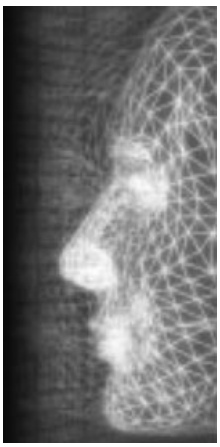
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2008; **19**: 537–552
Published online 22 July 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.230

A comparative study of awareness methods for peer-to-peer distributed virtual environments



*By S. Rueda, P. Morillo and J. M. Orduña**

The increasing popularity of multi-player online games is leading to the widespread use of large-scale Distributed Virtual Environments (DVEs) nowadays. In these systems, peer-to-peer (P2P) architectures have been proposed as an efficient and scalable solution for supporting massively multi-player applications. However, the main challenge for P2P architectures consists of providing each avatar with updated information about which other avatars are its neighbors. This problem is known as the awareness problem. In this paper, we propose a comparative study of the performance provided by those awareness methods that are supposed to fully solve the awareness problem. This study is performed using well-known performance metrics in distributed systems. Moreover, while the evaluations shown in the literature are performed by executing P2P simulations on a single (sequential) computer, this paper evaluates the performance of the considered methods on actually distributed systems. The evaluation results show that only a single method actually provides full awareness to avatars. This method also provides the best performance results. Copyright © 2008 John Wiley & Sons, Ltd.

Received: 15 June 2007; Revised: 19 February 2008; Accepted: 19 May 2008

KEY WORDS: distributed virtual environments; peer-to-peer architectures; awareness methods

Introduction

The enormous popularity of multi-player online games is leading nowadays to the widespread use of large-scale distributed virtual environments (DVEs). These systems allow users to share a 3D virtual world and to interact among them and with the virtual world. In a DVE system, each user is represented in the virtual world as an entity called *avatar*, whose state is controlled by the user through a client computer. This client computer 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 clients can be simultaneously connected to these systems through different networks, usually through the Internet. DVE systems are currently used in many different applications,[1] such as civil and military distributed training,[2] collaborative design,[3] and e-learning.[4] Nevertheless, the most extended example of DVE systems are commercial, massively multi-player online game (MMOG) environments.[5–9]

Architectures based on networked servers have been the major standard for DVE systems during last years.[10–14] 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. Figure 1(a) shows an example of a DVE based on a networked-server architecture. In this case, the system consists of 3 servers and 13 clients.

In these architectures, when a client computer modifies the state (usually the position, but it can also modify the appearance or other state 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. Servers in the DVE system must render different 3D models, perform positional updates of avatars, and transfer control information among different clients. Therefore, each new avatar represents an increase not only in the computational requirements of the

*Correspondence to: J. M. Orduña, Departamento de Informática, Universidad de Valencia, Av. Vicent Andrés Estellés, s/n., 46100 Burjassot, Valencia, Spain.
E-mail: juan.orduna@uv.es

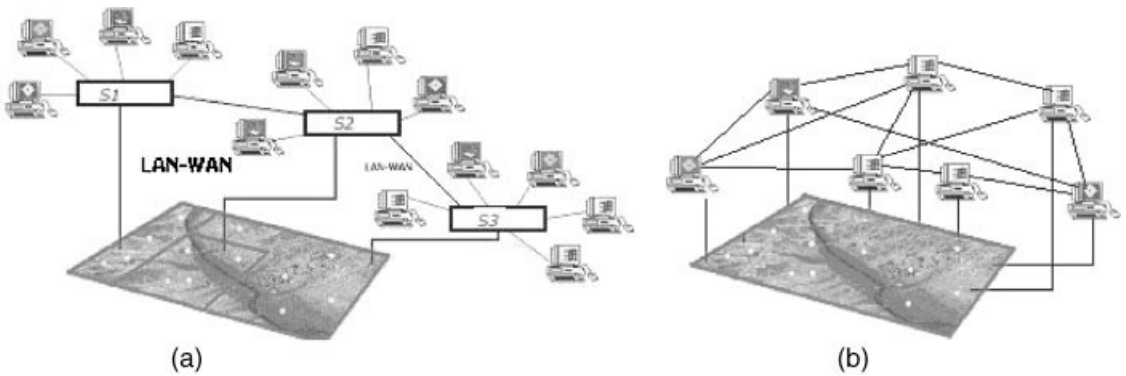


Figure 1. Example of DVE architectures: (a) networked-server and (b) peer-to-peer.

application, but also in the amount of network traffic. As a consequence, when the number of connected clients increases, the number of messages exchanged by avatars must be limited in order to avoid a message outburst. In this sense, concepts like areas of influence (AOI),[1] locales[15] or auras[16] have been proposed for limiting the number of neighboring avatars that a given avatar must communicate with. All these concepts define a neighborhood area for avatars, in such a way that a given client computer controlling a given avatar *i* must notify all the movements of *i* (by sending an updating message) only to the client computers that control the avatars located in the neighborhood of avatar *i*. The avatars located within the AOI of avatar *i* are denoted as *neighbor avatars* of avatar *i*. Usually, the AOIs of avatars are circular, but also view-dependent representations has been proposed.[17] However, although these techniques reduce both the computational and the communication requirements of the application, networked-server architectures do not properly scale with the number of existing users, particularly for the case of MMOGs (where up to hundreds of thousands of clients can be simultaneously connected to the system).[18]

Peer-to-peer (P2P) architectures have been proved to be the most adequate scheme for large-scale DVE systems, due to their inherent scalability.[19] In fact, several online games based on P2P architectures have been designed.[20–22] As an example, Figure 1(b) shows an example of a DVE based on a P2P architecture. In this scheme, each client computer is also a server.

However, P2P architectures must face the awareness problem. This 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 within its neighborhood.[23] Solving the awareness problem is a necessary condition to provide each participant with a consistent view of the environment. Effectively, 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. However, it is not a sufficient condition. Even when using an awareness method that determines at each moment which other avatars must each avatar exchange messages with, time–space inconsistencies can arise among different avatars because of clock drifts and/or network delays.[24] Awareness is crucial for MMOGs, since otherwise abnormal situations could happen. For example, a user provided with a non-consistent view of the virtual world could be shooting something that he can see although it is not actually there. Also, it could happen that an avatar not provided with a consistent view is killed by another avatar that it cannot see. Thus, providing awareness to all the avatars is a necessary condition to provide consistency (as defined in References [24–27]), but not a sufficient condition. In this sense, different synchronization proposals have been developed.[28,29]

In networked-server architectures, the awareness problem is easily solved by the existing servers, since they know the approximate location of all avatars during all the time. Effectively, the servers periodically exchange information about the location of the avatars assigned to them, using a synchronization technique.[22,30,31] Thus, when each avatar reports about its movement (by sending a message) to the server where it is assigned to, the server can easily decide which avatars should be the destinations of that message by using a criterion of distance. There is no need for a method to determine the neighborhood of avatars, since servers can easily do this task.

However, in DVE systems based on P2P architectures the neighborhood attribute must be determined

DOI: 10.1002/cav

in a distributed manner, in such a way that awareness is provided to all avatars during all the time. This is known as the awareness problem, and it is still an open issue. Currently, several strategies for providing awareness in DVE systems based on P2P architectures have been proposed.[22,32–37] Unfortunately, some of these proposals[33,35] are based on multicast communications, therefore being unsuitable for MMOGs (the most extended type of large-scale DVE systems) because Internet does not properly support multicast messages. Other proposals[32,36] do not seem to guarantee awareness to all avatars under certain movement patterns (they do not provide an awareness rate of 100%), due to the use of fixed size data structures, as described in Reference [37]. In its turn, the method proposed in Reference [37] is supposed to provide a full awareness rate, and it has been evaluated on a simulation tool.[38] However, the high number of neighbors that each avatar needs to communicate with in order to provide awareness suggests that this method is not able to provide a full awareness rate in a scalable way. In fact, the evaluation shown in Reference [38] is performed on a sequential simulation tool (executed on a single computer), and it is not performed with respect to well-known performance metrics in distributed systems like latency and/or throughput.[39] Finally, in a previous paper we proposed another full-awareness method for P2P DVEs.[34] Although this technique is able to provide a full awareness rate to avatars regardless of the movement pattern they follow, the performance evaluation shown in that paper was also performed on a sequential simulation tool.

In order to design truly efficient P2P DVE systems, the impact that the awareness (and other) computations and communications have on real system performance must be measured. There are a lot of situations in a real distributed system that do not arise in a sequential simulator, because it is executed on a single computer and therefore time–space inconsistencies due to network latency, computer delays, clock drifts, etc., do not actually exist. Only a distributed simulator is able to reproduce such situations.

This paper presents, in an unified manner, a comparative study of all the awareness techniques that seem to potentially provide a full awareness rate, in regard to well-known performance metrics for distributed systems. Concretely, we have considered the COVER technique,[34] the VON method,[37,28] and the technique proposed in Reference [32]. Unlike the evaluations shown in these previous proposals, the performance evaluation shown here has been performed on a distributed P2P DVE simulator. The evaluation results show that only the COVER technique actually provides a full awareness rate to avatars, regardless of their movement pattern and the population size. Additionally, this technique also provides the best results in regard to the well-known performance metrics, thus becoming the best option to achieve actually efficient P2P DVEs.

The rest of the paper is organized as follows: the next section analyzes the existing proposals for providing awareness in DVE systems based on P2P architectures and it also describes the techniques that seem to provide a full awareness rate. Then, the next two sections describe the experiments we performed in order to evaluate the considered techniques and the evaluation results, respectively. Finally, the last section presents some concluding remarks and future work to be done.

Awareness Methods

The expansion of MMOGs has made large-scale DVE systems to become popular, and networked-server architectures seem to lack scalability to properly manage the current number of avatars that these systems can support (up to some hundred thousands of avatars.[5]) As a result, some studies have proposed again the use of P2P architectures,[32,33,35,36,38] since these schemes seem to be the most scalable ones.[19] However, the awareness problem must be completely solved rather than a P2P architecture can be used to efficiently support large-scale DVE systems.

Some proposals use multicast communications to guarantee awareness.[33,35] Although multicast greatly improves scalability, it is hardly available on the Internet, which is the natural environment for multi-player online games. Therefore, this scheme cannot be used in most of the large-scale DVE systems.

The method proposed in Reference [32] is capable of providing an awareness rate of 95%. Since it is almost a full awareness rate, we have also considered this technique for evaluation purposes. Additionally, the VON method proposed in References [37,38] seems to provide full awareness. Although it requires that each avatar communicates with a high number of avatars (therefore suggesting that this is not a scalable technique with the number of avatars), we have also considered this technique for evaluation purposes. In this section, we describe these techniques as well as the COVER technique, since it also provides a full awareness rate when evaluated on a sequential simulation tool.

DOI: 10.1002/cav

Kawahara Method

Since it does not seem realistic to establish connections among all the avatars in the system, this method[32] tries to limit the number of connections established by a client. In order to achieve this goal, each client should maintain a *list of active entities* (AE) to which it must notify every change it performs. In order to determine the list of AE, this method uses a criterion of Euclidean distance.

Concretely, in this method each client computer should initially classify all its neighbor entities into two different groups, constructing a list for each of them: the list of AE and the list of latent entities (LE). Each client computer should establish the size of these lists depending on the computing bandwidth.

The client computers should initially establish a connection with each of the client computers in the list of AE, and they should exchange information about the changes performed as the simulation proceeds. The purpose of this method is that when an avatar *i* enters the AOI of a given avatar *j* (thus avatar *i* becoming a new neighbor of avatar *j*) this situation is detected by the current *j* neighbors (i.e., the current AE in *j*). Therefore, in this method each time an avatar performs an action it should notify its current location and the locations of all its AE to the client computers in the list of AE. With this information, every avatar should update the list of the AE in order to include the *N* closest avatars in the virtual world.

That is, in this method the awareness of a given avatar is implemented by the surrounding avatars. However, this method shows different limitations. On the one hand, the list of AE in each avatar is limited to a given value *N*. If more than *N* avatars approach to a given avatar, then the proposed method cannot properly manage that situation. That is, the scalability of the method is limited. An example of this situation can be seen in Figure 2(a), where the avatar A is surrounded by 13 neighbors (there are 13 neighbor avatars in the AOI of A). If $N = 10$, then avatar A can only be aware of 10 of these avatars.

On the other hand, the exchange of the lists of AE for each avatar movement adds a huge traffic overhead, also limiting the scalability of the method. Finally, the awareness rate for a given avatar *i* exclusively depends on the spatial distribution of its neighbors in the virtual world. If all the neighbors are located on the same side of *i*, then this awareness method fails. Effectively, in that situation if a new avatar approaches *i* by the opposite side then the neighbors of *i* cannot detect it before it enters the AOI of *i*. Therefore, in this case a correct awareness is not provided. As an example, Figure 2(b) shows an initial situation where the neighbor avatars of A are located in such a way that an important area of the AOI of A is not watched by any of the neighbors. As a result of this situation, Figure 2(c) shows how avatar G enters the AOI of A and A cannot not be aware of this fact.

VON Method

The VON technique[37,38] also tries to limit the number of connections that each client computer should establish. Additionally, it deals with the problem of leaving certain areas where new avatars can approach without being detected. In order to achieve these goals, this method proposes the use of Voronoi diagrams[40] for partitioning the virtual world in different regions, each one controlled by a given client computer. Each avatar should communicate with the avatars located either in the adjacent regions or in the AOI of the avatar.

Concretely, in this technique each avatar (using Voronoi diagrams) classifies the surrounding avatars as *enclosing* neighbors (those neighbor avatars in the adjacent regions of the Voronoi diagram) and/or

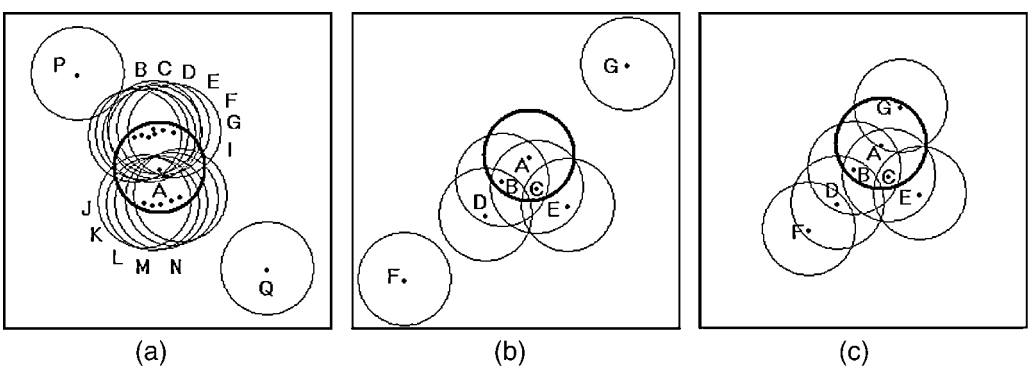


Figure 2. Different problems of the Kawahara awareness method: (a) excessive neighbors (b) non-overlapped AOI (c) incorrect detection of avatar G.

DOI: 10.1002/cav

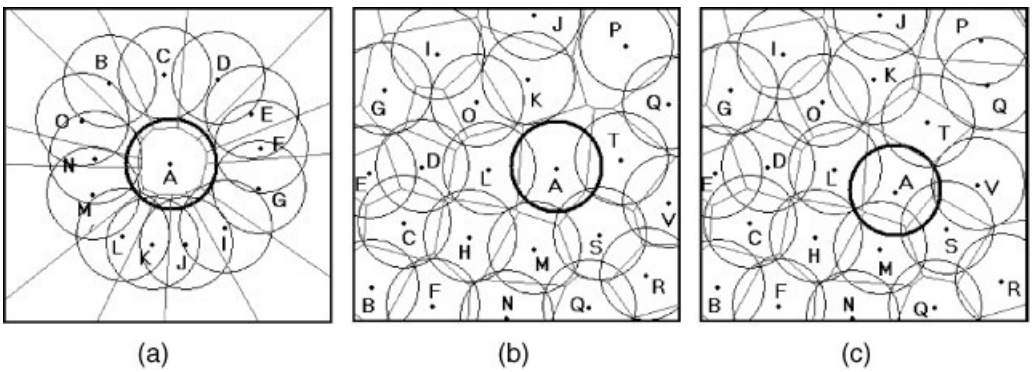


Figure 3. Different problems of the VON awareness method: (a) Distant neighbors (b) initial neighbors (c) final neighbors.

boundary neighbors (those neighbor avatars whose AOI intersects with the regions controlled by that avatar). Each avatar should construct and maintain a Voronoi diagram using the information about the location of its neighbors (the avatars inside its AOI). The detection of the approaching of new avatars depends on both the boundary and the adjacent neighbors.

When an avatar *i* moves, it should notify all its boundary neighbors about its movement. The boundary neighbors should check if at new location either *i* or the AOI of *i* will intersect with any of the regions of their adjacent neighbors. In such case, it should notify *i* about that situation.

In order to compute the Voronoi diagram, each avatar should communicate with all the clients sharing the same boundary neighbors, even though these clients are far from its AOI. These communications can add a significant overhead, suggesting that this method could lack of the required scalability. As an example, Figure 3(a) shows a situation where, according to the VON method and due the Voronoi diagram, avatar A should exchange with all its 13 surrounding avatars, although none of them are inside the AOI of A. Additionally, Figure 3(b) shows an initial situation where avatar A has established some connections with surrounding avatars. According to the VON method, Figure 3(c) shows that if some of the neighbors slightly move, then avatar A should close several connections with some avatars and it should establish new connections with other avatars. Closing and setting up a high number of connections can lead to system saturation, as shown in Reference [19].

COVER Method

The main weakness of the VON method and the Kawahara method is that both of them define a flat awareness hierarchy, where all the clients in the simulation are peers and the awareness relies on peer computers. As a result, when a given client has a lot of peers in its surroundings, it must exchange a lot of control messages with these peer computers. This overhead significantly decreases the overall performance of the awareness technique. Additionally, the inverse situation is not properly solved. Effectively, when a given client *c* has only a few neighbor clients, it may happen that these neighbors are located in such a way that they cannot watch the entire perimeter of *c*. In order to address this weakness, a hierarchical awareness technique is needed, in such a way that different strategies are used for different avatars, depending on the number of neighbor avatars.

We recently proposed a new awareness method,[34] denoted as COVER, with the purpose of providing a full awareness rate in a scalable way. This approach is based on a P2P hybrid organization called *Centralized+Decentralized*[41] or *Partially Centralized*,[42] where peer nodes (clients of the simulation) can play multiple roles in the DVE system (therefore implementing the role of different hierarchies). COVER uses a two-level awareness hierarchy, and it classifies avatars in two categories: covered or uncovered. However, as explained below, the number of avatars in the upper hierarchy can dynamically change as needed, therefore providing the required scalability.

In order to provide awareness to a given avatar *i*, the COVER method uses all the avatars surrounding *i* up to the second level of neighborhood, like the VON method proposed in Reference [37]. The first-level neighbors of an avatar *i* are those avatars in the DVE system in whose AOI avatar *i* appears. The second-level neighbors of *i* are all the neighbor avatars of the first-level neighbors of *i*. In order to avoid cyclic relationships (redundant messages), if a second-level neighbor is also a first-level neighbor, then it is not considered as a second-level neighbor. Each

DOI: 10.1002/cav

time an avatar *i* moves, it sends an updating message to each of its first-level neighbors. These neighbors in turn propagate the updating message of *i* to the second-level neighbors of *i*.

However, unlike the Kawahara and VON methods, COVER classifies avatars in two categories: covered or uncovered. This classification determines the behavior of P2P clients in order to get updated (consistent) awareness information. We denote an avatar *i* as *covered* if its first-level neighbors are located in such a way that the intersections of their AOIs totally cover the AOI of *i*. Otherwise, it is considered as an *uncovered* avatar. That is, when *i* has enough neighbors, these neighbors can watch the entire perimeter of *i* (*i* is then a covered avatar). Effectively, since the updating messages sent by any avatar arrives to its second-level neighbors, COVER method offers auto-awareness to covered avatars, because no avatar can approach them without being detected by their neighbor avatars. This mechanism for providing awareness to covered avatars is similar to the Kawahara and VON methods. If a given avatar is covered, then the condition of being a covered or uncovered avatar is computed by the client computer after making each movement. If the avatar is uncovered, then the condition of covered or uncovered is computed by the client computer after making each movement and also when receiving the location update of a neighbor avatar. It should be noticed that once the AOI of a given avatar is covered with some AOIs intersections, the rest of the neighbors AOIs are irrelevant. Thus, the cost of performing this computation (covered or uncovered) do not heavily depend on the number of neighbors, but how these neighbors are located. In our experiments, we have limited the number of avatars hosted by a client (20 avatars hosted by each client) taking into account this overhead. However, it should be noticed that in real P2P systems there is a single avatar in each client computer, and therefore this overhead is not significant.

However, it can happen that at a given moment a given avatar does not have enough neighbors around it to be provided with awareness (that avatar is uncovered). Unlike the Kawahara and VON methods, the COVER method provides awareness also for uncovered avatars, by means of the *supernode* avatars. These avatars play multiple roles, acting not only as simple avatars but also as pseudo-servers.[1] Supernodes represent an upper layer in the awareness hierarchy, and they provide the required scalability while ensuring an awareness rate of 100%. Supernodes are initially designated by the entity in charge of the initialization of new avatars (denoted as *Loader*[43] or Bootstrap server[32]) when they join the DVE system. At boot time, the Loader divides the 3D virtual scene into square sections called *regions*. For each region, the closest avatar to the geometric center of each region is selected by the Loader as the supernode for that region. From that instant, supernodes are responsible of providing awareness to those uncovered avatars that are located within their regions. Uncovered avatars must send their updating messages not only to their neighbors, but also to the corresponding supernode of the region where they are located. In this way, supernodes can notice uncovered avatars when another uncovered avatar(s) cross their AOI. The auto-awareness of covered avatars avoids the need of supernodes to notice uncovered avatars about the movement of covered avatars, significantly reducing the communications required for providing awareness.

As an example, Figure 4(a) shows a 2D region containing five avatars, represented as dots, and their respective AOIs, represented as circumferences around the dots. In this region, avatars B–E are uncovered avatars. Since the circumference around avatar A is totally covered by the AOIs of avatars D, C, and E, avatar A is classified as a covered avatar. Also, in this case avatar A has been

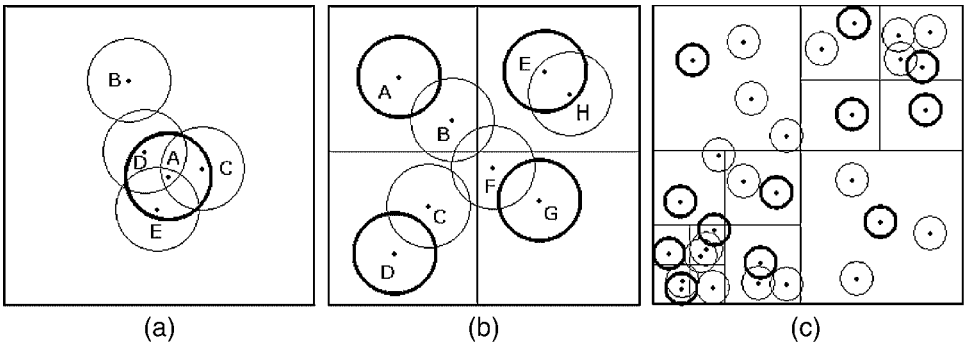


Figure 4. An example of: (a) a covered avatar being the supernode at the same time, (b) division of the virtual world, and (c) division of the same region into more subregions.

DOI: 10.1002/cav

chosen as supernode of the region (we have represented supernodes by depicting their AOI with a thicker circumference), and therefore this avatar will receive updating messages from all the uncovered avatars in this region (the rest of the avatars).

Although the client computers controlling the supernodes act as mirrored servers do in a server network architectures,[1] they are client computers of a DVE system based on a P2P architecture. Therefore, they should not be significantly overloaded by the awareness mechanism. Instead of using additional hierarchy levels for providing awareness in a scalable way, the COVER method uses a quad-tree segmentation of the virtual scene[44] to avoid the saturation of supernodes, adding new supernodes as necessary (and therefore providing scalability). Concretely, COVER limits the maximum number of uncovered avatars which are simultaneously connected to the same supernode. This parameter is called *MNUA*, (for maximum number or uncovered avatars). Whenever MNUA is exceeded, the supernode divides that region in four different subregions and computes a new supernode for each subregion, based on the criterion of geometric distance to the center of the subregion. Once the division has been performed and a new supernode is selected for each subregion, the uncovered avatars in each subregion are re-assigned to the new supernodes. It is worth mentioning that the criterion used for selecting new supernodes does not distinguishes between covered or uncovered avatars. When the system is running, this mechanism defines a dynamic quad-tree structure where each supernode has four sons. We have chosen the quad-tree structure because it has been proved as an efficient data structure for covering rectangular areas.[45–47] Two or more supernodes are *brothers* if they have been generated in the same division operation. Brother nodes automatically set up priorities among them when they are created, and the supernode with the highest priority periodically checks if all of the brother supernodes have less avatars in their region than one-fourth of the MNUA value. If it is the case, then a fusion operation is performed. In this operation, the four brother subregions are joined to become a unique, larger region, and a new supernode for the new region is computed based on the topological priority criterion. Unlike the VON method, this mechanism provides the required flexibility for situations such as avatars heading to the same location of the virtual world, as shown below. Additionally, it exploits the inherent scalability of the P2P scheme[19] to provide awareness to all avatars in a scalable way.

As an example, Figure 4(b) shows the evolution of Figure 4(a) when the proposed scheme is applied and the MNUA parameter has value of six avatars. Since three more avatars have joined the system and the MNUA value has been exceeded, the supernode has divided the region in four subregions. In this case, the resulting supernodes are now avatars A, D, E, and G. These brother supernodes monitor the total number of uncovered avatars in the zones that they control. COVER method does not require supernodes to exchange the position of the avatars in their respective regions, like networked-server architectures do.[1,12] This key issue allows the COVER method to use as many supernodes as needed without greatly increasing the amount of messages required to provide awareness.

In order to offer full awareness to those avatars located at the borders of different regions (denoted as *critical* avatars), *secondary* supernodes are used. Critical avatars are defined as those uncovered avatars whose AOIs intersect with more than a single region. Critical avatars should send updating messages not only to the supernode managing the region where they are located, but also to the supernodes managing the adjacent regions. These supernodes manage uncovered avatars located in different regions, and they are considered as secondary supernodes for critical avatars. Figure 4(b) shows that avatars B and F must be considered as critical avatars, because the area of their AOI exceed the limits of the subregion where they are located. The solution for this situation is to force B to send the updating messages not only to supernode A, but also to supernodes D and E, as discussed above (actually, supernode A is the one in charge of re-transmitting the updating messages from B to supernodes D and E). In the same way, avatar F must send updating messages to the four supernodes.

Following the above example, Figure 4(c) shows the result of the proposed awareness scheme when it is applied to a larger DVE system composed of 30 avatars. It shows the behavior of the proposed technique and how the different levels of the quad-tree structure are dynamically generated. In this figure, there are several avatars whose AOI intersect with different regions or subregions. These are critical avatars.

In order to show that the proposed method can be actually used in real P2P applications, Figure 5 shows a snapshot of simulation. This simulation is a P2P version of the Terra3D program[48] using the COVER method.

Characterization Setup

In this section, we present the performance evaluation of the COVER method, compared with another two

DOI: 10.1002/cav

Figure 5. An example of a simulation based on a P2P system using the COVER awareness method.

awareness techniques. We propose the evaluation of P2P DVE systems by simulation, as it was done in our previous work.[34]

However, as discussed in the introduction, the performance evaluation shown here is made by an actually distributed P2P DVE simulator. Additionally, we have extended our previous work[34] by adding the evaluation of the technique evaluated in Reference [38] in regard to latency and throughput.

The evaluation methodology used is based on the main standards for modeling collaborative virtual environments, FIPA,[49] DIS,[50] and HLA.[51] Since DVE systems 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.[39] 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 developed a simulator modeling a DVE system based on a P2P 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 that they must establish. These connections are performed by means of sockets.

Each client has a main thread for managing the actions asked 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 noted that once a client has joined the system, it is not necessary for that client to communicate with the central Loader. Since the goal of this characterization is to study how the system evolves as clients move rather than analyzing how new clients join the system, in our simulator each client is initially provided with the IP addresses of its initial neighbors.

A simulation consists of each avatar performing 100 iterations. 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. In our experiments, the virtual world is a 2D square whose sides are 200 m 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 acknowledgment arrives the sending avatar 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 neighbor avatars of each avatar are determined by the awareness technique. For comparison purposes, we have implemented the COVER method as well as the methods proposed in References [32] and [38].

We have used a cluster of 12 nodes. One of these PCs hosted the central Loader, and the rest of the 11 PCs hosted the clients in the system. Each node was a dual AMD 1.6 GHz Opteron processor with 6 Gbytes of RAM running SuSE Linux 10.1.

In order to study the efficiency of the awareness methods, we have implemented a monitoring algorithm to check the awareness rate. In this case, the 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 in the virtual space by exchanging messages. In the second phase, each client sends the central Loader a message containing its new location and also the identifications of the clients that it considers as its neighbors. Thus, the central Loader can compare the

DOI: 10.1002/cav

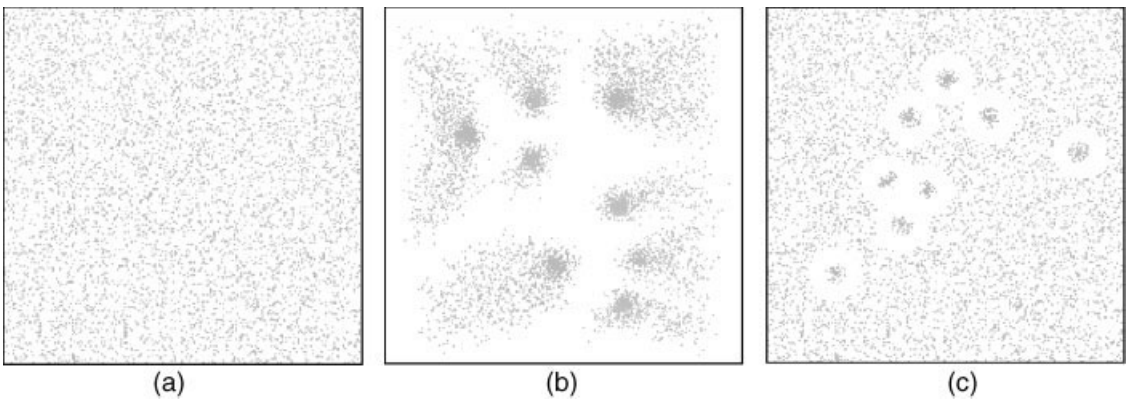


Figure 6. Final distribution of avatars for (a) CCP, (b) HPN, and (c) HPA movement patterns applied to an initial uniform distribution of avatars.

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 dedicate to the second phase. Finally, we used an AOI size radius of 10 m and an MNUA value (for the case of the COVER method) of 15 avatars. We chose the AOI radius size of 10 m because the Kawahara method provided the best performance results with this value. Also, this is the value commonly used in the literature. Regarding the MNUA value, the reason is that we tested different values, and the best results were obtained with a value of 15 avatars.

In order to study the system performance (ASR and throughput), we used a different characterization setup. In this case, we eliminated the second phase and then avatars did not send any information to the Loader computer. We also reduced the movement cycle to one movement each 2.1 seconds but we maintained the AOI radius size in 10 m. In this case, we only used 11 of the 12 nodes of the cluster, one for the Loader computer, and the other 10 nodes for hosting regular avatars.

For each of the awareness methods considered, we have simulated the behavior of 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[18] following three different and well-known initial distributions: uniform, skewed, and clustered.[10,11] 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),[12] HP-All (HPA),[52] and HP-Near (HPN).[53] 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. This pattern tests the considered techniques for those situations where the density of avatars in a certain region greatly increases. 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 6 shows the final distribution of avatars that a 2D virtual world (represented as a square) 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. It must be noticed that, as shown in Reference [54], these combinations of initial distributions of avatars and movement patterns represent an upper limit of the workload generated in real DVE systems.

Evaluation Results

In this section, we present the performance evaluation results of the proposed technique. First, we present the evaluation in terms of the awareness rate, next we present the evaluation in regard to latency and throughput. Finally, we have measured the round-trip delay between the instant when a new neighbor enters the AOI of a given

DOI: 10.1002/cav

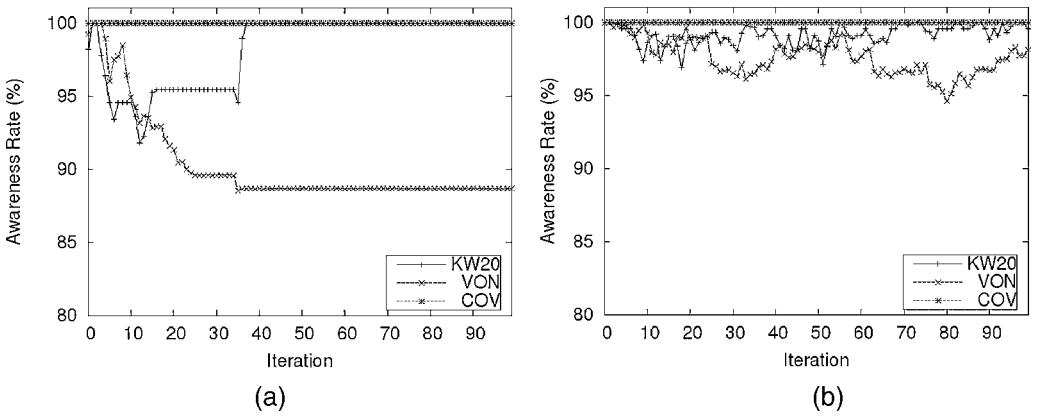


Figure 7. Awareness rate comparisons for UNF CCP pattern with (a) 110 avatars and (b) 220 avatars.

avatar and the instant when that avatar knows about that neighbor. For comparison purposes, we show in this section the results for the COVER method (plots labeled as COVER) as well as for the methods proposed in Reference [32] (plots labeled as KW20, that stands for Kawahara method enhanced to 20 AE) and in Reference [38] (plots labeled as VON).

Awareness Rate

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 is aware or not of all its neighbors. We have measured the proportion between the number of neighbors that should exist and the number of neighbors that avatars have actually detected. We have denoted this parameter as A_R (standing for Awareness Rate).

Figures 7(a) and 7(b) show some representative results obtained in the cluster simulator. Concretely, these figures show the simulations for population sizes of 110 and 220 avatars, respectively. In these simulations, the avatars followed a CCP movement pattern, starting from a Uniform initial distribution. In these figures, the X-axis shows the current iteration whereas the Y-axis shows the average value for the A_R parameter obtained in this iteration. In all simulations, we used an AOI of 10 m and a movement rate of 3.95 seconds. In each figure, there are three plots (one for each awareness method), as described above.

Figures 7(a) and 7(b) show that the only method that provides an awareness rate of 100% is the COVER method. Neither the VON method nor the KW20 method actually provide a full (100%) awareness rate.

In order to prove that the behavior shown in these figures does not depend neither on the movement pattern of avatars nor on the initial distribution of avatars, we have tested the nine combinations of movement patterns and initial distributions. However, for the sake of shortness we only show here the results for some of these combinations. We have obtained very similar results for the rest of combinations. Concretely, Figures 8(a) and 8(b) show the results obtained for a skewed initial distribution of avatars following an HPA movement pattern. These figures correspond to population sizes of 110 and 220 avatars, respectively.

Figures 8(a) and 8(b) show that again the only method that actually achieves a full awareness rate is the COVER method. When comparing these figures with Figures 7(a) and 7(b), the main difference is that the awareness rate achieved by the VON method is lower in the former ones.

Finally, Figures 9(a) and 9(b) show the results obtained for the same populations sizes, but when the simulations start from a skewed initial distribution of avatars and these avatars follow an HPN movement pattern. The results shown in these figures are similar to the ones shown in the rest of the figures. Therefore, we can conclude that only the COVER method provides a full awareness rate for any population size, regardless of both the movement pattern and the initial distribution of avatars.

Latency

Nevertheless, the evaluation results shown in the previous subsection do not prove that the COVER method can actually improve the performance of P2P DVEs. It is

DOI: 10.1002/cav

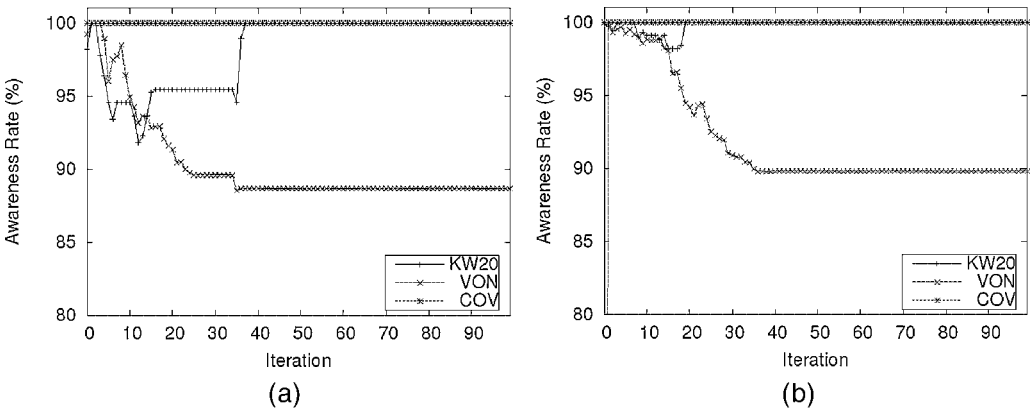


Figure 8. Awareness rate comparisons for SKW HPA pattern with (a) 110 avatars and (b) 220 avatars.

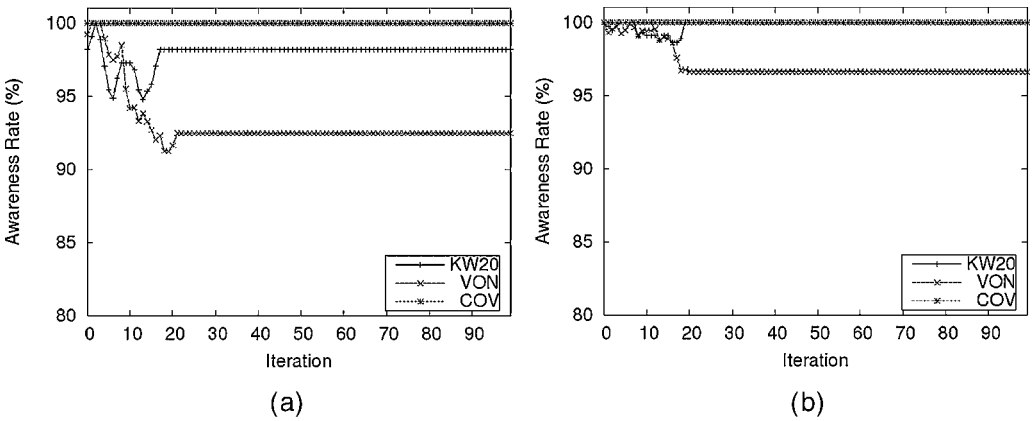


Figure 9. Awareness rate comparisons for SKW HPN pattern with (a) 110 avatars and (b) 220 avatars.

necessary to evaluate the performance of these systems in terms of well-known metrics when different awareness methods are used. Concretely, we have measured the system performance in terms of latency (ASR, time response) and system throughput. Additionally, we have measured the awareness delay, that is, the time interval from the instant when a new neighbor enters the AOI of an avatar until the instant when that avatar is aware of the new neighbor.

We have accomplished more than 4000 experiments, in which we studied these parameters for the nine combinations of movement patterns and initial distributions of avatars. Also, we performed simulations with different populations sizes (different numbers of avatars). Nevertheless, for the sake of shortness we present here only the results for a given combination and for population sizes of 100, 500, and 1000 avatars. The rest of the results were very similar to the ones shown here.

Concretely, Figure 10 shows the results for a population size of 100 avatars when these avatars follow a CCP movement pattern starting from a uniform initial distribution. In this case, the *X*-axis shows the iteration number and the *Y*-axis shows the average ASR value obtained for all the avatars in that iteration.

Figure 10 shows that all the considered methods provide acceptable ASR values that are far below the threshold values of hundreds of milliseconds considered as the maximum values that are acceptable for users.[55,56] However, the COVER method shows the plot with the lowest ASR values. In order to study the ASR values provided to larger populations, Figures 11 and 12 show the results for the same simulations but now performed with populations of 500 and 1000 avatars.

Figure 11 shows that the plot for the VON method adds an overhead that results in the system saturation, in such a way that the average ASR provided to avatars continuously increases during the simulation. It is worth mentioning that the values provided by this method are of tens and even hundreds of seconds. Although the KW20

DOI: 10.1002/cav

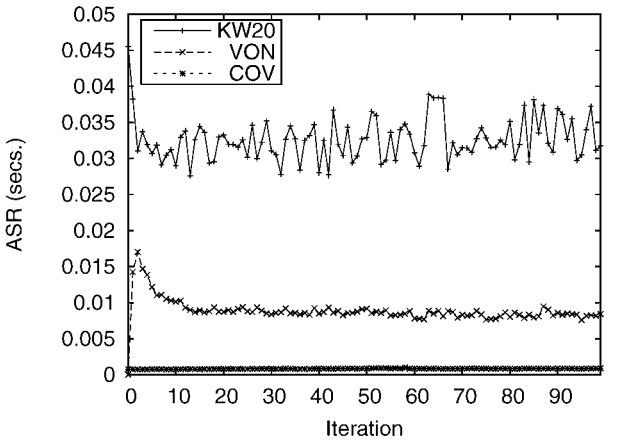


Figure 10. Average ASR comparisons for a population of 100 avatars following a UNF CCP pattern.

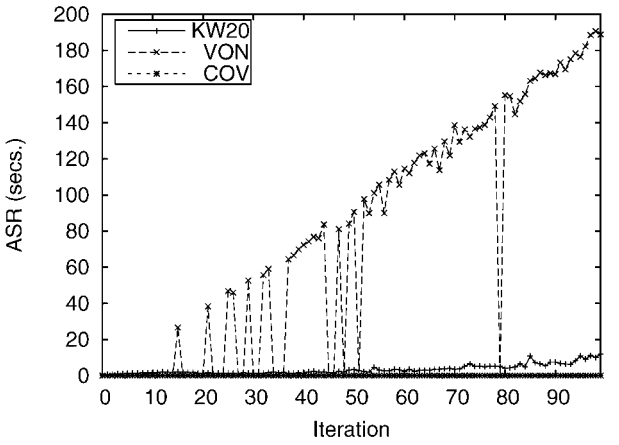


Figure 11. Average ASR comparisons for a population of 500 avatars following a UNF CCP pattern.

plot is not a completely flat slope, it only reaches values of seconds. Again, the only plot that shows a flat slope is the one corresponding to the COVER method.

Figure 12 shows the results for a population of 1000 avatars. In this case, the overhead added by the VON method collapsed the computer system, and the simulations could not finish properly. Therefore, this figure shows only two plots. The KW20 plot has an exponential behavior, reaching values of 20 and 30 seconds. In contrast, the COVER plot remains flat, close to the zero value. These results show that the COVER method also provides the best performance in terms of latency.

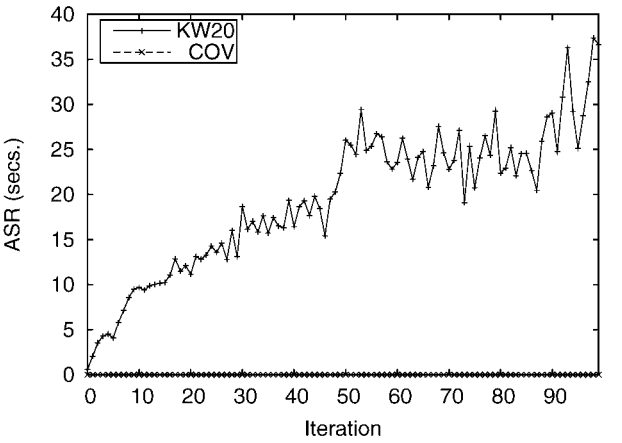


Figure 12. Average ASR comparisons for a population of 1000 avatars following a UNF CCP pattern.

Throughput

We have also studied the performance achieved with each method 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 by each method 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.

Figure 13 shows the results obtained for the KW20 method. This figure shows that this awareness method is not scalable with the population size, since the average ASR values remain in an order of magnitude of milliseconds only for a population of 100 avatars. For a population of 500 avatars, they increase up to an order of seconds, while for a population of 1000 avatars these values reach tens of seconds.

Figure 14 shows the results obtained for the VON method, showing that when the population increases to 500 avatars the awareness method adds an excessive overhead, saturating the system. The system saturation in turn results in a continuous increase of the average ASR values, as shown in Reference [11]. Moreover, this figure does not show the plot for a population of 1000 avatars because the system could not finish any complete simulation with this population size.

Figure 15 shows the results for the COVER method. It can be seen that all the plots have a flat slope, and they show values of milliseconds. These results show that

DOI: 10.1002/cav

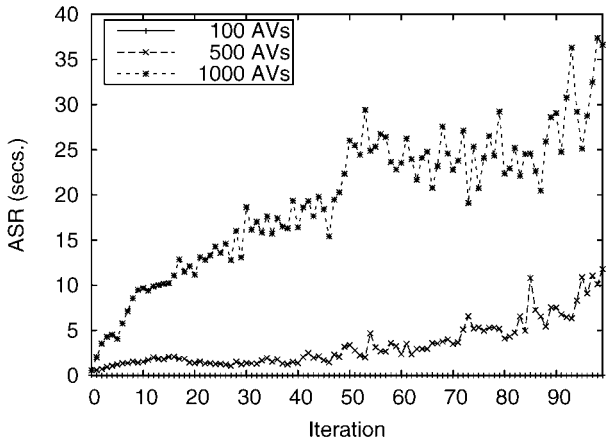


Figure 13. Average ASR values provided by the Kawahara method (AE = 20) for different population sizes.

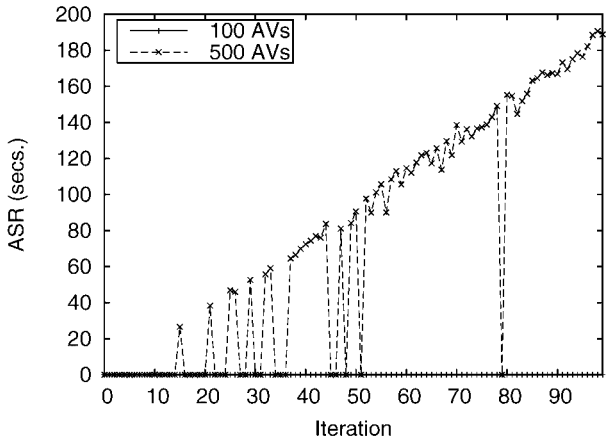


Figure 14. Average ASR values provided by the VON method for different population sizes.

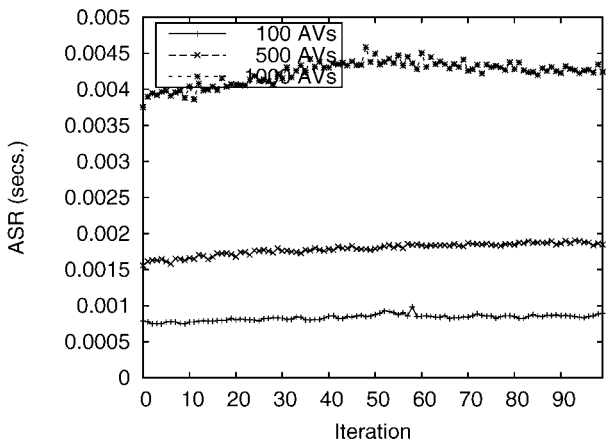


Figure 15. Average ASR values provided by the COVER method for different population sizes.

only the COVER method is scalable enough for supporting thousands of avatars. The key issue for this behavior is the scalable use of supernodes.

Awareness Delay

Apart from the well-known performance metrics like latency and throughput, in order to prove the effectiveness of an awareness method, it is also necessary to measure 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 i receives the acknowledgment from j as new neighbor. We have denoted this parameter as T_{AW}. This parameter is crucial, since it determines the maximum time–space inconsistencies that can arise in the system.

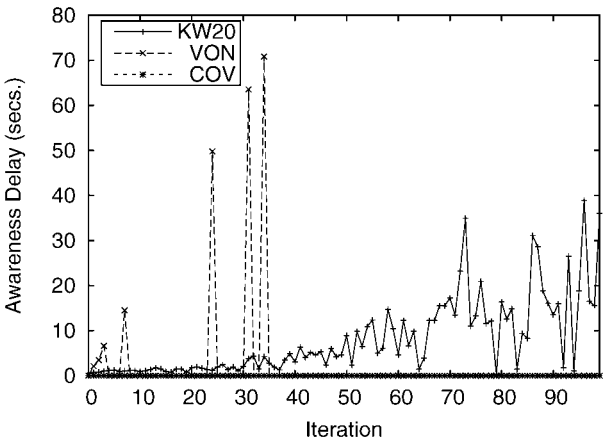


Figure 16. Awareness delays for a population of 500 avatars.

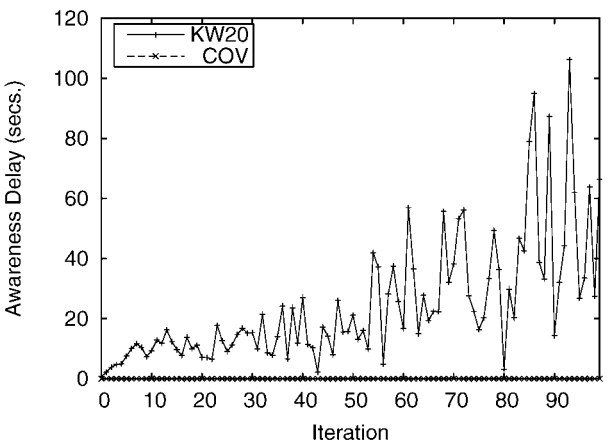


Figure 17. Awareness delays for a population of 1000 avatars.

Comp. Anim. Virtual Worlds 2008; **19**: 537–552
DOI: 10.1002/cav

Figures 16 and 17 show the results provided by the considered methods for different population sizes under a uniform-CCP combination of movement pattern and initial distribution of avatars. These figures show on the X-axis the iteration number, while they show on the Y-axis the average awareness delay (the T_{AW} value) for that iteration. These figures show that COVER is the awareness method that provides the lowest awareness delays. It is worth mentioning that this method manages to keep the awareness delay in an order of magnitude of 0.01 seconds, regardless of the considered population size. These results prove that the COVER method also provides an efficient performance in terms of Awareness Delay.

Conclusions and Future Work

In this paper, we have proposed a comparative study of the performance provided by those methods proposed in the literature that are supposed to fully solve the awareness problem in P2P DVE systems. This study has been performed using the well-known performance metrics in distributed systems, and the evaluation of the considered methods has been performed on real distributed systems.

The evaluation results show that only the COVER method provides a full awareness rate to avatars, regardless of the movement pattern that avatars follow in the virtual world. Both the VON and the Kawahara methods provide awareness rates around 95% for most of the simulation times. Additionally, the VON method provides even lower awareness rates (around 85%) for non-uniform movement patterns (HPA and HPN). These behaviors do not depend on the population size, since different sizes have been tested.

The results also show that the COVER method provides the best system performance in terms of well-known metrics like latency and throughput. This method is able to keep the average response times in an order of magnitude of several milliseconds, regardless of the population size and also regardless of the movement pattern of avatars. The key issue for achieving these results is the dynamic use of supernodes, that represent the second level of a hierarchical awareness scheme.

Finally, the COVER method also provides the best performance in terms of the awareness delay. This method provides similar awareness delays for different populations and for different movement patterns of avatars, thus keeping any time–space inconsistency that can arise among different clients within reasonable values.

These results validate the COVER method as an efficient and actually scalable awareness method for DVE systems based on P2P architectures.

As a future work to be done, we plan to characterize the behavior of P2P DVE systems implementing the COVER awareness method, in order to develop some technique that can improve the performance of these systems. Also, we plan to focus on systems where avatars can have different AOI sizes (like networked games where avatars can have instruments providing long-range views of the environment).

ACKNOWLEDGEMENTS

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

References

1. Singhal S, Zyda M. *Networked Virtual Environments*. ACM Press: New York, 1999.
2. Miller DC, Thorpe JA. SIMNET: the advent of simulator networking. In *Proceedings of the IEEE* 1955; **8**(83): 1114–1123.
3. Salles JM, Galli R, Almeida AC, *et al*. mworld: a multiuser 3D virtual environment. *IEEE Computer Graphics* 1997; **17**(2): 55–65.
4. Bouras C, Fotakis D, Philopoulos A. A distributed virtual learning centre in cyberspace. In *4th International Conference on Virtual Systems and Multimedia, VSMM98*, Gifu-Japan, 1998.
5. Everquest: http://everquest.station.sony.com/
6. Lineage: http://www.lineage2.com
7. Quake: http://www.idsoftware.com/games/quake
8. Anarchy Online: http://www.anarchy-online.com
9. Startcraft: http://www.blizzard.com/starcraft
10. Lui JCS, Chan MF. An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Transactions on Parallel and Distributed Systems* 2002; **13**: 193–211.
11. Morillo P, Orduña JM, Fernández M, Duato J. Improving the performance of distributed virtual environment systems. *IEEE Transactions on Parallel and Distributed Systems* 2005; **16**(7): 637–649.
12. Beatrice N, Antonio S, Rynson L, Frederick L. A multiserver architecture for distributed virtual walkthrough. In *Proceedings of ACM VRST'02*, 2002; 163–170.
13. Macedonia MR. A taxonomy for networked virtual environments. *IEEE Multimedia* 1997; **4**(1): 48–56.
14. Greenhalgh C, Bullock A, Frecon E, Llyod D, Steed A. Making networked virtual environments work. *Presence: Teleoperators and Virtual Environments* 2001; **10**(2): 142–159.

DOI: 10.1002/cav

15. Anderson DB, Barrus JW, Howard JH, Rich C, Shen C, Waters RC. Building multi-user interactive multimedia environments at merl. *IEEE Multimedia* 1995; **2**(4): 77–82.
16. Greenhlagh FC. Awareness-based communication management in massive systems. *Distributed Systems Engineering* 1998; **5**: 129–137.
17. Gerndt A, Hentschel B, Wolter M, Kuhlen T, Bischof C. Viracocha: an efficient parallelization framework for large-scale CFD post-processing in virtual environments. In *SC'04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, IEEE Computer Society, Washington, DC, USA, 2004; 50.
18. Alexander T. *Massively Multiplayer Game Development II*. Charles River Media Inc.: Hingham, MA, USA, 2005.
19. Rueda S, Morillo P, Orduña JM, Duato J. On the characterization of peer-to-peer distributed virtual environments. In *Proceedings of the IEEE Virtual Reality 2007 (IEEE-VR07)*, IEEE Computer Society Press, Charlotte, NC, USA; 107–114.
20. Mooney S, Games B. *Battlezone: Official Strategy Guide*. BradyGame Publisher: Indianapolis, Indiana, 1998.
21. Milojicic D, Kalogeraki V, Lukose R, *et al*. Peer-to-peer computing. *Technical Report, Technical Report HPL-2002-57*, HP Laboratories, Palo Alto, 2002.
22. Gautier L, Diot C. Design and evaluation of MiMaze, a multiplayer game on the internet. In *Proceedings of IEEE Multimedia Systems Conference*, 1998; 233.
23. Smith RB, Hixon R, Horan B. *Collaborative Virtual Environments*. Springer-Verlag; Heidelberg, Germany, 2001.
24. Zhou S, Cai W, Lee B, Turner SJ. Time-space consistency in large-scale distributed virtual environments. *ACM Transactions on Modeling and Computer Simulation* 2004; **14**(1): 31–47.
25. Fujimoto RM, Weatherly RM. Time management in the DoD high level architecture. In *Proceedings Tenth Workshop on Parallel and Distributed Simulation*, 1996; 60–67.
26. Roberts D, Wolff R. Controlling consistency within collaborative virtual environments. In *Proceedings of IEEE Symposium on Distributed Simulation and Real-Time Applications (DSRT'04)*, 2004; 46–52.
27. Smed J, Kaukoranta T, Hakonen H. A review on networking and multiplayer computer games. *Technical Report 454*, Turku Centre for Computer Science, 2002.
28. Mauve M, Vogel J, Hilt V, Effelsberg W. Local-lag and timewarp: providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia* 2004; **6**(1): 47–57.
29. Li L, Li F, Lau R. A trajectory-preserving synchronization method for collaborative visualization. *IEEE Transactions on Visualization and Computer Graphics* 2006; **12**(5): 989–996.
30. Morillo P, Orduña JM, Duato J. 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; 74–81.
31. Cronin E, Filstrup B, Kurc AR, Jamin S. An efficient synchronization mechanism for mirrored game architectures. *NetGames '02: Proceedings of the Ist Workshop on Network and System Support for Games, Bruanschweig, Germany.* ACM, New York, NY, USA, 2002: 67–73.
32. Kawahara Y, Aoyama T, Morikawa H. A peer-to-peer message exchange scheme for large scale networked virtual environments. *Telecommunication Systems* 2004; **25**(3): 353–370.
33. Knutsson B, Lu H, Xu W, Hopkins B. Peer-to-peer support for massively multiplayer games. In IEEE Infocom, March 2004.
34. 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.
35. Macedonia MR, Zyda M, Pratt DR, Brutzman DP, Barham PT. Exploiting reality with multicast groups: a network architecture for large-scale virtual environments. In *Proceedings of the 1995 IEEE Virtual Reality Annual Symposium*, 1995; 2–10.
36. Keller J, Simon G. Solipsis: a massively multi-participant virtual world. In *Proceedings of Parallel and Distributed Processing Techniques and Applications (PDPTA)*, C.S.R.E.A. Press (Computer Science Research, Education and Applications Press), Las Vegas, USA, 2003; 262–268.
37. Hu SY, Liao GM. Scalable peer-to-peer networked virtual environment. In *Proceedings of the ACM SIGCOMM 2004 Workshops on NetGames'04*, 2004; 129–133.
38. Hu S-Y, Chen J-F, Chen T-H. Von: a scalable peer-to-peer network for virtual environments. *IEEE Network* IEEE Communications Society, New York, NY, USA, 2006; **20**(4): 22–31.
39. Duato J, Yalamanchili S, Ni L. *Interconnection Networks: An Engineering Approach*. IEEE Computer Society Press Los Alamitas, CA, USA, 1997.
40. Guibas L, Stolfi J. Primitives for the manipulation of general sub-divisions and the computation of voronoi. *ACM Transactions on graphics* ACM, New York, NY, USA, 1985; **4**(2): 74–123.
41. Minar N. Distributed systems topologies. In *Proceedings of Peer-to-Peer and Web Services Conference (Seminar)*, O'Reilly Press, 2001.
42. Androutsellis-Heotokis S, Spinellis D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)* 2004; **36**(4): 335–371.
43. Oliveira M, Crowcroft J, Slater M. Components for distributed virtual environments. *PRESENCE: Teleoperators and Virtual Environments* 2001; **10**(1): 56–61.
44. Smith J, Chang S. Quad-tree segmentation for texture-based image query. In *Proceedings of the Second ACM International Conference on Multimedia*, 1994; 279–286.
45. Samet H. The quadtree and related hierarchical structures. *ACM Computing Surveys* 1984; **16**(2): 187–260.
46. Samet H. Hierarchical representations of collections of small rectangles. *ACM Computing Surveys (CSUR)* 1988; **20**(4): 271–309.
47. Samet H. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann Publisher Inc.: San Fransisco, CA, USA, 2006.
48. Wortham S. Terra3D: http://www.gldomain.com/Programs/Terra3D.htm
49. FIPA. Fipa agent management specification, 2000. Available at http://www.fipa.org/specs/fipa00023/
50. IEEE. *1278.1 IEEE Standard for Distributed Interactive Simulation-Application Protocols (ANSI)*, 1997.
51. Kuhl F, Weatherly R, Dahmann J. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice-Hall PTR: Indiana, USA, 1999.

DOI: 10.1002/cav

52. 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.
53. Matijasevic M, Valavanis KP, Gracanin D, Lovrek I. Application of a multi-user distributed virtual environment framework to mobile robot teleoperation over the internet. *Machine Intelligence & Robotic Control* 1999; **1**(1): 11–26.
54. Orduña JM, Morillo P, Fernández M. Workload characterization in multiplayer online games. *Lecture Notes on Computer Science* 2006; **3980**: 490–499.
55. Claypool M. The effect of latency on user performance in real-time strategy games. *Computer Networks* 2005; **49**(1): 52–70.
56. 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.

Authors' biographies:

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 algorthims 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. 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.

DOI: 10.1002/cav