Avoiding Client Saturation in Peer-to-Peer Distributed Virtual Environments

Silvia Rueda, Pedro Morillo, Juan Manuel Orduña[1]

Resumen— **The current expansion of multi-player online games has promoted the growth of large scale distributed virtual environments (DVEs). In these systems, peer-to-peer architectures have been proved as the most scalable scheme for supporting massively multi-player applications. Nevertheless, the interactions among clients that can take place in this type of systems can lead to the temporal saturation of some of the clients. Since a client saturation has an effect on other clients, these situations limit the performance of peer-to-peer DVEs. In this paper, we propose an adaptive technique for avoiding the saturation of the client computers in DVE systems based on peer-to-peer architectures. This technique is based on monitoring the client state and discarding some of the messages received from other clients when the client is close to saturation. The evaluation results show that the proposed method improves the system performance without having an effect on the awareness rate, regardless of the movement pattern that avatars can follow. As a result, both the performance and the scalability of peer-to-peer DVEs are significantly improved.**

Palabras clave— **Peer-to-Peer architectures, Distributed Virtual Environments.**

I. Introducción

THe current expansion of multi-player online games has promoted the growth of large scale distributed virtual environments (DVEs).

Users in these systems share a 3D virtual world and can interact among them and with the elements of the virtual scene. Usually, each system user is represented inside the virtual world by an entity called *avatar*. Users control their avatars through a client computer, which should render the images of the virtual 3D environment that the user would see if he was located at that point of the virtual world. Currently, large scale DVEs can simultaneously support thousands and even hundreds of thousands clients. Clients can connect to these systems through different networks, and usually through Internet. Although DVE systems are present in many different applications,such as civil and military distributed training or collaborative design, the most extensive example of DVE systems are commercial, massively multi-player online games (MMOG)[1], [2], [3].

Peer-to-peer architectures were proposed some years ago for DVE systems. 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 have been proved as the most scalable scheme for supporting massively multiplayer applications [4]. However, these architectures must face the awareness problem. It 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 [5]. Usually, the Area Of Interest (AOI) [6] of an avatar is considered as the neighborhood for that avatar. Providing awareness to all the avatars is crucial for MMOGs. For example, if two neighbor avatars are not aware of such neighborhood, they will not exchange messages about their movements and/or changes, and therefore they will not have the same vision of the shared environment.

Recently, the expansion of MMOGs has made large scale DVE systems to become usual, and networked-server architectures seem to lack scalability to properly manage the current number of avatars that these system can support (up to some hundred thousands of avatars [3]). As a result, some studies have proposed again the use of P2P architectures [7], since these schemes are the most scalable ones (each new client also becomes a new server, thus proportionally increasing the computing bandwidth). In DVE systems based on P2P architectures, the neighborhood attribute of the different avatars must be determined in a distributed manner. In this sense, several techniques for providing awareness to avatars in P2P DVE systems have been proposed, and some of them seem to provide a full awareness rate [8], [9]. Therefore, peer-to-peer architectures seem to be the most efficient and scalable scheme for supporting large-scale DVE systems.

Nevertheless, the use of a peer-to-peer scheme does not prevent client computers from reaching saturation. A recent study shows that when the number of avatars in the system increases and they move frequently, the density of avatars in the virtual world requires some clients to process a high number of messages, in order to achieve both the 100% of the awareness rate and an acceptable time-space consistency. Depending on the computing power of these clients, such requirements can lead to the client saturation, decreasing the performance not only of that client, but also of the clients controlling the surrounding avatars [4]. Since the number of client computers that can reach saturation depends on many factors (computing power of the client, number of neighbor avatars, movement rate of avatars, etc.) but in general it is unbounded, these situations can seriously affect the scalability and/or the performance of P2P

[1]Dpto. de Informática, Universidad de Valencia, e-mail: {Silvia.Rueda,Pedro Morillo,Juan.Orduna}@uv.es

DVE systems.

In this paper, we propose an adaptive technique for avoiding the saturation of client computers in P2P DVE systems. This technique consists of monitoring the CPU utilization of the client computer. When the CPU utilization exceeds a threshold value (the client is reaching saturation), then the proposed method consists of discarding the oldest updating messages still not processed, since under such situation is very likely that they contain obsolete information and some of the most recent messages contain more updated information. The 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 proposed method avoids client saturation on DVE systems based on P2P architectures while maintaining the awareness rate close to 100%, regardless of the movement pattern and the initial distribution of avatars. Therefore, this technique can significantly improve the performance and the scalability of P2P DVE systems.

The rest of the paper is organized as follows: Section II describes the proposed method for avoiding the saturation of the client computers. Next, Section III shows the performance evaluation results obtained with the proposed method. And finally, Section IV outlines some concluding remarks.

II. Avoiding Client Saturation

The workload that a given avatar adds to a DVE system basically depends on two factors, the movement rate of that avatar and the number of neighboring avatars[10]. Therefore, the computational workload that a given client computer should support in a P2P DVE system 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 DVE systems, a given avatar *a* can be frequently surrounded by a high number of neighbors. In such situations (and depending on the awareness technique the system uses) 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 of the 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.

We have denoted the proposed method as *DPMess*, for Discarding Pending messages. 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). We have chosen to execute this algorithm just prior to the movement of the associated avatar because at that moment the client computer will have to send a new updating message to each of its neighbor avatars, still increasing the workload that the client computer supports. In order to prevent the client computer to reach saturation, the useless workload is discarded before increasing the useful workload. The pseudocode of the proposed algorithm could be the one shown in Figure 1:

```
IF CPU_current > CPU_threshold
   FOR ALL messages IN pending_msg_queue
      IF msg.type = location_update
         IF Time - msg.t_recv > t_threshold
            discard(msg)
         ENDIF
      ENDIF
   ENDFOR
ENDIF
```

Fig. 1. Pseudocode for the proposed algorithm

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 a networked-server architecture) as saturated [10]. 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.

III. Performance Evaluation

We propose the evaluation of the proposed method by simulation. We have used an evaluation methodology based on the main standards for modeling collaborative virtual environments: FIPA, DIS, and

HLA. Concretely, we have developed a simulator modeling a DVE system based on a peer-to-peer architecture. The simulator is written in C++ and it is composed of two applications, one modeling the clients and the other one modeling the central loader, to which the clients must initially connect with in order to join the system. Both applications use different threads for managing the different connections they must establish. Such connections are performed by means of TCP sockets.

Each client has a main thread for managing the actions requested by the user, and different threads for communicating with its neighbor clients. For each neighbor, two threads are executed, one for listening and the other 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 noted that once a client has joined the system, it is not necessary for that client to communicate with the central loader.

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. The virtual world is a 2D square whose sides are 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 when the acknowledgments arrive to the sending avatar then it can compute the round-trip delay for each message sent. 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). The neighboring avatars of each avatar are determined by the awareness technique. We have used the COVER method, because this technique provides an awareness rate of 100% [9].

In order to simulate a peer-to-peer DVE system in a feasible way, we used 51 personal computers interconnected by a fast Ethernet network. One of these PCs hosted the central loader, and the rest of the 50 PCs hosted the avatars in a uniformly distributed way.

We have implemented a monitoring algorithm to check the awareness rate. This algorithm consists of each client dividing its cycle time in two phases. In the first phase, clients move following a given movement pattern (described below) and they communicate their new location to their neighbors in the virtual space by exchanging messages. In the second phase, each client sends a message to the central loader containing both its new location and also which other clients it considers as its neighbors. In this way, the central loader can compute in real time the percentage of avatars that have correctly computed which other avatars are its neighbors (that is, the awareness rate). We used an AOI size of 10 meters.

We have simulated a set of independent avatars in a generic DVE system based on a P2P architecture. These avatars are located within a seamless 3D virtual world following three different and well-known initial distributions: uniform, skewed and clustered [10]. 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) [11], HP-All (HPA) [12] and HP-Near (HPN) [13]. CCP considers that all avatars in the virtual world move randomly around the virtual scene following circular trajectories. HPA considers that there exist certain "hot points" where all avatars tend to 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 2 shows the final distribution of avatars that a 2-D virtual world representing a square would show if these movement patterns were applied to a uniform initial distribution of avatars. In this Figure, avatars are represented as grey dots. 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.

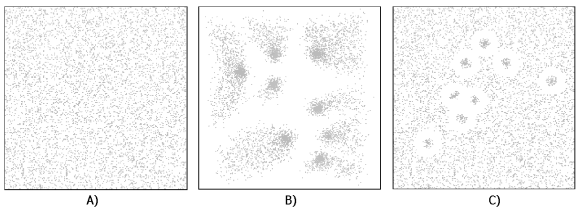

Fig. 2. Movement patterns a) CCP, b) HPN, and c) HPA

For evaluation purposes, we have used different metrics. In order to measure the overall performance of the system, we have used a well-known metric in distributed systems. Concretely, we have used the round-trip delay of the messages sent by each client computer, instead of latency. In this way, any possible clock skewing between the sending and the receiving client computer is avoided, since the sending and the receiving instants are both computed by the clock of the sender computer.

Additionally, we have studied other parameters specific from peer-to-peer DVE systems. Concretely, we have studied the awareness rate achieved in each simulation and also the 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 discarded messages 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. 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".

A. Latency

First, we have studied the system latencies (Average System Response) achieved with the proposed technique. Figure 3 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.

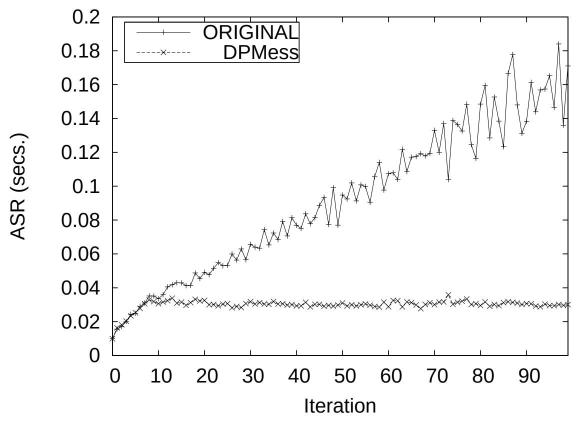

Fig. 3. Average ASR values obtained under a high workload

The plot for the DPMess method in Figure 3 shows a flat slope, keeping the average ASR values below 0.04 seconds, far away from the latency values considered as acceptable for users [14]. 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.

Figure 4 shows the results for the same system but when supporting a low workload. Concretely, it shows the results for the same combination of initial distribution and movement pattern of avatars, but in this case when avatars move at a lower movement rate (each avatar makes a new movement every 0.5seconds). In this figure, both plots are very similar, showing a flat slope and average ASR values around 0.02 seconds. The plot corresponding to the DPMess technique does not significantly differ from the Original plot, showing that the proposed method provides similar performance (in terms of latency), when the system is under a low workload.

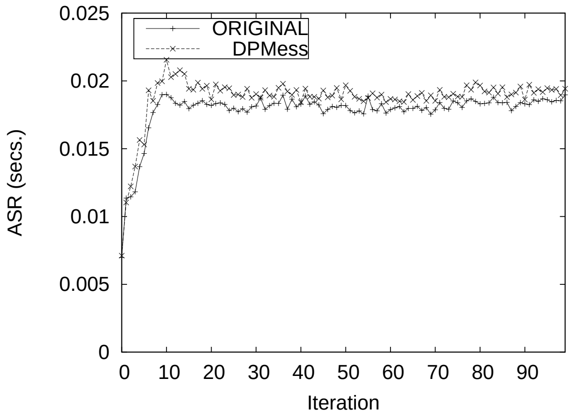

Fig. 4. Average ASR values obtained under a low workload

B. 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 DVE systems. 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 5 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.

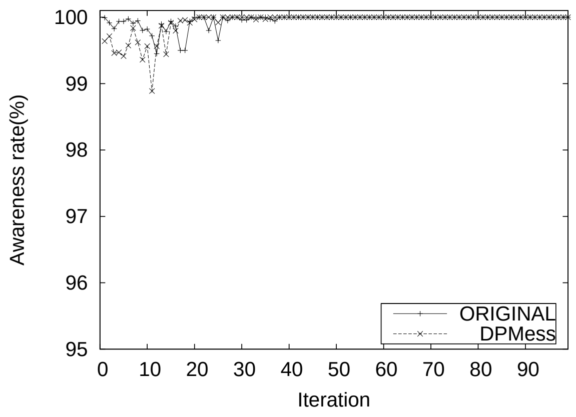

Fig. 5. Awareness Rates (%) provided under a high workload

Figure 5 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 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.

Due to space limitations, we do not show the equivalent results for the same system under a low workload. These results show that the DPMess method doesn't have a significant effect, since the CPU utilization is not high.

C. 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 6 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.

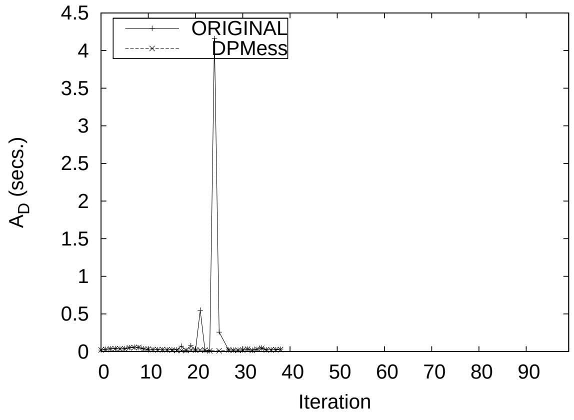

Fig. 6. Awareness Delay values provided under a high workload

Figure 6 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 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.

Although we do not show the results for a low workload due to spece limitations, they show that the DPMESS method does not have an effect on the awareness delay, because the CPU utilization is low.

D. 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

$$D_R = \frac{Discarded\,Messages}{Received\,Messages} \quad (1)$$

Due to space limitations, 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 Figure below shows the percentage of messages discarded in order to obtain the results shown in the previous subsections.

Concretely, Figure 7 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.

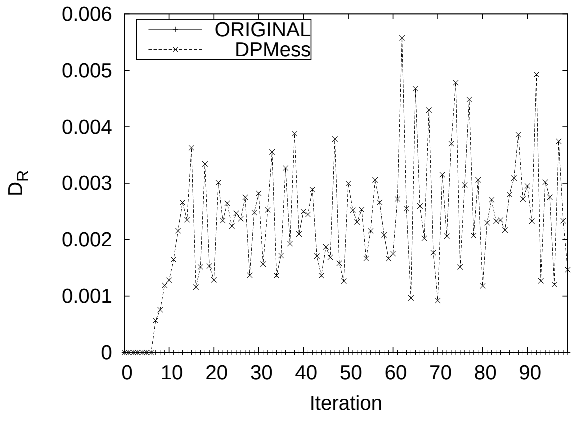

Fig. 7. Average Discarding Rate values provided under a high workload

Figure 7 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.

In this case, due to space limitations we have not shown the results obtained under a low workload, because they are trivial: no messages are discarded if the CPU utilization is not high, so the proposed method has no significant effects on the discarding rate.

IV. Conclusions

In this paper, we have proposed an adaptive method (DPMess) for avoiding the saturation of client computers in peer-to-peer DVE systems. Unlike other proposals in the literature, we have evaluated the proposed technique on actually distributed systems. We have measured the impact of the proposed technique in regard to well-known performance metrics in distributed systems. Also, we have measured some specific performance metrics for peer-to-peer DVE systems.

The evaluation results show that when the system is under a high workload then preventing client computers from reaching saturation can significantly reduce the response time offered to avatars without affecting the awareness rate, regardless of the movement pattern that avatars follow in the virtual world.

Additionally, the proposed strategy processes faster the most recent messages, in such a way that the awareness delay is kept below acceptable values. As a result, the proposed method not only does not have an effect on this parameter, but it improves the system behavior. All these improvements are achieved by discarding a very small percentage of the exchanged messages, thus not significantly wasting network bandwidth.

Since the proposed method is dynamic, it does not have any effect on system performance when the system is under a low workload. As a result, both the performance and the scalability of peer-to-peer DVEs are significantly improved.

Acknowledgment

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

Referencias

[1] ," Lineage: http://www.lineage2.com.
[2] ," Quake: http://www.idsoftware.com/games/quake.
[3] ," Everquest: http://everquest.station.sony.com/.
[4] Silvia Rueda, Pedro Morillo, Juan Manuel Orduña, and José Duato, "On the characterization of peer-to-peer distributed virtual environments," in *Proceedings of the IEEE Virtual Reality 2007 (IEEE-VR07), Charlotte, NC, USA*. 2007, pp. 107–114, IEEE Computer Society Press.
[5] Randall B. Smith, Ronald Hixon, and Bernard Horan, *Collaborative Virtual Environments*, Springer-Verlag, 2001.
[6] S. Singhal and M. Zyda, *Networked Virtual Environments*, ACM Press, 1999.
[7] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in IEEE Infocom, March 2004., 2004.
[8] Shun-Yun Hu, Jui-Fa Chen, and Tsu-Han Chen, "Von: a scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, pp. 22–31, 2006.
[9] P. Morillo, W. Moncho, J. M. Orduña, and J. Duato, "Providing full awareness to distributed virtual environments based on peer-to-peer architectures," *Lecture Notes on Computer Science*, vol. 4035, pp. 336–347, 2006.
[10] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato, "Improving the performance of distributed virtual environment systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 7, pp. 637–649, 2005.
[11] N. Beatrice, S. Antonio, L. Rynson, and L. Frederick, "A multiserver architecture for distributed virtual walkthrough," in *Proceedings of ACM VRST'02*, 2002, pp. 163–170.
[12] 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–.
[13] M. Matijasevic, K. P. Valavanis, D. Gracanin, and I. Lovrek, "Application of a multi-user distributed virtual environment framework to mobile robot teleoperation over the internet," *Machine Intelligence & Robotic Control*, vol. 1, no. 1, pp. 11–26, 1999.
[14] T. Henderson and S. Bhatti, "Networked games: a qos-sensitive application for qos-insensitive users?," in *Proceedings of the ACM SIGCOMM 2003*. 2003, pp. 141–147, ACM Press / ACM SIGCOMM.