Managing Objects in P2P DVEs

Rueda S.[1], Giannaka E. [3,4], Morillo P. [1], Bouras C. [2,3], Orduña J.M. [1]
[1]*Departamento de Informática, Universidad de Valencia, Spain*
[2]*Research Academic Computer Technology Institute, Greece*
[3]*Computer Engineering and Informatics Department, University of Patras, Greece*
[4]*Athens Information Technology Center, Athens, Greece*

***Abstract*—Peer to Peer Distributed Virtual Environment systems have become a scalable solution for supporting a large number of users. One of the main challenges for these systems is to solve the awareness problem, since it is necessary for providing a consistent view of the environment to each participant of the simulation. Although different strategies have been proposed, they exclusively focus on users. Nevertheless, most of current DVE systems include additional non-autonomous elements, denoted as objects, whose state should also be known to system clients. This paper studies the different attributes and characteristics that objects can have and can affect to their management. Based on this study, this paper presents an extension of a previously proposed avatar awareness method (COVER), re-designed and modified for providing object awareness in a distributed way. The performance evaluation results show that the resulting awareness technique allows system users to efficiently manage objects in a distributed way without affecting the overall performance of the system.**

***Index Terms*— Distributed/network graphics, Virtual Reality, Presence, Collaborative Interaction.**

I. INTRODUCTION

DVE systems are adopted in a wide range of areas, varying from civil and military training, learning and collaborative work to Multiplayer Online Games (MOGs). In a DVE system, users geographically scattered around the globe can interact with each other, inside a common scenario, in real time. One of the most important challenges that DVE systems designers need to address is to ensure that participants share the same view of the virtual world. For that purpose, all the changes performed by an avatar need to be propagated to all the avatars inside its neighborhood [23]. Usually, the Area of Interest (*AoI*) [9] of an avatar is considered as the neighborhood for that avatar.

Recent studies ([1], [8], [10], [11]) show that network delays make impossible providing all avatars of the virtual world with the same vision of the environment at every moment. In particular, these studies show that avatars need to be aware of all other avatars only inside their neighborhood. This problem is known as the awareness problem and it is a necessary condition for maintaining consistency among all users' view ([2], [18]). In any case, each new avatar represents an increase not only in the system workload but also in the amount of network traffic. In large scale DVE systems, with thousands or even millions of connected clients, scalability is one of the key aspects for the communication architecture in a DVE system. In this context, P2P architectures have emerged as the solution due to their inherent scalability [13]. However, providing avatars with a consistent view of the virtual environment in DVE systems based on P2P architectures (commonly denoted as P2P-DVE [13]) is a difficult task. The reason is that (unlike other communication architectures) P2P scheme does not have servers with information about the location of other avatars. As a result, avatars should correctly find their neighbors in a distributed way.

Existing approaches for providing awareness in P2P-DVE systems ([3], [4], [5], [6], [19]) mainly focus on the avatar entity. However, DVE systems usually include additional elements, which state should be managed and communicated to all the participants of the system. We use the term *object* for defining these non-autonomous elements. Objects are part of the virtual environment and are placed within it for serving the contextual purposes of each scenario. The behavior of the objects depends on the users' actions rather than on system decisions. Typically, objects could be weapons, books, stamina, etc. Recent contributions highlight the impact of the presence of objects on the system and the importance of managing objects in P2P-DVE systems [19].

Every change that takes place, either avatar or object related, needs to be propagated to all affected participants. For the former kind of interaction (avatar-to-avatar), the client computer controlling one of the avatars must send updating information to the client computer controlling the other avatar. Depending on the communication architecture, this message will be directly submitted (P2P) or will pass through one or more servers. However, for the later kind of interaction (avatar-to-object) the message destination depends on which avatar controls this object. In P2P-DVE systems, objects should be managed by clients in a distributed way. Moreover, in order to provide avatars with a consistent view of the virtual scenario, the awareness problem should be extended in order to take into account not only avatars but also objects.

This paper proposes an extension of a previously proposed distributed awareness method for P2P DVE systems [14] in order to take into account the presence of dynamic objects within the virtual world. The extended awareness method takes into account current studies on the concept of objects, their attributes, characteristics and the effect that they could have both on users' behavior and system's performance. Performance evaluation results show that the proposed method provides full object awareness and it does not add a significant overhead to the DVE system.

The rest of the paper is organized as follows: Section II

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants Consolider-Ingenio 2010 CSD2006-00046 and TIN2009-14475-C04-04.

studies the properties and attributes of objects, describing the problems that arise when managing objects in P2P DVEs. Section III explains the design decisions made for extending the awareness method. Section IV defines the interactions that avatars can perform with objects. Next, Section V describes the evaluation setup and shows the performance evaluation results of the proposed awareness method. Finally, Section VI presents some concluding remarks.

II. OBJECTS IN DVE SYSTEMS

Most of 3D virtual reality scenes are comprised by two types of entities, avatars and objects. The avatars constitute the graphical representation of the participating users, whose state is controlled through a client computer, while the objects represent the individual entities that compose the virtual scene. The presence of objects in a DVE system increases the complexity of the application as both the workload and the number of exchanged messages is increased. In particular, given the fact that objects in the virtual scene can affect users' behavior and can play an important role in the realization of each simulated scenario, the maintenance of a consistent state of all types of entities becomes critical. Any delayed or lost messages, containing information about changes in the state of objects, could significantly affect the level of realism, users' interactivity and perception and the general system behavior and consistency. As mentioned above, in the case of DVE systems based on networked server architectures, the management of the objects is performed in the same way that avatars are handled. However, in the case of P2P-DVE systems, the presence of objects increases the simulation complexity and introduces a number of issues that need to be handled. In particular, two aspects need to be emphasized:

Propagation of object status and changes to connected peers: All the changes that take place within the DVE should be propagated to all avatars concerned. Thus, when an avatar interacts with an object of the virtual scene by changing one or more of its attributes, then all the surrounding avatars should be informed about the performed change.

Handling awareness for concurrent interactions on the same object-Conflict Resolution: As in real world, different users could try to interact with the same object at the same time (i.e. for picking up a weapon). Allowing these simultaneous actions implies offering a high level of awareness not only to the avatars that interact with the same object, but also to all other connected peers, which are affected by the modification. In the case of networked server architectures, messages from involved avatars would reach the server that handles the objects, which in turn would be communicated to all the concerned clients. In P2P-DVE system, all the avatars involved in a modification for the same object would consider themselves as the "owners" of this object when the interaction occurs. In order to solve this conflict, the approach presented in this paper considers that the owner of the object is the last avatar that interacted with it.

In most cases, avatars that participate in a virtual environment are allowed to perform certain types of actions, such as navigating, interacting and communicating. However, the objects placed in the virtual environment can significantly differ on the type of actions and interactions that can be performed on each of them, which is related to the scope of their existence and the functionality that they aim to support. All the objects of a virtual environment can have a variety of attributes (e.g. shape, colour, position, size) and the interaction of an avatar with an object could be considered as the ability to modify one or more of these attributes. At this point it should be mentioned that the appointment of objects' importance was identified in [24], where features and characteristics of these entities were also identified.

The interactions of avatars with objects are critical and have vital importance for the awareness of the DVE and should be therefore taken into account. Furthermore, the different types and characteristics of existing objects introduce further issues that need to be faced and handled within a DVE simulation. In the subsections that follow the attributes of virtual objects (as they were introduced in [12]) are briefly described. These attributes are then used in the simulation process for handling the issues that arise.

A. Degree of Interaction

In a virtual environment, users have the ability to communicate both with each other and with the objects of the world for achieving the contextual goals that each environment aims to support. The interactions that take place within a virtual environment are significantly affected by the number and type of actions allowed and supported by the system. In particular, objects that allow a high number of actions to be performed on them tend to have a higher possibility of constituting points of interaction with the users' avatars. The number of actions that can be performed on an object may vary and can concern the modification of its location, size, shape, color, texture, etc. We define the term degree of interaction (*DoI*) for a given object as the number of actions that could be potentially performed on this object.

B. Level of Importance

Apart from the degree of interaction, we also adopt the level of importance (*LoI*) for the objects of the virtual environment. The level of importance of an object indicates that this object is often visited in a session by the participating users. Thus, even though the *DoI* value of an object may not be very high, if the object tends to constitute a point of interest for users that join the virtual environment, then this object is set with a high level of importance. The *LoI* factor in a virtual environment is affiliated to the possibility that a user will visit an object, when this object is within its Area of Interest (*AoI*).

C. Area of Interaction

Each participant in the DVE is characterized by its the Area of Interest (*AoI*), which is the region of the virtual world within which the avatar needs to be aware of all entities and activities that take place, so as to assure the awareness. Similarly, we define the term Area of Interaction for a given object (*OAI*). This attribute defines the area of the virtual scene in which the objects are interactive among themselves or with the avatars of the DVE systems. Figure 1 illustrates the

concept of *OAI*. Even though both avatar A and avatar B can see the object, only avatar B can interact with it. In most of the simulations, the *OAI* of an object is related to its size in the 3D virtual scene. In particular, larger objects tend to have wider areas of interaction, while for small objects this area is narrowed.

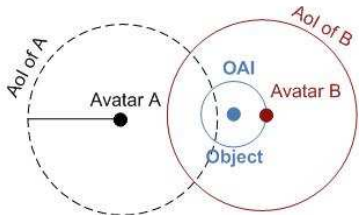


Figure 1: Objects' Area of Interaction.

D. *Objects' Classification*

The objects situated in a 3D virtual scene can be categorized regarding their level of interactivity and importance. As we mentioned before, objects' features and characteristics were identified in [24]. The approach presented in this paper, takes into account the qualitative criteria for objects and extends the original approach. Thus, based on the above, the objects of a virtual environment could be categorized as follows:

Static inactive objects: this type of objects does not support any type of interaction with the participating users. Examples of such objects could be walls, floors, etc.

Static active objects: in the case of these objects avatars have the ability to interact with them and modify one or more of their attributes, except from their position, in the sense that these objects cannot be moved within the virtual scene.

Non-static active objects: this type of objects allows the modification of all of its attributes, including the position, by the avatars they interact with. Examples of such objects could be books, swords, cups, etc.

The different types of objects involve different levels of complexity when providing a consistent view of all these objects to the participants of the DVE system. In the case of static inactive objects, users need to be aware of their presence, while in the case of static active objects the approach falls in the case that any interaction should be propagated to other peers. However, in the case of active and non-static objects (which can be transported by avatars from one place to another), the system needs to ensure that all the transitions will be performed successfully regardless the destination and origin points.

III. OBJECT AWARENESS TECHNIQUE

As mentioned above, in P2P DVE systems, there is no central entity in charge of maintaining world awareness. Therefore, it is important to define who will be responsible for object management in a distributed manner. For managing objects and assuring that all avatars of the system are aware of all objects within their *AoI*, a new algorithm has been designed and implemented. The algorithm, presented in this paper, is based on the COVER method [14], which is a tested avatar-awareness method and is proved to achieve good results for P2P DVEs. In this sense, the method presented in the paper, extends the original COVER method, on the one hand by modifying existing management methods for avatars and on the other hand, by incorporating new considerations for managing objects.

The COVER method is based on a P2P hybrid organization called *Centralized+Decentralized*, where peer nodes can play multiple roles in the DVE system, denoted as *nodes* and *supernodes*. However, COVER method does not include object management and awareness. As mentioned above, the DVE system should take into account not only objects' presence in the world but also the interactions that avatars carry out with these objects. To this direction, the COVER method has been extended for encountering the concept of objects.

As defined in the previous section, objects could be categorized as static-non-active, static-active and non-static, while only static-active and non-static objects can be updated during simulation. Avatars should be aware of both objects' location within the virtual environment and of their state. Due to the fact that the properties of static-non-active objects cannot be modified, this type of objects is not encountered for the awareness technique because this information can be easily provide to on boot time by the *Loader* or Bootstrap server, the entity in charge for the initialization of new avatars when they join the DVE system. Following the same criterion that COVER uses to distinguish between covered and uncovered avatars, objects are classified in two different categories in order to support object presence: *managed* an *unmanaged* objects. Managed objects are those which are inside the *AoI* of at least one avatar. On the other hand, *unmanaged* objects are those, which are not situated within the *AoI* of any avatar.

For providing awareness to *managed* objects, the avatars that have these objects inside their *AoI* s are t responsible of maintaining the awareness about them. In this sense, when an avatar *A* receives an updating message from a neighbor avatar *B*, it will send information to avatar *j* about the objects inside the *AoI* of avatar *B*. This mechanism is only valid if all avatars that have an object *O* inside their *AoI* are connected and send updating information between them. Therefore, the concept of neighbor avatars, used by the COVER method, has been extended and modified. Whereas COVER method considered two avatars to be neighbors (thus needed to exchange information) only if one of them was inside the *AoI* of the other, the new method considers two avatars to be neighbors when their *AoI*s intersect. We can see on the left in Figure 2, two neighbor avatars following the definition of the original COVER method. Following the original criterion, the avatars on the right image cannot be denoted as neighbor. However, given the fact that both of them have object *O* inside their *AoI,* if avatar *A* performs an action over object *O*, avatar *B* should also be informed and vice versa. Therefore, the term "neighbors" is redefined so that avatar *A* can inform directly avatar *B* of any update on the state of the objects inside its *AoI*.

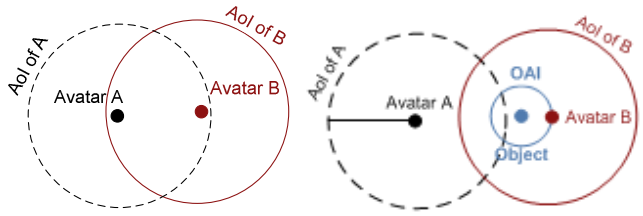


Figure 2: Avatar Area of Interaction Redefinition.

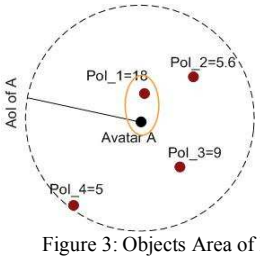
For providing awareness to *unmanaged* objects, the supernodes are used, similarly to the case of avatars. In that sense, at boot time, the Loader distributes the objects of the scene among the initial supernodes. At that time, all objects are considered as *unmanaged*, so as to inform all system avatars about the objects inside their *AoI*. Moreover, during the simulation, when the supernode detects that an avatar has an *unmanaged* object inside its region this object will become *managed* and when the supernode detects that the last *uncovered* avatar, which has an object inside its *AoI* goes away from this object, it will become again *unmanaged*. Furthermore, when the change of the supernode of a region takes place, the information about all the objects (*managed* and *unmanaged*) in the region will be sent to the new supernode. In such situations, it's possible that there is no remaining avatar in the region that could be assigned as supernode. For avatars' awareness this was not a problem because if there were no avatars, no supernode was needed. However, for managing objects this situation must be taken into account so as to assure that there will be no loss of information for the objects located inside a region without a supernode. Taking into account that this situation does not happen very often and based on the fact that no restrictive overload is introduced, the new approach overcomes this problem by sending this information to the Loader. In addition, in the original COVER method, *supernodes* also send to the Loader updating information about the quadtree structure, in order to detect supernodes failures. Similarly, in the approach presented, when the Loader detects a new supernode in an empty region, it sends to this supernode the information about all the objects inside the region.

For taking into account the interactions between avatars and objects, the *owner* of an object is defined as the avatar which is interacting with this object. Two different kinds of interactions could be performed: updating an attribute of an object (if static) or updating and moving the object (if moving). The reason for distinguishing these two kinds of interactions is due to the different levels of complexity that arise for their management. In particular, static objects do not produce any change for the region and are always controlled by the same supernode, unless a change of the quadtree structure takes place. On the contrary, moving objects could exit the region controlled by a supernode and could be added to a different region during the simulation. When an avatar get close enough to an interactive object (in a distance lower than the *OAI* described above), an interaction could be performed on this object, only if the object has no owner or if this avatar is already the owner of that object. In case the interaction is performed, the avatar should notify the change to all of its neighbor avatars. Moreover, when an avatar makes a movement, it should notify all of its neighbors about the objects which it no longer manages. Additionally, the first time an avatar enters the *AoI* of an object and the first time it undertakes an object (becomes its *owner*) and the time it releases it, it should notify the supernode about the interaction, regardless if it is a *covered* or *uncovered* avatar, because the supernode needs to be updated about the object inside its region and if they becomes *managed* or *unmanaged*. Finally, if two different avatars try to possess the same object at the same time, the algorithm assigns as owner the last one that performed the action. When given this ownership, the avatar also notifies all other avatars about it.

IV. INTERACTION SIMULATION

An object can be viewed by a given avatar when it is located within the *AoI* of this avatar. Since an avatar can potentially access to a wide variety of objects, it is necessary to define the different ways in which the avatars will interact with objects in our DVE system. We have defined a two step avatar procedure in order to carry out an interaction with an object. The first step uses the Degree of Interaction *DoI* and Level of Importance *LoI* parameters for selecting an object to interact with from the set of feasible objects included in its *AoI*. This approach exploits the attributes of the objects and defines a probability of interaction for each of them with the specific avatar. As we have already mentioned, the users of a DVE system tend to gather among objects regarding to their level of interaction and their importance. For that reason, the higher the value of the *DoI* and *LoI* parameters, the higher the probability of an interaction between avatars and objects. Furthermore, it is noted that users tend to visit objects located closer to them. Therefore, the closer the object is, the higher the probability of an avatar interacting with it. Based on the above, we have defined the *Probability of Interaction* of an object as the normalized value of those parameters (ranging from 1 to 10) in regard to the distance from the object to the avatar, as shown in (1). Thus, we consider that an avatar interacts with the object with the higher *PoI*, from the list of surrounding objects within its *AoI* .

$$PoI = \frac{DoI + LoI}{dis\tan ce} \qquad (1)$$



Obj.	*DoI* + *LoI*	Dis.	*PoI*
1	9	0.5m	18
2	7	1.25m	5.6
3	9	1m	9
4	10	2m	5

Figure 3: Objects Area of Interaction and Probability of Interaction.

Figure 3 shows an example for the different *PoI* values assigned to the objects inside the *AoI* of an avatar. The second step of the interaction process is related to how the moving avatar crosses the *OAI* of an object when it approaches. The

procedure uses the object's area of interaction *OAI* and the Degree of Interaction *DoI* for determining if the interaction can be performed or not.

V. PERFORMANCE EVALUATION

This section presents the experiments conducted for assessing the efficiency of the proposed approach. In particular, this section presents the evaluation setup along with the results obtained by the conducted experiments.

A. Evaluation Setup

In order to evaluate the performance of the proposed method, we propose the evaluation of P2P DVE systems by simulation. We have performed different experiments with a custom simulator, modeling a DVE system based on P2P architecture. The simulator is written in C++, and it is composed of two kinds of applications, one modeling the clients and the other one modeling the central *Loader*. All clients must initially join the system through the central *Loader*. Both kinds of applications use different threads for managing the different connections they must establish. Such connections are performed by means of sockets. We have simulated the behavior of a set of independent avatars in a P2P DVE system where non-autonomous entities (objects) exist. These avatars are located within a seamless 3D virtual world following three different and well-known initial distributions [13]: uniform (UNF), skewed (SKW) and clustered (CLS). Starting from these initial locations, in each simulation, avatars can move into the scene following one of three movement patterns: Changing Circular Pattern (CCP) [21], HP-All (HPA) [22] and HP-Near (HPN) [7].

Objects have been distributed in the DVE following the uniform and the clustered distribution. The reason is that in the majority of DVE systems, objects are uniformly scattered within the virtual environment or placed in certain areas of special interest. In any case, these two distributions could provide the average and the worst-case placement of objects within the virtual world. In order to analyze the impact on the system of the different kinds of objects (static active objects and non-static active objects), we have performed simulations considering two different concentrations: a) Type A encounters 50% of static active objects and 50% of non-static active objects and b) Type B encounters only non-static active objects.

For each parameter studied we have measured the results for the four combinations of the two different concentrations of kinds of object and the two distributions of objects in the environment. Furthermore, for comparison purposes, in all experiments conducted the case of having no objects was also considered. Finally, we have studied these five combinations when avatars where distributed and moving following the nine combinations of initial distribution and movement pattern described. Considering the extension of the experiments and due to space limitations we only present here some representative results.

In order to study the awareness provided for the proposed method, we have used the same monitoring algorithm used by COVER to check at runtime the awareness rate [15]. Using this algorithm the central *Loader* can determine if each avatar is aware or not of all its neighbors because at each iteration, each avatar sends information about its position and which other avatars it considers as its neighbors to the central *Loader*. We have extended this algorithm so that the *Loader* can also measure objects awareness. Concretely, each avatar also sends to the *Loader* information about the objects they consider inside its *AoI*. Each time an avatar makes an interaction on an object it sends a message to the *Loader*, so that the *Loader* can know the location of each object in the system. In this way, the *Loader* can also compute the percentage of correct object awareness made by each client.

For measuring the latency, we have used the average round-trip delay for all the messages sent by an avatar, denoted as the Average System Response (ASR) for that avatar (for that client computer). In order to measured this parameter, each time an avatar moves it sends a message to all of its neighbor avatars. Then, these destination avatars send back an acknowledgment to the sending avatar, in such a way that when the acknowledgment arrives the sending avatar can compute the round-trip delay for each message.

The experiments were performed on a cluster of workstations with 21 nodes. One of these nodes hosted the central *Loader*, and the rest of the 20 nodes uniformly hosted the clients in the system. Each node was a dual AMD 1.6GHz Opteron processor with 6 GBytes of RAM running SuSE Linux 10.1. When measuring Awareness, Latency and Communication overhead, we simulated a virtual world with 100 avatars. When measuring throughput, different world sizes (number of avatars in the virtual world) were used, rating from 100 to 1000 avatars.

B. Awareness

As described above, we have used a monitoring algorithm for measuring awareness in real time, so that the central Loader can determine whether each avatar is aware or not of all its neighbors and objects. We have separately measured the awareness rate of avatars and the awareness rate of objects. The awareness rate of avatars is the proportion between the number of neighbors that avatars have actually discovered and the number of neighbors computed by the *Loader*. The awareness rate of objects is the proportion between the number of objects that avatars have actually detected and the number of objects that they should have detected (computed by the central *Loader*).

We measured awareness for all the combinations of avatar moving pattern, initial distribution of avatars and type of simulation. For all these cases, we obtained a full awareness rate both for avatars and objects. Therefore, we can state that the proposed modifications to the COVER method provide a full awareness rate (objects and avatars) regardless the distribution and the moving pattern of avatars, the type of objects and the distribution of the objects in the virtual world.

Moreover, in order to prove the effectiveness of the proposed awareness method, it is also necessary to determine

the maximum duration of time-space inconsistencies that can arise in the system. Those, we have measured the awareness delay or time to awareness, as the time interval from the instant when an avatar *i* enters the *AoI* of an avatar *j* to the instant when *i* receives the acknowledgment from *j* as new neighbor. We have denoted this parameter as T_{AW}. T_{AW} was measured for the different combinations of object kinds concentration and distributions when avatars where distributed and moving following the 9 combinations of initial distribution and movement pattern described on the previous section. In all conducted experiments, the existence of objects slightly increased the awareness delay with respect to the absence of objects. However, this increase is not significant with respect to the average delay when no objects are considered. Moreover, we didn't appreciate any differences between moving and non moving objects nor object distribution in the scene in terms of awareness delay. So that, we can state that with this extension to the original COVER method we can grant full objects awareness rate without affecting nor the avatars awareness nor the awareness delay.

C. *Latency*

Nevertheless, the evaluation results shown in the previous subsection it is necessary to evaluate the performance of the proposed extension in terms of well-known metrics in order to prove that managing objects does not affect the system response. Concretely, we have measured the system performance in terms of latency (ASR, time response) and system throughput. Additionally, we have measured the communication overhead that supposes the handling of objects. We show some representative results from all the possible combinations of avatar moving pattern, initial distribution of avatars and type of simulation.

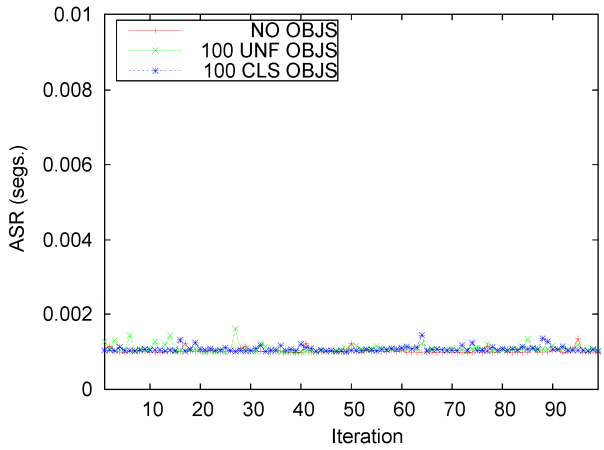


Figure 4: ASR value for UNF CCP avatars combination.

Concretely, Figure 5 shows the results for the CCP movement pattern of avatars and uniform initial distribution of avatars in the virtual world and Figure 6 presents the results when avatars follow a HPA movement pattern and are initially distributed following a cluster scheme. In these experiments, we used a Type A concentration of objects, the results obtained for the concentration Type B were very similar. Each plot on these Figures represents one of the 5 combinations of objects kinds concentrations and distributions considered. On the X-axis, this figure shows the iteration number of the simulation, and on the Y-axis it shows the average value (in seconds) of the ASR for all avatars and for five different executions.

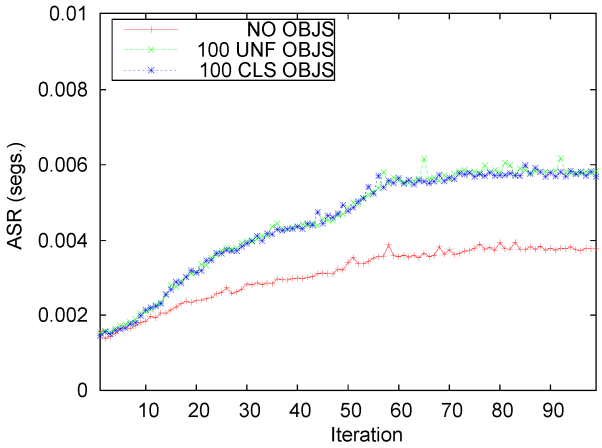


Figure 5: ASR value for CLS HPA avatars combination.

Figure 5 shows no significant differences between the case of having no objects and the rest of the cases, while Figure 6 shows that the latency increases in all the plots with respect to the case of having no objects, particularly for the type B configurations. The reason for this behavior is that the CLS-HPA combination is the one that imposes the highest computational workload (although it is not shown here due to space limitations, we measured the percentage of CPU utilization and the simulations with this configuration showed the highest values). When the system is close to saturation, the workload added by interactive objects slightly increases the latency provided to avatars. Nevertheless, the average ASR value remains far below 250 ms during the whole simulation while providing a full awareness rate. This is the threshold value for providing interactivity to users [16], [17]. Therefore, we can conclude that the increase in latency has no practical effects for users.

D. *Throughput*

We have also studied the performance achieved in terms of system throughput, that is, the number of maximum avatars that the system can support while providing acceptable latency values. In order to achieve this goal, we have grouped the average ASR values provided 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 the results for a single combination, the uniform-CCP pattern. All the cases showed similar results. It can be seen that all the plots have a flat slope, and they show values of milliseconds. These results show that, despite the proposed extension to the awareness method for managing objects, this is still scalable enough for supporting thousands of avatars.

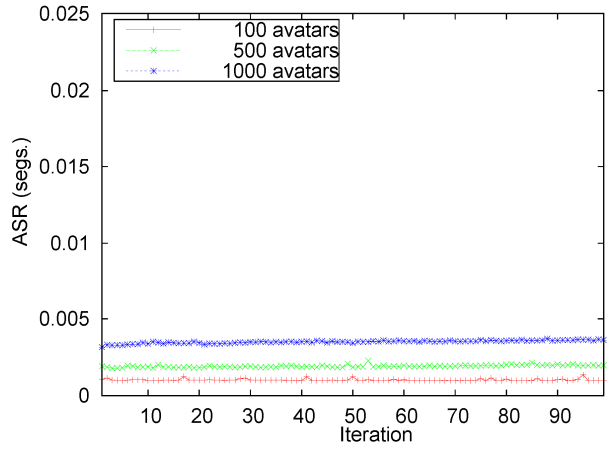


Figure 6: ASR values for different S values, UNF-CCP avatars combination.

E. *Communication Overhead*

Finally, we have evaluated the communication overhead imposed by the proposed technique. For evaluating this magnitude, we have measured the number of messages exchanged among all the clients in the system, since this metric is directly related to the computational requirements of the application [20]. In particular, we have studied the average number of messages received by any avatar in each iteration, with respect to the total number of avatars in the system, denoted as *S*. We have defined this parameter as N_{msg} and Figures Figure **8** and Figure 9 show these results for the same representative cases shown when measuring the latency.

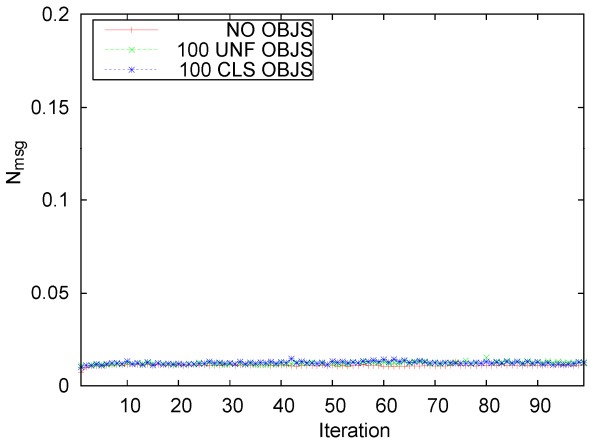


Figure 7: N_{msg} value for UNF CCP avatars combination.

Figure 8 shows that the number of messages exchanged in each configuration (plot) is not significantly higher when compared to the case of managing no objects in the DVE. However, Figure 9 shows that when the system supports a high workload, the management of different kind and amount of objects can represent a significant overhead for the proposed method. Thus, it can be seen that the plots for the type B configuration almost double the percentage of messages with respect to the plot for the case of managing no objects. The reason for this behavior is that for the CLS-HPA combination of initial distribution and movement pattern of avatars there is a high concentration of both avatars and objects in certain regions of the DVE, and as a result there is an important increase in the number of messages propagated to the peers.

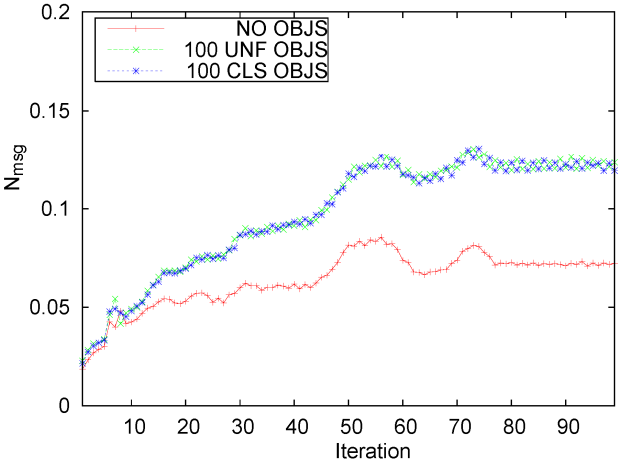


Figure 8: N_{msg} value for CLS HPA avatars combination.

Nevertheless, a comparison between Figures Figure 6 and Figure 9 shows that there is a high correlation between the latency and the percentage of message exchanged, but Figure 6 shows that for these percentages of messages acceptable latencies are provided. Thus, we can conclude that the proposed method impose a significant overhead for the worst case, but this overhead is kept below limits that ensure acceptable interactivity to users.

VI. CONCLUSIONS

In this paper, we have studied the different attributes and characteristics that objects in DVEs can have and how they affect the awareness method in P2P DVEs. Based on this study, we have extended the COVER method for taking objects into account. The performance evaluation results show that the modifications to the original awareness technique provide full object awareness with a minimal impact on system performance.

REFERENCES

[1] R. M. Fujimoto, R. Weatherly. Time management in the dod high level architecture. In Proceedings tenth Workshop on Parallel and Distributed Simulation, pages 60–67, 1996.

[2] F. C. Greenhlagh. Awareness-based communication management in massive systems. Distributed Systems Engineering, 5, 1998.

[3] S. Y. Hu, G. M. Liao. Scalable peer-to-peer networked virtual environment. In Proceedings ACM SIGCOMM 2004 workshops on NetGames '04, pages 129–133, 2004.

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

[5] J. Keller, G. Simon. Solipsis: A massively multi-participant virtual world. In Proceedings of Parallel and Distributed Processing Techniques and Applications (PDPTA), pages 262–268, Las Vegas, USA, 2003.

[6] B. Knutsson, H. Lu, W. Xu, B. Hopkins. Peer-to-peer support for massively multiplayer games. In Proceedings of IEEE InfoCom'04, 2004.

[7] 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):11–26, 1999.

[8] D. Roberts, R.Wolff. Controlling consistency within collaborative virtual environments. In Proceedings of IEEE Symposium on Distributed Simulation and Real-Time Applications (DSRT'04), pages 46–52, 2004.

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

[10] J. Smed, T. Kaukoranta, H. Hakonen. A review on networking and multiplayer computer games. Technical report, Turku Centre for Computer Science. Tech Report 454., 2002.

[11] S. Zhou, W. Cai, B. Lee, S. J. Turner. Time-space consistency in large-scale distributed virtual environments. ACM Transactions on Modeling and Computer Simulation, 14(1):31–47, 2004.

[12] C. Bouras, E. Giannaka, T. Tsiatsos. Exploiting Virtual Objects Attributes and Avatars Behavior in DVEs Partitioning. The 17th International Conference on Artificial Reality and Telexistence- ICAT 2007, Esbjerg, Denmark, 28-30 November 2007, pp. 157-163

[13] Rueda S, Morillo P, Orduña JM, Duato J. On the characterization of peer-to-peer distributed virtual environments. In Proceedings of the IEEEVirtual Reality 2007 (IEEE-VR07), IEEE Computer Society Press, Charlotte, NC, USA; 107–114.

[14] Morillo P, Moncho W, Orduña JM, Duato J. Providing full awareness to distributed virtual environments based on peer-to-peer architectures. Lecture Notes on Computer Science 2006; 4035: 336-347.

[15] Rueda S, Morillo P, Orduña JM. A comparative study of awareness methods for peer-to-peer distributed virtual environments. Comp. Anim. Virtual Worlds 2008; 19: 1–16 Published online in Wiley InterScience DOI: 10.1002/cav.230

[16] Claypool M. The effect of latency on user performance in real-time strategy games. Computer Networks 2005; 49(1): 52–70.

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

[18] Gregor Schiele, Richard Suselbeck, Arno Wacker, Tonio Triebel, Christian Becker. Consistency management for peer-to-peer-based massively multiuser virtual environments. In MMVE '08: Proceedings of 1st International Workshop on Massively Multiuser Virtual Environments, pages 14–18, March 2008.

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

[20] P. Morillo, M. Fernandez, and J.M. Orduña, "On the Characterization of Avatars in Distributed Virtual Worlds," Proc. EUROGRAPHICS 2003, pp. 215-220, Sept. 2003

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

[22] Greenhalgh FC. Analysing movement and world transitions in virtual reality tele-conferencing. In Proceedings of 5th European Conference on Computer Supported Cooperative Work (ECSCW'97), 1997; 313-

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

[24] S. Benford, L. Fahlén. A Spatial Model of Interaction in Large Virtual Environments. In Proceedings of the third conference on European Conference on Computer-Supported Cooperative, pages 109-124, Milan, Italy, 1993.