzing how new clients join the system, our simulator initially provides each client with the IP addresses of its initial neighbors.

A simulation consists of each avatar performing 100 movements. An iteration of the whole system consists of all avatars making a movement. Each avatar notifies its neighbors as well as the central loader when it reaches the 101th iteration, and then it leaves the system. We have chosen the number of 100 iterations (movements) for a simulation because it is the number of movements that the most distant avatar needs to reach the center of the square virtual world used for characterization purposes. This virtual world is a 2D square whose sides were 100 meters long. Each time an avatar moves, it sends a message to all its neighbor avatars (the client computer controlling that avatar sends a message to the client computers controlling the neighbor avatars). These destination avatars then send back an acknowledgment to the sending avatar, in such a way that the sending avatar can compute the round-trip delay for each message send. We have denoted the average round-trip delay for all the messages sent by an avatar as the Average System Response (ASR) for that avatar (for that client computer).

Although we have implemented several different awareness methods in the simulator, we have used the COVER method for characterization purposes, since to our knowledge this is the only method that is capable to provide full awareness under non-uniform movement patterns of avatars [40].

Two different characterization setups have been made, depending on the metrics to be studied. In order to study the system throughput (the maximum number of clients the system can support while maintaining full awareness and reasonable latency levels), we have used a cluster of 14 nodes. One of these PCs hosted the central loader, and the rest of the 13 PCs hosted the clients in the system in a uniformly distributed way. Each node was a dual

Opteron processor running Linux (SuSE 10.1 distribution).

Although a real system does not require communication between each client and the central loader, we have implemented a monitoring algorithm to check the awareness rate supported by the system. The awareness rate can be defined as the percentage of neighbor avatars that are correctly computed by the avatars themselves. In order to compute the awareness rate in a distributed system, some central entity should know the current location of all the avatars in the system. Taking into account the AOI of the avatars, this entity can compute which neighbors are within the AOI of each avatar. Therefore, each time that an avatar i makes a movement the corresponding client computer should report to this central entity not only its new location but also which other avatars it considers as its neighbors. Since the central entity knows the current location of all the avatars and their AOI size, it can compute the correct neighbors for i and compare if the neighbors reported by i are correct or wrong. Since the same process is performed with all the avatars, it can compute the percentage of avatars that have correctly computed which other avatars are their neighbors (that is, the awareness rate). Concretely, we have designed the Loader as the central client in charge of computing the awareness rate. Thus, the monitoring algorithm consists of each client dividing its cycle time in two phases. In the first phase, clients move following a given movement pattern (described in Section 3) and they communicate each other their new location in the virtual space by exchanging messages. In the second phase, each client sends the central loader a message containing information about its new location and also about which other clients it considers as its neighbors. Thus, the central Loader can compare the awareness information sent by each avatar with its own awareness computations (the right ones, since it has information about the location of all the avatars), and it can compute the percentage of correct awareness computations made by that client. In this way, the central Loader can compute the awareness rate in real time. We have used a movement cycle of one client movement each 3.95 seconds. From this period, 2.85 seconds are dedicated to the first phase and 1.1 seconds are dedicated to the second phase. As a reasonable latency level, we have considered a threshold ASR value of 250 ms., since this is the highest ASR value that provides users with a good interaction level [41].

In order to study the system latency, we have used a different characterization setup. In this case, we have used personal computers interconnected by a fast Ethernet network, hosting the avatars in a uniformly distributed way. The purpose of such setup is to establish an upper number for the average value of the ASR when client computers are slightly loaded. Finally, in order to study the effects that the saturation of a single client computer has on the rest of clients, we have used the same setup except that one of the PCs hosts a single client, and the rest of the PCs host the rest of the clients in a uniformly distributed way. In these two setups we have not monitored the awareness rate, and the movement cycle time has been reduced accordingly. The idea is to move the single hosted client as fast as possible, to force that client to send a huge number of messages (therefore saturating that client) and to study the effect that this saturation has on the neighbor clients.

3. Characterization of P2P DVEs

We have simulated the behavior of a set of independent avatars in a generic DVE based on a P2P architecture. These avatars are located within a seamless 3D virtual world [1] following three different and well-known initial distributions: uniform, skewed and clustered [7, 8]. Starting from these initial locations, in each simulation avatars can move into the scene following one of three different movement patterns: Changing Circular Pattern (CCP) [9], HP-All (HPA) [42] and HP-Near (HPN) [43]. CCP considers that all avatars in the virtual world move randomly around the virtual scene following circular trajectories. HPA considers that there exists certain "hot points" where all avatars approach sooner or later. This movement pattern is typical of multiuser games, where users must get resources (as weapons, energy, vehicles, bonus points, etc,) that are located at certain locations in the virtual world. Finally, HPN also considers these hot-points, but only avatars located within a given radius of the hot-points approach these locations. In order to illustrate these movement patterns, Figure 1 shows the final distribution of avatars that a 2-D virtual world (represented as a square map) would show if these movement patterns were applied to a uniform initial distribution of avatars. For evaluation purposes, we have considered the nine possible combinations of the three initial distributions of avatars in the virtual world and the three movement patterns.

3.1. Scalability

The first feature to be proved in a peer-to-peer scheme is its scalability. With this purpose, we have performed more than 4000 experiments to study the behavior of DVEs based on a P2P architecture. In all of them, the system

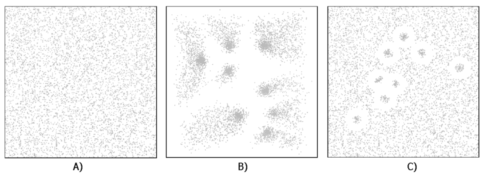

Figure 1: Final distribution of avatars for a) CCP, b) HPN, and c) HPA movement patterns applied to an initial uniform distribution of avatars.

has provided a full awareness rate while saturation has not been reached by any CPU. We have made tests with different world sizes S (the number of connected clients), although for the sake of shortness we show here a single case with 120 avatars. In all the cases the results were very similar, and the average ASR values obtained did not vary from one population to another, showing the inherent scalability of the peer-to-peer scheme.

Concretely, Figure 2 shows a representative example of the experiments performed in order to study the system throughput. On the X-axis this figure shows the iteration number of the simulation performed, and on the Y-axis it shows the average value in seconds for the ASR of all the avatars. Each point in the plots represents the average ASR of ten different executions of the same simulation.

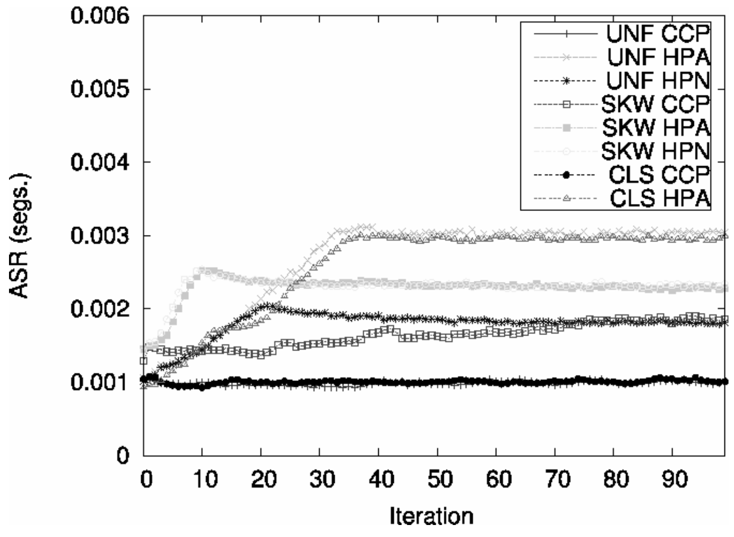

Figure 2: Average ASR values for different configurations of a DVE with 120 avatars

Figure 2 shows that the ASR remains always far below the threshold of

250 ms., and the awareness rate provided for all the plots in this figure was 100%. Since in these experiments each cluster node hosted 40 avatars, we can state that there would not be any problem (the CPU utilization would be far from 100%) when executing one avatar in one PC.

Additionally, we have analyzed the system behavior in a transverse way, that is, we have studied the average ASR values for different population sizes. Although we have performed this analysis for all the combinations of initial distributions and movement patterns, for the sake of shortness we show here a single case. All the cases showed similar results. Concretely, Figure 3 shows the average ASR values provided to avatars when following the Uniform-CCP scheme.

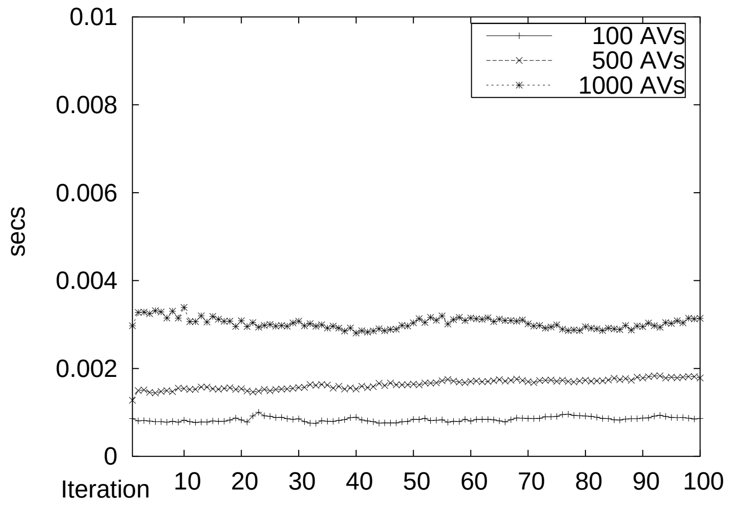

Figure 3: Average ASR values for different populations of avatars under a Uniform-CCP combination

Figure 3 shows three flat lines at a different levels, showing that while the population increases in an order of magnitude (from 100 to 1000 avatars) the average ASR increases by a factor of two, and it is still far from the QoS threshold of 250 ms.. Therefore, these experiments prove that the peer-to-peer scheme is fully scalable, regardless of the number of client computers connected to the system. Although these results could be expected due to the inherently distributed nature of the peer-to-peer scheme, it is worth mention the remarkable differences with the results provided by the networked-server scheme [8].

3.2. Levels of Interaction

Another aspect of P2P DVEs to be characterized is the behavior of the system under different levels of interactions, that is, under different communication rates and different numbers of neighbors to communicate with. Concretely, we have measured the average value of the ASR provided to all avatars for different movement patterns and initial distributions of avatar in the virtual worlds. Since the workload that a given avatar adds to the system depends on both the movement rate of the avatar and also on the number of neighbor avatars in the virtual world [44], we have measured the ASR provided by the system for different values of these two parameters. In these tests we used the platform of interconnected PCs and a wide range of populations sizes. However, for the sake of shortness we will show in this section the results for a system with 101 avatars. Figure 4 shows the results for all of the distributions with a cycle period T of 0.6 seconds (all avatars make a movement every 0.6 seconds) and an AOI of 10 meters, while Figure 5 shows the results for the same AOI but with a movement period of 0.1 seconds. Then, Figure 6 shows the results for an AOI of 20 meters and a movement period of 0.6 seconds. All these figures show on the X-axis the iteration number (a simulation is composed of 100 iterations), and they it show on the Y-axis the average value of the ASR provided to all the avatars.

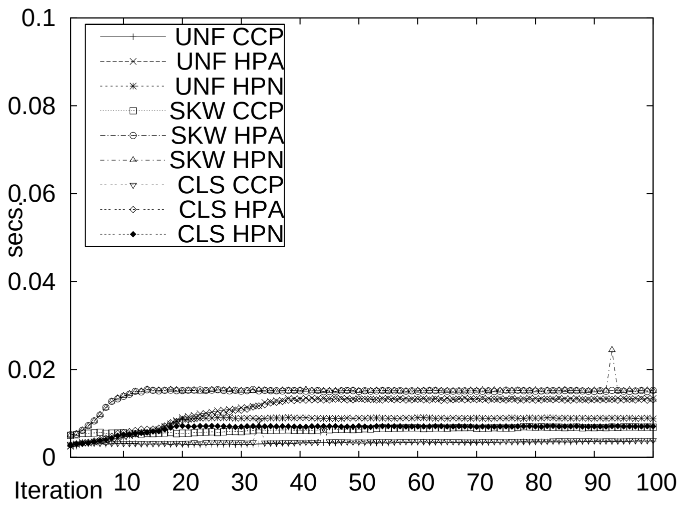

Figure 4: Average ASR values for T=0.6 s.& AOI=10 m.

All the plots in Figure 4 are flat lines of different ASR values, showing that the system is below its saturation point and the average ASR does not indefinitely increase. Moreover, the plots for all the movement patterns are below 0.02 seconds, far away from the QoS threshold of 250 milliseconds. Figure 5 shows the results for the same number of avatars when the cycle period is reduced to 0.1 seconds (the movement rate of all avatars is increased). In this case, the three plots corresponding to the HPA movement pattern and the plot for the HPN pattern with a skewed initial distribution of avatars show a linear increasing with the number of iterations. That is, for these patterns the client computers are not capable of sending, returning and processing the number of messages generated by avatars when they move in a period cycle. This is due to the fact that in the HPN and particularly in the HPA pattern, avatars tend to crowd the hot points, increasing the number of neighbors in the AOI in the subsequent iterations. Since they must send more messages in the same cycle period, the client computers become saturated and the average ASR increases. This behavior shows that the behavior of the clients (their movement rate and their movement pattern) can have a significant effect on the overall performance of P2P DVEs.

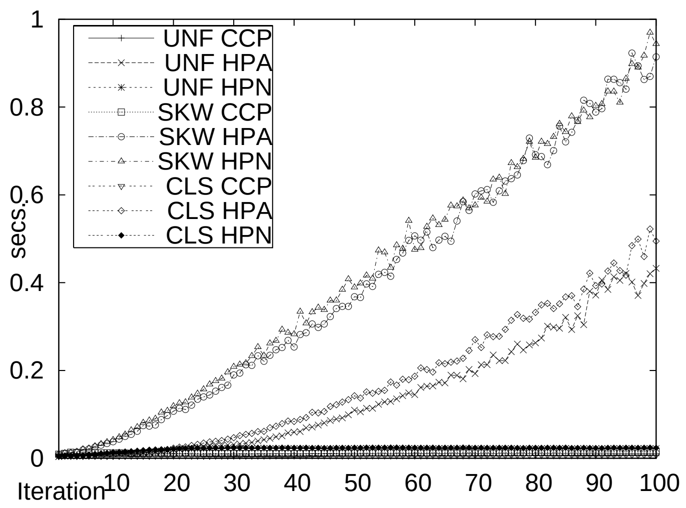

Figure 5: Average ASR values for T=0.1 s.& AOI=10 m.

Figure 6 shows the results for the same population of avatars and the same patterns when the AOI size is doubled. The results shown in this figure are similar to those shown in Figures 4 and Figure 5. Figure 6 shows all the plots except one with a flat slope. Although the average ASR values are slightly higher than the ones shown in Figure 4, all of them are far away from 0.25 seconds.

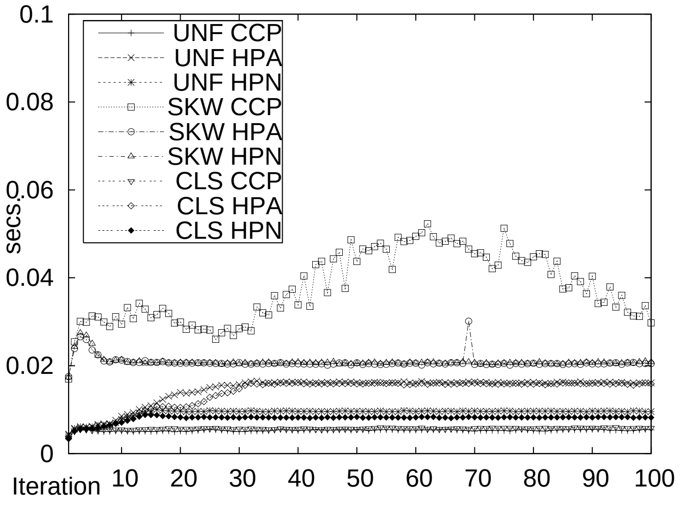

Figure 6: Average ASR values for T=0.6 s. & AOI=20 m.

These results show that increasing the AOI size has less significant effects on system latency than the movement rate of avatars. Also, these results show that under certain circumstances (for example, when avatars move fast) the ASR provided to the avatars in a P2P DVE can exceed any threshold value of the interaction level (several plots in Figure 5 shows a parabolic slope). Additionally, the number of clients that can potentially suffer from these effects is not bounded. Therefore, we can conclude that an efficient behavior of the DVE system is not inherently guaranteed by the Peer-to-Peer architecture.

3.3. Client Saturation

Additionally, we have studied the effects that a slow client computer or network link(s) can have on the rest of client computers. For this study, we have tested the 9 different combinations of initial distributions of avatars and movement patterns on a real DVE based on a P2P architecture composed of 101 PCs. However, for the sake of shortness we show here only the results for some of these combinations, since all of them were similar. In these experiments, we have forced a given avatar (controlled by a single client computer) to move faster and faster, until the number of movements per second is so high that the client computer cannot send all the messages to the corresponding neighbors within the same cycle period. Under these conditions, we have measured the latency (the ASR) provided to different avatars.

The COVER awareness method [30] distinguishes between different kinds of neighbors of a given avatar i. First, the first level neighbors (L1) are those avatars located inside the AOI of avatar i. Next, the second level neighbors (L2) are those avatars that are L1 neighbors of the L1 neighbors of i and are not inside AOI of avatar i. That is, the second level neighbors are the neighbors of the neighbors of i. Therefore, we have computed the average ASR value provided to i, the average ASR value provided to its L1 neighbors, the average ASR value provided to its L2 neighbors and the average ASR value provided to rest of the avatars in the system.

For the sake of shortness, we will show the results for DVEs with 101 avatars and AOI of 10 meters. Figure 7 shows the results for the uniform distribution with a movement rate of 2.1 seconds and the CCP movement pattern. In this and the subsequent figures, we have labeled the ASR value provided to avatar i as AV0. The average ASR values provided to the L1 neighbors of i have been labeled as L1N, the average ASR values provided to the L2 neighbors of i have been labeled as L2N, and the average ASR values provided to the rest of avatars have been labeled as NTN.

Figure 7 shows that for the combination of uniform distribution of avatars and a CCP movement pattern the saturation of AV0 mainly has an effect on its L1 neighbors, and it has no significant effects neither on its L2 neighbors nor on the rest of avatars.

Figure 8 shows the results for the combination of a uniform distribution of avatars and an HPAll movement pattern. In this case, while the plot for AV0 has a parabolic shape, the shape of the plot for L1 neighbors is linear with a low slope. The plot for the L2 neighbors is similar to the plot for L1 neighbors, except that from the 70th iteration up it becomes negligible. The plot for the rest of avatars is a flat line with a negligible value. These plots indicate that the effects of the saturation of AV0 (the client computer controlling AV0) are one order of magnitude lower in its L1 and L2 neighbors, while they are negligible in the rest of avatars.

Figure 9 shows the results for a uniform initial distribution of avatars

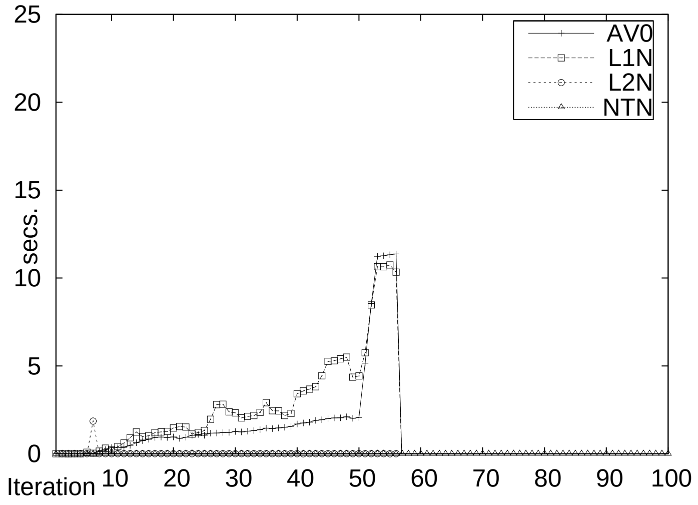

Figure 7: Effects of a client saturation on different kinds of avatars under a Uniform-CCP combination

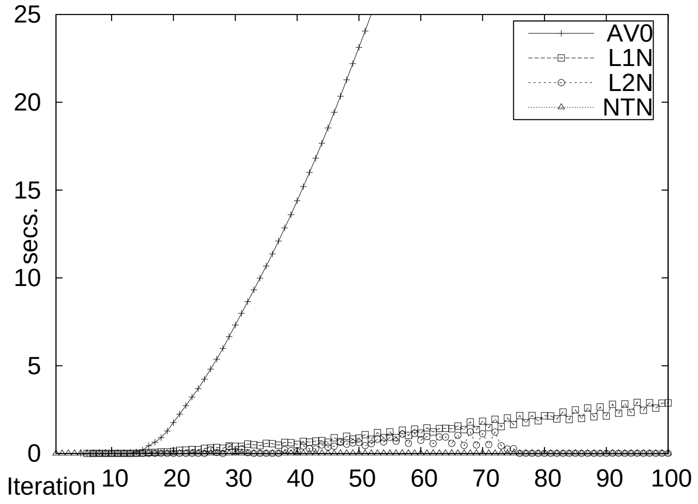

Figure 8: Effects of a client saturation on different kinds of avatars under a Uniform-HPALL combination

following an HPNear movement pattern. In this case, the shape of the plot for AV0 is linear with a high slope, while the shape of the plot for L1 neighbors is linear with a low slope and the plots for the L2 neighbors and the rest of the avatars are flat lines. That is, the difference between the HPA and HPN movement patterns makes that in the latter case the L2 neighbors are not affected by the saturation of a given avatar.

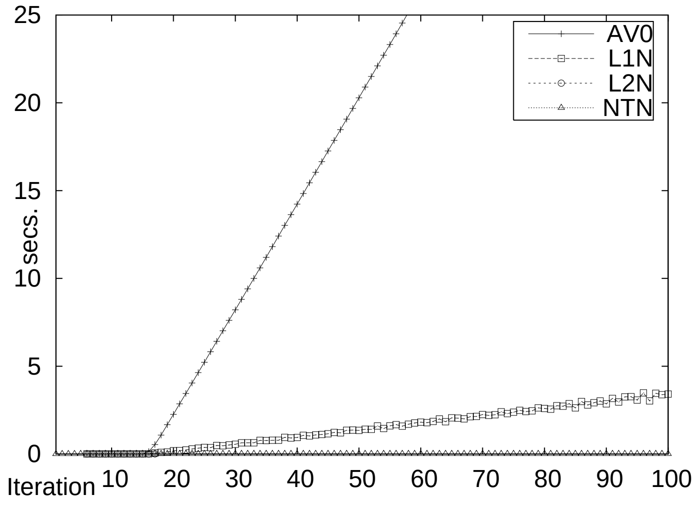

Figure 9: Effects of a client saturation on different kinds of avatars under a Uniform-HPNear combination

Figures 10 and 11 show the results for the skewed initial distribution of avatars and CCP and HPAll movement patterns, respectively. The results for the HPNear movement pattern were very similar to the ones for HPAll movement pattern, and we are not including them for the shake of shortness. The results in these two Figures are similar to those in Figures 8 and 9, respectively. That is, the saturation of a given client has a significant effect on the L1 neighbors, and this effect is one order of magnitude lower than the ASR provided to the saturated avatar if an HPN movement pattern is used. On the contrary, if a HPAll movement pattern is used then the L2 neighbors are also affected by the saturation.

Therefore, with all these results we can state that the effects of the saturation of a given avatar are limited to L1 and L2 neighbors. The rest of avatars are not affected. The reason for the behavior shown in Figure 7 is the circular movement of CCP pattern. This pattern propagates and increases the high ASR values of the L1 neighbors to other L1 neighbors as the

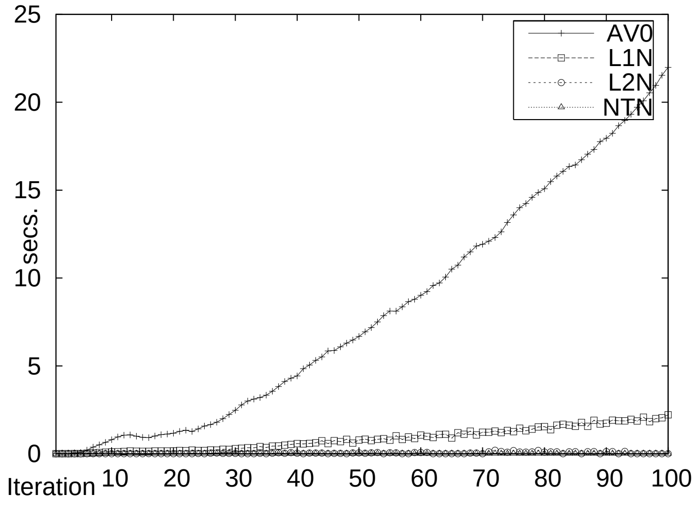

Figure 10: Effects of a client saturation on different kinds of avatars under a Skewed-CCP combination

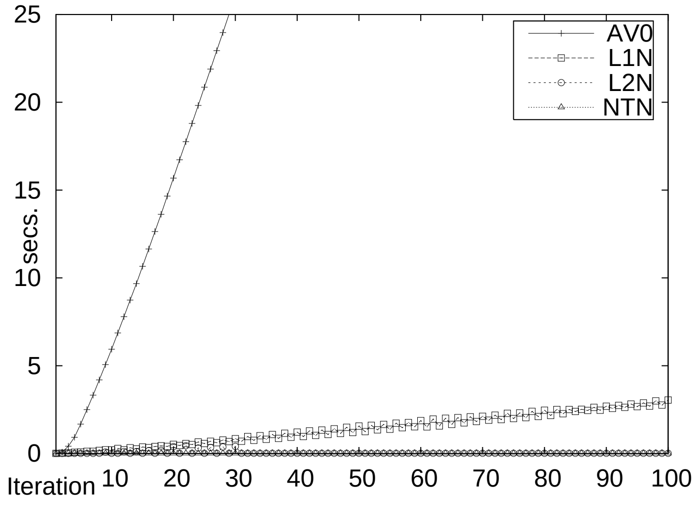

Figure 11: Effects of a client saturation on different kinds of avatars under a Skewed-HPAll combination

simulation proceeds. However, we can conclude that the Peer-to-Peer architecture does not prevent client computers from reaching saturation if a large amount of neighbors move fast. Moreover, the number of client computers that can suffer from these effects is not bounded. Therefore, the performance of the DVE can be significantly decreased under certain user behaviors. In order to ensure a good system performance regardless of the users behavior, a saturation avoidance technique should be developed for P2P DVEs.

4. A Saturation Avoidance Technique for P2P DVEs

In this section, we discuss and propose a saturation avoidance technique for P2P DVEs. Unlike another existing techniques proposed for MMOGs [33, 34], our proposal is focused on the state of the client computer, rather than on the application requirements, and it exclusively drops obsolete messages containing location updates. As a result, As a result, it does not require the analysis of *obsolescence* relationships that depend on the application, and it can be adjusted to fulfill any given time constraint. Some preliminary results show that this technique can effectively avoid the client saturation in P2P DVEs [45] without the need of analyzing *obsolescence* relationships [34]. In this way, the proposed technique can be used with any P2P DVE application.

The workload that a given avatar adds to a DVE basically depends on two factors, the movement rate of that avatar and the number of neighboring avatars[8]. Therefore, the computational workload that a given client computer should support in a P2P DVE is directly related to the number of neighbor avatars in the virtual world and also to the movement rate of that avatar and its neighbors. Additionally, the computational requirements of each client computer also depends on the current state of the simulation (computing requirements for updating and rendering the 3D virtual environment, the time required for establishing new connections, etc.).

In large-scale DVEs, a given avatar a can be frequently surrounded by a high number of neighbors moving fast. In such situations, avatar a will receive a new message containing the updated location of its neighbors each time that any of its neighbors moves. If the client computer controlling the avatar a supports a high load (its CPU(s) utilization rate is (are) close to 100% due to the simulation state), it cannot process such updating messages at the required rate, and the processing of such messages is delayed (they are saved in a FIFO buffer). As a result, the processing of these messages becomes useless (since they provide obsolete information). Moreover, the delayed processing of such messages also requires some computational power, therefore contributing even more to the saturation of the client computer. The basic idea of the proposed method is to discard the oldest updating messages when the client is close to saturation, and to process only the newest messages.

It must be noticed that, unlike the random packet dropping that can take place within some overlay frameworks proposed for P2P DVEs [46, 47], our proposal consists of selectively dropping the oldest messages. The discarding of the obsolete messages is performed at the instant when the avatar is going to make a new movement. We have chosen that instant because a new movement implies the sending of updating messages to the neighbor avatars, thus increasing the workload on that client. In order to prevent the client computer to reach saturation, in our proposal the useless workload is discarded before increasing the useful workload. Our proposal does not take into account *obsolescence* relationships among messages as other proposals for MMOGS do [33, 34]. The reason is that our proposal exclusively discards messages containing locations update, and therefore an older message inherently becomes obsolete. Also, our proposal differs from the RIO-like technique [34] in that it is focused on system performance, instead of user perception. Therefore, it exclusively discards obsolete messages, based on the client computer state (when it is close to saturation), regardless of the interactivity threshold. Nevertheless, the final goal is the same one: to avoid the performance degradation.

We have considered four different alternative methods for discarding obsolete updating messages. These are the following ones:

- Discarding Messages DM1: First, this method checks if the queue of messages pending of processing is empty. If not, all the messages in the queue are discarded. This method is based on assuming that under normal circumstances all the messages are quickly processed, and therefore the existence of pending messages indicates that the client computer has reached saturation.

- Discarding Messages DM2: This method is similar to the previous one, except that the queue of pending messages is emptied only if the queue size exceeds the number of current neighbors of the avatar. This alternative is based on the fact that each avatar approximately receives as much messages as the number of neighbors it has, and therefore we should allow to increase the queue size up to this threshold value.

- Discarding Messages DM3: This alternative selectively discards those messages whose waiting time at the queue is greater than a certain threshold value. This is a less aggressive alternative (it does not discard recent messages).

- Discarding Messages DM4: This alternative takes into account the percentage of CPU utilization in order to activate the discarding of messages. If the CPU utilization of the client computer exceeds a threshold percentage, then it discards all the messages whose waiting time at the queue is greater than a certain threshold value.

We have measured the performance of all the alternative techniques. For comparison purposes, we have also measured the performance of the original system without applying any saturation avoidance technique. We have used the ASR as the metric for evaluating the performance of the different alternatives. The figures below show different plots, labeled as DM1 to DM4, denoting the four alternative strategies. The plot labeled as "ORIGINAL" denotes the original system (without using any saturation avoidance technique). We have tried the nine combinations of initial distributions and movement patterns, but for the shake of shortness we only show here the results for some of these combinations.

Concretely, Figure 12 the performance results for the considered alternatives and for the combination of Uniform initial distribution and HPA movement pattern. Figure 12 shows that only the plot for the original system has a parabolic slope. The rest of the plots show an slight increase of the ASR values and then all of them show an almost flat slope. These results show that all of the considered techniques are able to avoid the system saturation (the ASR values reached by any of the alternatives are not greater than 100 milliseconds, still far from the threshold value of 250 milliseconds).

Figure 13 shows the performance results for the considered alternatives and for the combination of Skewed initial distribution and the CCP movement pattern. Figure 13 shows that again the only plot whose ASR values exceed the threshold value of 250 milliseconds is the plot for the original method. The four considered alternative methods manage to keep the system below the saturation point. Although there are no significant differences among the plots corresponding to the alternative methods, the DM4 technique seems to provide the lowest ASR values. Additionally, this technique guarantees that the messages are discarded only if necessary (it is the only

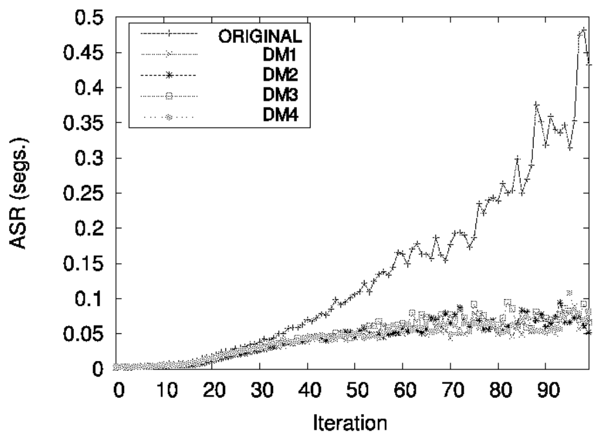

Figure 12: Average ASR values for different methods of message discarding under a UNF-HPA combination

technique that takes into account the current state of the CPU). Therefore, we have chosen the DM4 option as the technique to be implemented.

We have denoted the DM4 alternative method as *DPMess*, for Discarding Pending messages. Thus, the DPMess technique consists of checking the CPU utilization rate of the client computer each time that the avatar hosted by that client moves, in order to detect if the computer is close to saturation. In that case, the client computer should check all the updating messages that are pending from processing, and it should discard those messages older than a certain threshold value (by deleting them from the FIFO buffer). The pseudocode of the proposed algorithm could be the one shown in Figure 14

It is worth mention that the DPMess technique only discards messages containing location updates of other avatars. It does not discard any message containing information concerning the awareness method. In this way, it provides an awareness rate as high as possible.

The DPMess technique has two parameters that should be tuned, the *CPU_threshold* value and the *t_threshold* value. The first one defines the limit for considering a client computer as saturated, and the second one defines the limit for considering an updating message as obsolete. We have chosen for the first parameter a CPU utilization of 90%, because this is the limit proposed in the literature for considering a server (in a DVE based on

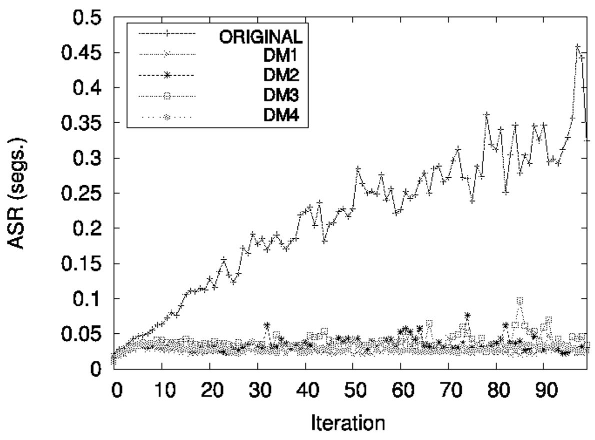

Figure 13: Average ASR values for for different methods of message discarding under a SKW-CCP combination

a networked-server architecture) as saturated [8]. We have experimentally tuned the second parameter. Although the results corresponding to this tuning are not shown here due to space limitations, we have obtained the best results for a $t_{threshold}$ value of 0.005 seconds.

5. Performance Evaluation

We propose the performance evaluation of the saturation avoidance technique by simulation. For completely evaluating the performance of the proposed technique we have used different metrics. In order to measure the overall performance of the system, we have used the round-trip delay (the ASR) of the messages sent by each client computer. Additionally, we have studied other parameters specific from peer-to-peer DVEs. Concretely, we have studied the awareness rate achieved in each simulation and also the average delay between the instant when a new neighbor enters the AOI of a given avatar and the instant when that avatar knows about that neighbor. We have denoted this parameter as the *Awareness Delay*. Finally, we have also studied the percentage of messages discarded by the proposed method.

We have studied the behavior of the proposed algorithm for the nine combinations of initial movement patterns and initial distributions of avatars.

```
if $CPUcurrent > CPUthreshold$ then
  For ALL $messages$ in $pending\_msg\_queue$
    if $msg.type = location\_update$ then
      if $Time - msg.t\_recv > t\_threshold$ then
        discard($msg$)
      end
    end
  end
end
```

Figure 14: Algorithm for discarding messages.

Also, we have performed simulations with different populations sizes (different numbers of avatars) and for different movement rates. Nevertheless, for the shake of shortness we only present here some representative results for a population size of 100 avatars. The results for the different possible configurations were similar to the ones shown in this section.

In order to study the performance of the proposed method, we have studied the system behavior under both a high and a low workload levels. Concretely, we have used a high movement rate (all avatars performing a new movement every 0.15 seconds) in order to generate a high workload, and a lower movement rate (a new movement every 0.5 seconds) to generate a low system workload.

For comparison purposes, we show in this section the simulation results for each DVE configuration when using the DPMess technique and also the results obtained without applying the DPMess. We have denoted the plots corresponding to the former option as "DPMess", and the ones corresponding to the latter option as "Original". We have not compared these results with the results provided by the existing techniques [33, 34] because these techniques are not comparable to the DPMess technique. The former ones focus on providing the interactivity required by the application, regardless of the system state, while the latter focuses on avoiding the saturation of the client computers.

5.1. Latency

First, we have studied the system latencies (Average System Response) achieved with the proposed technique. Figure 15 shows the average ASR values obtained for a system supporting a high workload (each avatar performing a movement every 0.15 seconds) when avatars move following the combination of HPA movement pattern-skewed initial distribution. This figure shows on the X-axis the iteration number, and on the Y-axis it shows the average ASR value obtained for all the avatars in that iteration.

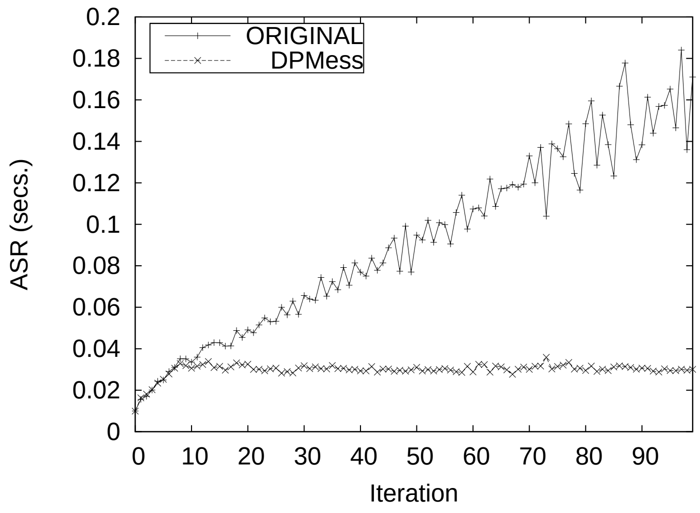

Figure 15: Average ASR values obtained under a high workload

The plot for the DPMess method in Figure 15 shows a flat slope, keeping the average ASR values below 0.04 seconds, far away from the latency values considered as acceptable for users [41]. However, the plot corresponding to the simulation without the proposed technique (Original) shows a significant and constant slope, linearly increasing the average ASR values as the simulation proceeds. These results show that when the system is under a high workload then preventing avatars from reaching saturation provides significant benefits in term of the response time offered to avatars.

5.2. Awareness

Additionally, we have studied how the proposed technique affects to the awareness rate provided to avatars, since providing a good awareness rate is a necessary condition for achieving time-space consistency in DVEs. On the one hand, if an avatar becomes saturated and it does not respond to its neighbors in time, then the awareness rate of its neighbors could be affected.

On the other hand, the impact of rejecting messages could have an effect on the awareness rate, and therefore it should be analyzed.

In order to measure the awareness rate, at each iteration each avatar sends information about its position and which other avatars it considers as its neighbors to the central loader, as we described above. The central loader can determine from this information if each avatar must be aware or not of all its neighbors. By means of the central loader, we have measured the ratio between the number of neighbors that each avatar should detect and the number of neighbors that each avatar has actually detected. We have denoted this parameter as the awareness rate Cs for each avatar.

Figure 16 shows the results for the awareness rate when the system is under a high workload. In this Figure, the X-axis shows the current iteration, whereas the Y-axis shows the average value for the Cs parameter obtained in each iteration.

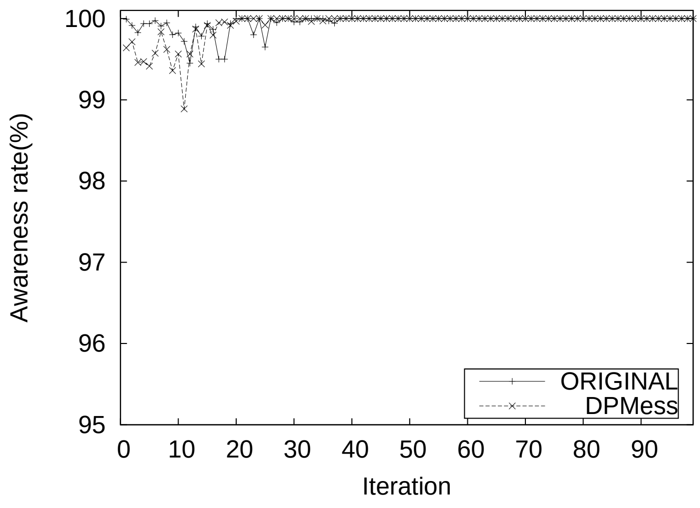

Figure 16: Awareness Rates (%) provided under a high workload

Figure 16 shows that preventing avatars from reaching saturation (by discarding messages) does not have any significant effect on the awareness rates provided to avatars when the system is under a high workload. Although the awareness rate provided by the DPMess method is slightly lower in some initial iterations, it reaches a rate of 100% and keeps on providing that rate for most of the iterations. The awareness rate provided by this method is not lower than 99% in any case.

5.3. Awareness Delay

Another important parameter that could be affected by the proposed method is the awareness delay. This parameter can be defined as the time interval from the instant when an avatar i enters the AOI of an avatar j to the instant when avatar i receives the acknowledgment from j as a new neighbor. We have denoted this parameter as A_D. This parameter is crucial, since it determines the maximum time-space inconsistencies that can arise in the system. We must study if the use of the DPMess method has any significant effect on this parameter.

Figure 17 shows the results for the awareness delay when the system is under a high workload (combination SKEWED-HPA and a new movement every 0.15 s.). This figure shows on the X-axis the iteration number, while it shows on the Y-axis the average awareness delays (the average A_D value) obtained for all the avatars in that iteration. The plots in this figure (and also the plots in the next one) only show forty iterations. The reason for this behavior is the combination SKEWED-HPA. When using this movement pattern, all the avatars tend to crowd on a single point of the virtual world. From iteration 40, no avatar enters in the AOI of another avatars, since all of them are so close among them that they can only make small movements trying to find alternative paths to their destination point. Therefore, from that iteration these small movements are no large enough to allow the avatars to enter or exit another avatars AOI.

Figure 17 shows that if the proposed method is not used, then some significant delays appear (two peaks arise in the "Original" plot). Although these peaks do not last more than several iterations, they reach an order of magnitude of several seconds. Therefore, unacceptable time-space inconsistencies can occur during some iterations. These peaks are due to the distribution of avatars and the movement pattern in these experiments. A significative number of clients reach saturation during some of the iterations, greatly increasing the awareness delay. However, when using the DPMess method (DPMess plot) these two peaks produced by the momentary saturation of some clients dissappear. These results indicate that if clients are close to saturation, then (when the messages are processed) they provide obsolete information about the location of other avatars. If messages are not processed within a given period, then it is a better strategy to discard them in order to process faster the most recent messages. In this way, the awareness delay

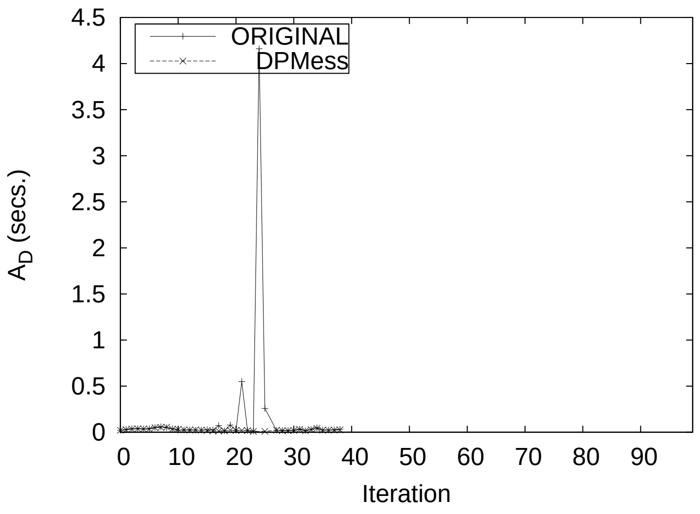

Figure 17: Awareness Delay values provided under a high workload

is kept below acceptable values during the whole simulation. Therefore, the proposed method not only does not have an effect on this parameter, but it improves the system behavior.

5.4. Discarding Rate

Another important parameter to be studied is the *Discarding Rate*, that is, the percentage of received messages that the proposed method discards. This parameter is important in order to study how the network efficiency is reduced by the DPMess method, because the more messages are discarded, the more network bandwidth is wasted. This parameter also shows the percentage of messages that should be discarded in order to avoid the system saturation. Concretely, we have defined the *Discarding Rate* D_R as

$$D_R = \frac{Discarded\,Messages}{Received\,Messages} \tag{1}$$

For the shake of shortness, we only present here the results for the combination SKEWED-HPA, that is the combination whose results has been shown when studying the rest of parameters. That is, the next figure shows the percentage of messages discarded in order to obtain the results shown in the previous subsections.

Concretely, Figure 18 shows the results obtained when the system is under a high workload (a new movement every 0.15 seconds).In this figure the X-axis shows the iteration number and the Y-axis shows the average Discarding Rate value obtained for all the avatars in that iteration.

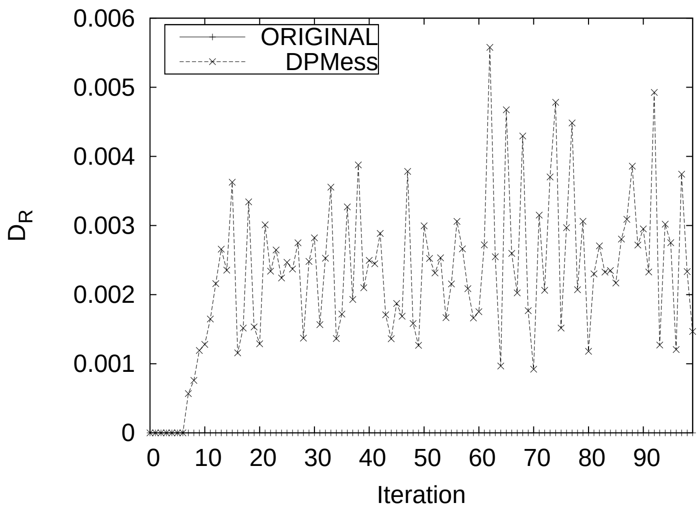

Figure 18: Average Discarding Rate values provided under a high workload

Figure 18 shows that the percentage of discarded messages by the proposed method is very low (it does not reach 0.6%). Only by discarding such a small percentage of messages, the rest of performance parameters are improved as shown above.

However, all these results can be due to the fact that the system is not under deep saturation. In order to ensure that the proposed method provides good performance regardless of the saturation level of the system, we have measured the performance when the movement rate of avatars is one order of magnitude higher. Again, we have simulated the nine combinations of initial distributions and movement patterns, although we show here only a single case, for the sake of shortness. All the results were similar. Concretely, Figure 19 shows the results for the combination of Skewed initial distribution and HPA movement pattern.

Figure 19 shows that the proposed method is efficient also under deep saturation conditions. Effectively, the plot corresponding to DPMess method has a flat slope and it reaches ASR values around 250 milliseconds, while the

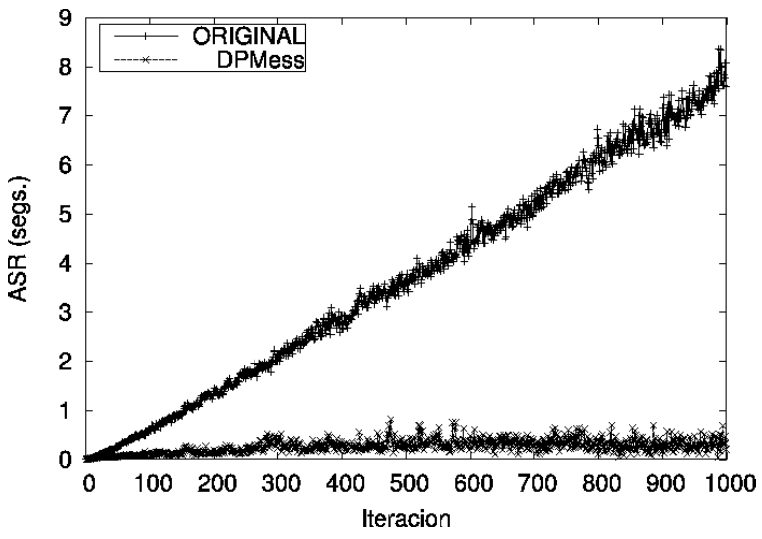

Figure 19: Average ASR values provided under deep saturation

plot labeled as original has a constant slope and it reaches values around eight seconds. When comparing Figure 15 and Figure 19 it can be clearly seen that the plot corresponding to DPMess method is similar in both figures. Therefore, the proposed method efficiently avoids saturation, regardless of the workload conditions.

6. Conclusions

In this paper, we have proposed the characterization of peer-to-peer DVEs by performing real-system measurements. The characterization results show that, unlike in networked-server architectures, the system flexibility remains unchanged with the number of client computers connected to the system. That is, the peer-to-peer scheme easily adapts to the workload generated by users behavior, since every client can also act as a server. Also, the results show that the saturation of a given client exclusively has an effect on the surrounding clients in the virtual world, having no effects at all on the rest of avatars. Finally, the characterization results show that the use of a peer-to-peer scheme does not prevent client computers from reaching saturation. That is, under certain behaviors an unbounded number of clients could reach saturation, seriously affecting the scalability and/or the performance of P2P DVEs. In order to solve this problem, in this paper we have also proposed a technique for avoiding client saturation. Unlike another existing techniques proposed for MMOGs, our proposal is focused on the state of the client computer, rather than on the application requirements, and it exclusively drops obsolete messages containing location updates. As a result, it does not depend on the application, and it can be adjusted to fulfill any given time constraint.

The performance evaluation results show that the benefits achieved by preventing client computers from reaching saturation are higher than the drawbacks of loosing information about the current state of other client computers. Thus, the selected method avoids client saturation on DVEs based on P2P architectures while maintaining the awareness rate (the percentage of correct detections of new neighbors) close to 100%, regardless of the workload generated by the users behavior. Therefore, the proposed technique can ensure the performance and the scalability of P2P DVEs.

Acknowledgment

This work has been jointly supported by the Spanish MICINN and European Commission FEDER funds under grants Consolider-Ingenio 2010 CSD2006-00046 and TIN2006-15516-C04-04.

References

[1] T. Alexander, Massively Multiplayer Game Development II, Charles River Media, 2005.

[2] World of warcraft: http://www.worldofwarcraft.com.

[3] Everquest: http://everquest.station.sony.com/.

[4] Lineage: http://www.lineage2.com.

[5] Quake: http://www.idsoftware.com/games/quake.

[6] Anarchy Online: : http://www.anarchy-online.com.

[7] J. C. Lui, M. Chan, An efficient partitioning algorithm for distributed virtual environment systems, IEEE Trans. Parallel and Distributed Systems 13.

[8] P. Morillo, J. M. Orduña, M. Fernández, J. Duato, Improving the performance of distributed virtual environment systems, IEEE Transactions on Parallel and Distributed Systems 16 (7) (2005) 637–649.

[9] N. Beatrice, S. Antonio, L. Rynson, L. Frederick, A multiserver architecture for distributed virtual walkthrough, in: Proceedings of ACM VRST'02, 2002, pp. 163–170.

[10] M. R. Macedonia, A taxonomy for networked virtual environments, IEEE Multimedia 4 (1) (1997) 48–56.

[11] P. Morillo, S. Rueda, J. M. Orduña, J. Duato, A latency-aware partitioning method for distributed virtual environment systems, IEEE Transactions on Parallel and Distributed Systems 18 (9) (2007) 1215–1226. doi:http://dx.doi.org/10.1109/TPDS.2007.1055.

[12] C. Greenhalgh, A. Bullock, E. Frecon, D. Llyod, A. Steed, Making networked virtual environments work, Presence: Teleoperators and Virtual Environments 10 (2) (2001) 142–159.

[13] R. B. Smith, R. Hixon, B. Horan, Collaborative Virtual Environments, Springer-Verlag, 2001.

[14] S. Singhal, M. Zyda, Networked Virtual Environments, ACM Press, 1999.

[15] P. Morillo, J. Orduña, J. Duato, A scalable synchronization technique for distributed virtual environments based on networked-server architectures, in: Proceedings of the 35th IEEE International Conference on Parallel Processing (ICPP'06) Workshops, IEEE Computer Society Press, 2006, pp. 74–81.

[16] L. Gautier, C. Diot, Design and evaluation of mimaze, a multi-player game on the internet, in: Proceedings of IEEE Multimedia Systems Conference, 1998, p. 233.

[17] E. Cronin, B. Filstrup, A. R. Kurc, S. Jamin, An efficient synchronization mechanism for mirrored game architectures, Kluwer Multimedia Tools and Applications 23 (1).

[18] J. Calvin, A. Dickens, B. Gaines, P. Metzger, D. Miller, D. Owen, The simnet virtual world architecture, in: Virtual Reality Annual International Symposium, IEEE, 1993, pp. 450–455.

[19] E. Frecon, M. Stenius, Dive: A scalable network architecture for distributed virtual environments, Distributed Systems Engineering Journal 5 (3) (1998) 91–100.

[20] S. Mooney, B. Games, Battlezone: Official Strategy Guide, BradyGame Publisher, 1998.

[21] S. Rueda, P. Morillo, J. M. Orduña, J. Duato, On the characterization of peer-to-peer distributed virtual environments, in: Proceedings of the IEEE Virtual Reality 2007 (IEEE-VR07), Charlotte, NC, USA., IEEE Computer Society Press, 2007, pp. 107–114.

[22] A. R. Bharambe, M. Agrawal, S. Seshan, Mercury: Supporting scalable multi-attribute range queries, in: In SIGCOMM, 2004, pp. 353–366.

[23] A. Bharambe, J. Pang, S. Seshan, Colyseus: a distributed architecture for online multiplayer games, in: NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation, USENIX Association, Berkeley, CA, USA, 2006, pp. 12–12.

[24] T. Hampel, T. Bopp, R. Hinn, A peer-to-peer architecture for massive multiplayer online games, in: NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, ACM, New York, NY, USA, 2006, p. 48. doi:http://doi.acm.org/10.1145/1230040.1230058.

[25] L. Chan, J. Yong, J. Bai, B. Leong, R. Tan, Hydra: a massively-multiplayer peer-to-peer architecture for the game developer, in: NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games, ACM, New York, NY, USA, 2007, pp. 37–42. doi:http://doi.acm.org/10.1145/1326257.1326264.

[26] H. Backhaus, S. Krause, Voronoi-based adaptive scalable transfer revisited: gain and loss of a voronoi-based peer-to-peer approach for mmog, in: NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games, ACM, New York, NY, USA, 2007, pp. 49–54. doi:http://doi.acm.org/10.1145/1326257.1326266.

[27] L. Fan, H. Taylor, P. Trinder, Mediator: a design framework for p2p mmogs, in: NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games, ACM, New York, NY, USA, 2007, pp. 43–48. doi:http://doi.acm.org/10.1145/1326257.1326265.

[28] Y. Kawahara, T. Aoyama, H. Morikawa, A peer-to-peer message exchange scheme for large scale networked virtual environments, Telecommunication Systems 25 (3) (2004) 353–370.

[29] B. Knutsson, H. Lu, W. Xu, B. Hopkins, Peer-to-peer support for massively multiplayer games, in IEEE Infocom, March 2004. (2004).
URL `citeseer.ist.psu.edu/knutsson04peertopeer.html`

[30] P. Morillo, W. Moncho, J. M. Orduña, J. Duato, Providing full awareness to distributed virtual environments based on peer-to-peer architectures, Lecture Notes on Computer Science 4035 (2006) 336–347.

[31] M. R. Macedonia, M. Zyda, D. R. Pratt, D. P. Brutzman, P. T. Barham, Exploiting reality with multicast groups: A network architecture for large-scale virtual environments, in: Proceedings of the 1995 IEEE Virtual Reality Annual Symposium, 1995, pp. 2–10.

[32] S.-Y. Hu, J.-F. Chen, T.-H. Chen, Von: a scalable peer-to-peer network for virtual environments, IEEE Network 20 (4) (2006) 22–31.

[33] C. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, On maintaining interactivity in event delivery synchronization for mirrored game architectures, Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE (2004) 157–165doi:10.1109/GLOCOMW.2004.1417568.

[34] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, A rio-like technique for interactivity loss-avoidance in fast-paced multiplayer online games, Comput. Entertain. 3 (2) (2005) 3–3. doi:http://doi.acm.org/10.1145/1063723.1063730.

[35] FIPA, Fipa agent management specification, available at http://www.fipa.org/specs/fipa00023/ (2000).

[36] IEEE, 1278.1 IEEE Standard for Distributed Interactive Simulation-Application Protocols (ANSI) (1997).

[37] F. Kuhl, R. Weatherly, J. Dahmann, Creating Computer Simulation Systems: An Introduction to the High Level Architecture, Prentice-Hall PTR, 1999.

[38] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks: An Engineering Approach, IEEE Computer Society Press, 1997.

[39] M. Oliveira, J. Crowcroft, M. Slater, Components for distributed virtual environments, PRESENCE: Teleoperators and Virtual Environments 10 (1) (2001) 56–61.

[40] S. Rueda, P. Morillo, J. M. Orduña, A comparative study of awareness methods for peer-to-peer distributed virtual environments, Comput. Animat. Virtual Worlds 19 (5) (2008) 537–552. doi:http://dx.doi.org/10.1002/cav.v19:5.

[41] T. Henderson, S. Bhatti, Networked games: a qos-sensitive application for qos-insensitive users?, in: Proceedings of the ACM SIGCOMM 2003, ACM Press / ACM SIGCOMM, 2003, pp. 141–147.

[42] F. C. Greenhalgh, Analysing movement and world transitions in virtual reality tele-conferencing, in: Proceedings of 5th European Conference on Computer Supported Cooperative Work (ECSCW'97), 1997, pp. 313–. URL `citeseer.ist.psu.edu/greenhalgh97analysing.html`

[43] M. Matijasevic, K. P. Valavanis, D. Gracanin, I. Lovrek, Application of a multi-user distributed virtual environment framework to mobile robot teleoperation over the internet, Machine Intelligence & Robotic Control 1 (1) (1999) 11–26.

[44] P. Morillo, J. M. Orduña, M. Fernández, J. Duato, On the characterization of avatars in distributed virtual worlds, in: EUROGRAPHICS' 2003 Workshops, The Eurographics Association, 2003, pp. 215–220.

[45] S. Rueda, P. Morillo, J. M. Orduña, A saturation avoidance technique for peer-to-peer distributed virtual environments, in: Proceedings of

International Conference on Cyberworlds 2007 (Cyberworlds'07), Hannover, Germany., IEEE Computer Society Press, 2007, pp. 171–178. doi:10.1109/CW.2007.9.

[46] I. Baumgart, B. Heep, S. Krause, Oversim: A flexible overlay network simulation framework, in: IEEE Global Internet Symposium, 2007, 2007, pp. 79–84. doi:10.1109/GI.2007.4301435.

[47] S. D. Webb, W. Lau, S. Soh, Ngs: an application layer network game simulator, in: IE '06: Proceedings of the 3rd Australasian conference on Interactive entertainment, Murdoch University, Murdoch University, Australia, Australia, 2006, pp. 15–22.

*** Biographies (Text)**

Silvia Rueda received the M.S. degree in Computer Engineering from the University of Valencia (Spain) and the Ph.D., from the same university, with a dissertation about "Improving the Scalability of Distributed Virtual Environments". Currently, Dr. Rueda is an Assistant professor in the Department of Informatics, at the University of Valencia (Spain). She belongs to the GREV group of this Department. Her research adresses distributed virtual environments systems, virtual reality, nature inspired algorithms and scalability.

Pedro Morillo received the M.S. degree in Computer Engineering from the University of Valencia (Spain) and the Ph.D., from the same university, with a dissertation about “Improving the Performance in Distributed Virtual Environments”. Currently, Dr. Morillo is an Associate professor in the Department of Informatics, at the University of Valencia (Spain). In this department, he belongs to the Networking and Virtual Environments group (GREV), where he focuses on the design and the development of network architectures for Distributed Virtual Environments. Furthermore, his research adresses distributed virtual environments systems, load balancing, metaheuristics and cluster computing. He served also as Visiting Scientist at Iowa State University (Ames, IO, USA) in 2004 and University of Louisiana (Lafayette, LA, USA) in 2006. For more details on his work, go to: http://informatica.uv.es/pmorillo

Juan M. Orduña received the MS in computer engineering from the Technical University of Valencia, Spain, in 1990. He worked at Telefónica de España, Manpel Electrónica, S.A. and at the Technical University of Valencia as a computer engineeer. He received the PhD. in computer engineering from the University of

Valencia in 1998. His research has been developed inside the ACCA team. Currently, he is a lecturer professor in the Department of Informatics, at the University of Valencia, SPAIN, where he leads the GREV research group (http://grev.uv.es). He is member of the HiPEAC network of excellence (http://www.hipeac.net/), and his research is currently supported by the Spanish MEC and the European Commission. It addresses Networks-on-Chip, Distributed Virtual Environments and Crowd simulations.

Jose Duato received the MS and PhD degrees in electrical engineering from the Technical University of Valencia, Spain, in 1981 and 1985, respectively. He is currently a professor in the Department of Computer Engineering (DISCA) of the same university. He was also an adjunct professor in the Department of Computer and Information Science, Ohio State University. His current research interests include interconnection networks, multiprocessor architectures, networks of workstations, and switch fabrics for IP routers. He has published more than 280 refereed papers. He proposed a powerful theory of deadlock-free adaptive routing for wormhole networks. He served as a member of the editorial boards of IEEE Transactions on Parallel and Distributed Systems and IEEE Transactions on Computers. He has served as cochair or member of the program committee for more than 40 conferences, including the most prestigious conferences in his area (HPCA, ISCA, IPPS/SPDP, ICPP, ICDCS, Europar, and HiPC). He is a member of the IEEE Computer Society.

Biographies (Photograph) Rueda

Biographies (Photograph) Morillo

Biographies (Photograph) Orduña

Biographies (Photograph) Duato