Virtual Reality (2012) 16:187–203
DOI 10.1007/s10055-011-0198-6

ORIGINAL ARTICLE

Performance improvement of Distributed Virtual Environments by exploiting objects' attributes

Christos Bouras · Eri Giannaka · Thrasyvoulos Tsiatsos

Received: 23 November 2010 / Accepted: 10 August 2011 / Published online: 23 August 2011

Abstract Distributed virtual environments need to address issues related to the control of network traffic, resource management, and scalability. Given the distributed nature of these environments, the main problems they need to overcome are the efficient distribution of workload among the servers and the minimization of the communication cost. In this direction, a lot of work has been done and numerous relevant techniques and algorithms have been proposed. The majority of these approaches mainly focus on user entities and their interactions. However, most of actual DVE systems include additional and non-dynamic elements, denoted as objects, whose presence can affect users' behavior. This paper introduces virtual objects' attributes and proposes two approaches that exploit these attributes in order to handle workload assignment and communication cost in DVE systems. Both approaches take into account scenario-specific aspects of DVE systems, such as the impact that entities' attributes have on each other and the way this impact can affect the system's state. These scenario-specific aspects are then combined with quantitative factors of the system, such as workload, communication cost, and utilization. The experiments conducted in order to validate the behavior of the proposed approach show that the incorporation of object's presence can improve the DVE system's performance. More specifically, objects' presence and their attributes can assist in the significant reduction in the communication cost along with effective workload distribution among the system's servers.

Keywords Distributed virtual environments · Load balancing · VR techniques and systems

1 Introduction

A virtual environment can be considered as a simulation of either an imaginary or real-world generated by a computer. Virtual reality applications became notably popular the last two decades. This can be attributed to the wide expansion of high-speed Internet access that is required to support these systems, as well as the significant advances of both hardware and software. These advances led to the design and development of distributed systems, which allow geographically scattered concurrent users to communicate, collaborate, and interact in a highly realistic virtual environment. This type of shared, computer-resident virtual world, where users share the same 3D synthetic scene, is called a distributed virtual environment (DVE). A large number of applications were developed for supporting DVEs, which were gradually adopted in a wide range of both academic and industrial environments. Their most common application is still met in the entertainment area,

C. Bouras
Computer Engineering and Informatics Department, University of Patras, 26500 Rion, Patras, Greece

C. Bouras (✉)
Research Academic Computer Technology Institute, N.Kazantzaki Str. Patras University, 26500 Rion, Patras, Greece
e-mail: bouras@cti.gr

E. Giannaka
Athens Information Technology, 19.5 km Markopoulou Ave., 19002 Peania, Athens, Greece
e-mail: elgi@ait.gr

T. Tsiatsos
Department of Informatics, Aristotle University of Thessaloniki, P. O. Box 114, 54124 Thessaloniki, Greece
e-mail: tsiatsos@csd.auth.gr

where applications such as Massive Multimedia Online Games (MMOGs) have become extremely popular. Some representative examples of such applications are the World of Warcraft (http://www.worldofwarcraft.com/) game, Second Life (http://secondlife.com/), and EVE Online (http://www.eveonline.com).

In networked servers DVEs, the simulated world does not run on one computer system but on several that are connected through a network. Connected users view the virtual world on their computer (client), thus having their own local copy of the virtual environment. In the majority of existing DVE systems, users have the ability to navigate in the virtual world (i.e., changing their position coordinates), to interact with the objects of the virtual environment (i.e., changing some of their attributes such as location, shape, color), as well as to interact and communicate with other participating users. Therefore, interactivity is a basic requirement for the vast majority of virtual environments. In addition, achieving a high sense of realism and maintaining consistency among all users' views are of critical importance, so that all connected users are always aware both of the presence of other entities (either users or virtual objects) as well as of any actions performed. Persistence is realized by distributing and synchronizing both user input and user independent behavior. One common characteristic of DVEs is their dynamically changing state with users entering, navigating, interacting, and leaving the system randomly (at will), resulting in continuously changing utilization of resources for the DVE system. These changes, in turn, call for effective load distribution and management of the inter-server communication (for synchronization purposes), so that consistency is always maintained and scalability is supported. Regarding scalability, the work of Morillo et al. (2005) has shown that in a networked server DVE system when even one of the servers reaches 100% of CPU utilization, the performance of the overall DVE system is downgraded.

Existing approaches for handling load distribution, communication cost, and scalability in DVE systems mainly focus on the avatar entity that represents users and their behavior. However, DVE systems include additional non-dynamic elements, whose state should be managed and communicated to the participants. The term object is used for defining these non-autonomous elements. Recent research highlights the impact of objects' presence in DVE systems (Hu et al. 2006). The objects are active parts of the virtual environment and play an important role in supporting the scenario that the virtual environment simulates. There are different types of objects within the virtual world, with different attributes. More specifically, there are objects that users can interact with, objects that can be moved, and objects placed for supporting the visual and graphical representation of the virtual scene. For persistence, any change of the objects' state needs to be propagated to all users concerned. This propagation of states increases not only the workload but also the communication cost among the system's servers for their synchronization.

This paper presents a load distribution and rebalancing approach for DVE systems that exploits the presence of objects along with their attributes within the virtual environment. More specifically, the proposed approach uses the objects' location, their individual attributes, and the impact these attributes have on users' behavior to (a) create partitions of the virtual environment that are tolerant to changes, (b) distribute the workload among the participating servers in a way that none of them reaches saturation, and (c) minimize the communication among system's servers. It is therefore advancing the state-of-the-art that is mainly focusing on avatars' presence by introducing new parameters and scenario-specific aspects of the virtual environment. More specifically, the object-oriented approach presented takes into account scenario-specific aspects of DVE systems, such as the impact that entities' attributes in diverse simulated virtual scenarios have on each other and the way this impact can affect their social behavior and in turn the virtual environment's behavior and state. These aspects are then combined with quantitative factors of the system, such as workload, communication cost, and utilization for handling workload distribution, rebalancing, and minimization of the communication cost.

The proposed approach is evaluated through simulations in large-scale environments and is compared to an avatar-based technique. The results of the experiments clearly show that the major contribution of the approach is the significant reduction in the inter-server communication cost. Given the fact that DVEs depend strongly on the underlying network characteristics, the reduction in the messages exchanged among the system's servers is of increased value for the viability, scalability, and performance of the DVE system. Furthermore, the proposed approach achieved an effective load distribution scheme for longer time intervals, when compared to the avatar-based approach. Finally, the CPU utilization for all DVE's servers never reached the saturation point of 100% of CPU utilization that is proven to downgrade the overall DVE system's performance (Morillo et al. 2005).

This paper is structured as follows: Sect. 2 presents the research work done both in the direction of load balancing and prediction techniques in virtual environments. Section 3 presents the main attributes of objects in a virtual environment for highlighting the impact they can have on users' behavior along with virtual world characteristics. Sections 4 and 5 present the distribution and rebalancing approaches in terms of their main concepts and the parameters they incorporate. The section that follows presents the validation of the proposed approach and Sect. 7

presents the experiments conducted for assessing the algorithm's efficiency for a large-scale DVE. Finally, Sect. 8 provides conclusions of the paper.

2 Related work

The load distribution problem and the prediction of users' intentions and behavior have drawn increased research interest and a number of algorithms and methods have been proposed. This section presents part of the work that has been done both in the field of solving the load distribution problem as well as in the design of efficient prediction techniques for optimizing the DVE's performance.

In the direction of the partitioning problem, the approach presented in Morillo et al. (2003a) proposes a heuristic search based on Ant Colony Systems (ACS). This heuristic search method uses positive feedback to improve the use of good search paths, while using negative feedback to escape from local minima. Another approach (Macedonia et al. 1995) is to logically partition virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. This is accomplished by exploiting the actual characteristics of the real-world environments that are to be simulated and by focusing an entity's processing and network resources to its area of interest via an Area of Interest (*AoI*) Manager. A third approach rejects dynamic concepts associated with avatars like *AoI*, aura, or local (Tam 1998). This proposal divides the three dimensional (3D) virtual scene in a regular grid. A multicast group is created for each grid cell, in such a way that avatars sharing a cell also share multicast packets and are assigned to the same server. Although this approach provides a fast way of solving the partitioning problem, the performance of the static partitioning is quite low when avatars show a clustered distribution. Furthermore, the approach presented by Beatrice et al. (2002) considers that all objects within each region of the virtual environment are managed by a given server and adaptive load balancing is provided. In addition, the DVE system (Frecon et al. 2001) places a multicast partitioning scheme under the control of a scripting language, making dynamic partitioning dependent on object behaviors as well as simple spatial partitions. In the LOT approach, Lui and Chan (2002) have described the importance of finding a good assignment of the participating clients to the available servers with the aim to manage the workload and the communication cost and to achieve better network performance. This partitioning algorithm currently achieves very good results for DVEs (Morillo et al. 2003b). Finally, Morillo et al. (2005) presented a partitioning approach for improving the performance of DVEs that is targeted to keep all the servers in the system below a certain threshold value of CPU utilization, regardless of the amount of network traffic. Evaluation results show that this partitioning method can improve the DVE system performance, regardless of both the movement pattern of clients and the initial distribution of clients in the virtual world. Furthermore, areas such as parallel and distributed simulation have similar issues, such as load balancing of communication and computation resources and communication cost reduction. Some approaches perform load balancing at runtime by taking into account the spatial locality in the virtual world (Chen et al. 2005), while De Vleeschauwer et al. (2005) and Verdickt et al. (2007) propose algorithms to distribute the load by relocating the parts of the virtual world at runtime.

In the direction of prediction techniques and algorithms, there is a number of efforts that have been made concerning mainly the prediction of entities movement within the virtual environment. In particular, one type of path prediction methods is based on the statistical method using random process analysis and past navigation patterns (Liu and Maguire 1996) while dead reckoning is another popular technique used in DVE systems to reduce bandwidth consumption in transmitting the positional information of moving objects (DIS Steering Committee 1998). Furthermore, the work presented from McCoy et al. (2004) explores the human user behavior in multiplayer distributed media environments for determining a classification of entities' behavior to recognize and exploit statistically similar patterns of behavior in order to preempt state changes ahead of time and reduce latency problems. Finally, the work presented by Chertov and Fahmy (2006) describes an adaptive load balancing scheme for the server core that exploits the fact that clients tend to cluster around points of interest.

Research in the direction of load balancing for DVEs has been very active with numerous approaches presented and adopted. More specifically, existing work in the area of virtual worlds' partitioning shows that many approaches achieve good partitioning schemes. The majority of these approaches is based on the optimal handling of resources (either network or computational) and the assignment of the participating entities to the available resources. Regarding the prediction techniques, the majority of existing work is related to finding patterns of users' behavior or predicting the path of avatars' movement by using different methods to analyze the interactions as they occur in the virtual environment (Schroeder et al. 2006). However, in their vast majority, the existing approaches mainly focus on users' presence within the virtual environment, their actions, and the way they can be handled in a more quantitative manner.

The work presented in this paper is based on findings of existing approaches and advances the state-of-the-art by

taking into account interaction and communication aspects that arise from objects' presence in the virtual environment, their attributes, and the impact they have on the users' behavior. The work of Morillo et al. (2005) has shown that in a DVE when even one of the servers reaches 100% of CPU utilization, the performance of the overall networked server DVE system is importantly affected. Taking this limitation into account, the objective of the approach presented in this paper is twofold; it aims to create initial partitions more viable and tolerant to changes and also to achieve rebalancing in such a way that all connected servers stay below the prohibitive saturation point. More specifically, the load distribution is based on objects' and virtual world's attributes aiming at identifying the regions (partitions) with higher probability of being overloaded and assigning these regions to the available servers in an optimum way for extending the system's tolerance to changes. Regarding rebalancing, the approach adopts the concepts of utilization thresholds for performing changes as proposed in Morillo et al. (2005) and contrary to the LOT approach (Lui and Chan 2002), which performs rebalancing in regular time intervals.

3 Virtual world and entities' attributes

In their vast majority, virtual environments comprise two types of entities, avatars and objects. The avatars constitute the graphical representation of the participating users, while objects are non-autonomous entities of the virtual scene. Interaction among the participating entities is a central requirement for virtual environments. In both cases of interaction, all changes either to avatars' or to objects' states need to be propagated to all affected users to ensure persistence and awareness. In most cases, the avatars that participate in a virtual environment are allowed to perform certain types of actions. However, objects that constitute the virtual environment may significantly differ on the type of actions they support. Each of the objects of a virtual environment could have a variety of attributes, such as shape, color, position, size. The interaction of an avatar with an object could be considered as the ability to modify one or more of these attributes. Thus, based on the supported interaction level, 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, or other objects that are usually present to support the shape, form, and structure of the virtual world or even weather conditions, such as shadow or sun descriptions that correspond to objects such as the sky and sun.
- *Static active objects*: In the case of these objects, the avatars have the ability to interact with them and modify one or more of their attributes, apart from their position as these objects cannot be moved within the virtual scene. Examples of such objects could be machines, libraries, etc.
- *Non-static active objects*: This type of objects allows the modification of all of their attributes, including the position, by the avatars they interact with. Examples of such objects could be books, swords, cups, etc.

Varvello et al. (2008) studied the social behavior of human beings in a virtual context for Second Life along with its global and local-scale characteristics in terms of content. Their research showed that there are certain regions of the virtual world that users tend to visit, which are in most cases related to the objects each region contains. Furthermore, the majority of virtual environments, either used for educational, entertainment, or business purposes, simulate worlds with specific goals and achievement paths. For example, in an educational virtual environment, the user needs to follow certain paths in order to gain knowledge on a subject. In a virtual game, the player needs to accomplish a mission in order to win. In this process, objects play an important role for supporting the context of the scenario that the virtual world simulates. They are, thus, placed and displayed within the virtual world in a certain way, which is scenario-specific. Similarly, each of the objects, based on its role for the scenario, is assigned specific attributes. The approach presented in this paper is driven by the objects' attributes and the impact they can have on users' behavior.

For assessing objects' presence, we introduce three object attributes: (a) the Degree of Interaction (*DoI*), (b) the Level of Importance (*LoI*), and (c) the Space of Interaction (*SoI*). These attributes are presented in detail in the subsections that follow.

3.1 Degree of interaction (*DoI*)

In a virtual environment, users have the ability to interact with each other and with the objects of the world in order to complete the "mission" that each environment simulates. As mentioned above, the interactions in a virtual environment are strongly related to the number of actions supported by the application. In a virtual world comprising only avatars, the interactions are mainly driven by their social behavior, which usually follows the principles of real-world interaction. In a virtual environment comprising avatars and objects, the users' behavior can be driven by the purpose of these objects and their role in the virtual world. Objects that allow a number of actions to be performed on them have a higher probability of constituting points of interaction with

the connected users of the DVE system. The number of actions that can be performed on an object may vary and is related with the modification of one or more of its attributes (i.e., location, size, shape, color, texture).

We define as object's i degree of interaction (*DoI*) the number of actions that can be performed on this object. Thus,

$$DoI_i = \sum_{j=0}^{j=n} pr_j, \quad (1 \leq pr \leq 5) \tag{1}$$

where i is the object, j the number of actions supported for this object, and pr the priority, in terms of importance, of each action, in relation to the objectives of the DVE. The proposed approach takes into account this degree of interaction in order to predict avatars' distribution within the virtual environment. The priority pr factor is included in the calculation of the *DoI* attribute in order to include cases of highly context-sensitive environments, where some actions on the objects could be more important than others. For example, consider a virtual laboratory for medical experiments where the users can select a substance among a variety of medical substances in order to prepare virtual medicine. The users can move the substances in the virtual laboratory (thus changing their position), dilute them (thus changing their texture), and break them (thus, changing their shape). If all actions were of equal importance, then the *DoI* of a medical substance object, i.e., a vial would be three (as the number of actions). However, in a context-aware medical environment, the dilution could be the most important action that could be performed on the object, followed by the breaking of the object. For this case, the modification of the object's texture could have the maximum priority (i.e., $pr = 5$), the modification of its shape could be lower (i.e., $pr = 4$), and finally the modification of its location could be of the lowest priority (i.e., $pr = 1$). Therefore, in such an environment, the *DoI* for this object would equal to 10. Though in the majority of virtual environments, all actions are treated the same way and for these cases, the priority factor could be ignored.

The calculation of the *DoI* attribute in a virtual environment could be realized with a script that parses the virtual world file and calculates the supported actions for each object. The results of this parsing can be assigned to the 3D object as a new attribute (the *DoI* attribute). At this point, it should be mentioned that the calculation of the *DoI* attribute is strongly related to the programing language the virtual world has been programed with.

3.2 Level of importance (*LoI*)

Let us consider a virtual environment that simulates the museum of Louvre. Mona Lisa by Leonardo da Vinci is among the most famous artworks of the museum, which gathers a great number of visitors every year. In real life, visitors can admire the painting but are not allowed to touch it or perform any type of interaction with it. For simulating this reality in the virtual Louvre, the painting of Mona Lisa would be a static inactive object that the user could view but would not be allowed to interact with. This means that the *DoI* factor for this painting would equal to zero. However, as in real life, users' avatars would still gather around the piece, thus transforming it to a point of interest for the virtual environment. For capturing the importance of objects, we introduce the Level of Importance (*LoI*) attribute for each object of the virtual environment. The *LoI* factor of an object indicates whether an object is often visited by the participating users. The *LoI* values of objects could be used as indicators for users' movement and behavioral trends. Contrary to the *DoI* value of an object which is static, the *LoI* value changes over time according to users' preferences. Tracking of the objects' *LoI* attribute can provide useful information for the virtual world. More specifically, the objects' *LoI* values can highlight areas that need special resource handling either due to overcrowding or underpopulation.

The calculation of the *LoI* value is as follows: we initially assume that at time T_0 all objects of the virtual world have the same *LoI*. We set this initial value to 0. This value needs to be increased or decreased according to users' behavior. We set the scale for the *LoI* value within a range from 0 to 10, with 10 declaring an object that is of high importance to the virtual world. An object of the virtual environment would be the most important if all users gathered around it. However, for large-scale virtual environments, the number of concurrent users can be of the order of thousands and there is no realistic probability that such a high number of users could gather around a single object. Furthermore, the DVE system can be comprised by servers with different technical specifications, in terms of processing power, memory, etc. This heterogeneity of the system's servers defines the maximum number of concurrent users that each server can support. We define this maximum number as S_iavmax. This maximum number is also strongly related to the CPU utilization of each DVE server. More specifically, the presence of any number of users exceeding the S_iavmax threshold would lead to 100% of CPU utilization and the performance of the DVE system would be degraded (Morillo et al. 2007). We, therefore, consider that the *LoI* value of an object would be set to 10 if for a given period of time (t), there were S_iavmax users around it. For a real DVE system, this is the worst-case scenario as a single server would be dedicated in serving the demands around one single object.

Based on the above, we define a decision matrix for setting the values of the *LoI* parameter as follows:

The selection of the time period t that will be used for monitoring the changes of the *LoI* factor depends on the type of the virtual environment and the detail of information we need to acquire. A longer time period t can provide better results for identifying busy or idle regions of the virtual environment over time. A shorter time period can provide more accurate results for virtual environments that follow deep and major changes over short periods of time. In most cases, the time period t can be shorter at the initialization of the virtual world and when knowledge has been maintained on users' preferences, the time period could be prolonged.

At this point, it should be mentioned that in cases that the designers of the DVE systems have indications or actual data for the way that the users will initially behave in the virtual environment, they can themselves provide some initial values for the *LoI* parameter. Though these data and predictions could assist as a starting point for the values of objects' *LoI* because, as mentioned, the value of this attribute changes dynamically and shapes itself so as to better predict actual users' behavior as this behavior changes over time.

3.3 Space of interaction (*SoI*)

Users in the DVE are assigned the Area of Interest (*AoI*) attribute. The *AoI* represents the region of the virtual world within which the user's avatar needs to be aware of all entities and activities that take place, so as to assure awareness and persistence. Therefore, if an activity happens in the *AoI* region, the user state needs to be updated. Similarly, we introduce the term Space of Interaction for a given object (*SoI*). This attribute defines the area of the virtual scene, within which the objects are interactive among themselves or with the avatars of the DVE system. Figure 1 illustrates the concept of *SoI*. Even though the object is in the line of sight of both avatar A and avatar B, it is only avatar B that can interact with this object. The *SoI* 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 narrower. The *SoI* factor could constitute a useful parameter and filtering mechanism for the assignment and/or reassignment of entities to the servers of the DVE system. More specifically, if the difference of an object's *SoI* with an avatars location is small, then the avatar will be able to interact with this object.

If during rebalancing this object and the corresponding avatar are assigned to different servers, inter-server messages should be sent for synchronization and persistence purposes. Therefore, the object and the avatar should be kept in the same server in order to avoid increased communication load.

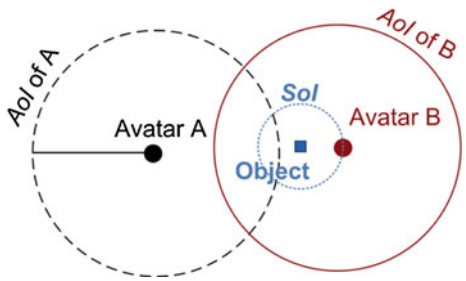


Fig. 1 Objects' space of interaction

3.4 Virtual world's cell division

For the spatial distribution of the virtual environment, the world is divided in equal-sized square cells. The number of cells is related to the average diameter D_A of avatars' Area of Interest (*AoI*). More specifically, the calculation of the cell size equals to D_A^2. The selection of D_A^2 as the cell size derives from the need to limit cases that the *AoI* of avatars intersects with more than one cells, as presented in Fig. 2a (cell edge is 1 m and D_A is 2 m). In particular, if the D_A of the avatars is larger than the cell's edge, the avatar has visibility of actions that take place outside its cell. It will, therefore, need to be notified for events that take place not only in the cell that it is located but also for events that take place in neighboring cells. Consequently, if the neighboring cells are located in different servers, then inter-server communication is required for maintaining avatar awareness.

Therefore, by selecting D_A^2 as the cell size, the probability for the intersection of the avatars *AoI* with neighboring cells is reduced (Fig. 2b) with the cell edge set to 4 m and D_A set to 2 m)) and so is the need for message exchange among neighboring cells and servers.

At this point, it should be mentioned that hexagon neighborhoods that are used in literature (Macedonia et al. 1995; Shirmohammadi et al. 2008) can also be applied for the division of the virtual world.

3.5 Virtual world's entry points and hotspots

In the majority of virtual environments, there are certain regions/areas where users are transferred when entering the virtual world or when transported from one place to another. In these areas, either they comprise objects or not, there is often a high concentration of avatars. Although users do not tend to stay for long in entry point areas, these cells can provide useful information on users' movement. More specifically, when avatars enter the virtual environment or are transported from one point to another (entry point), they will most likely move toward neighboring cells of these entry points. This information combined with the effect that *DoI*, *LoI*, and *SoI* play in avatars' behavior

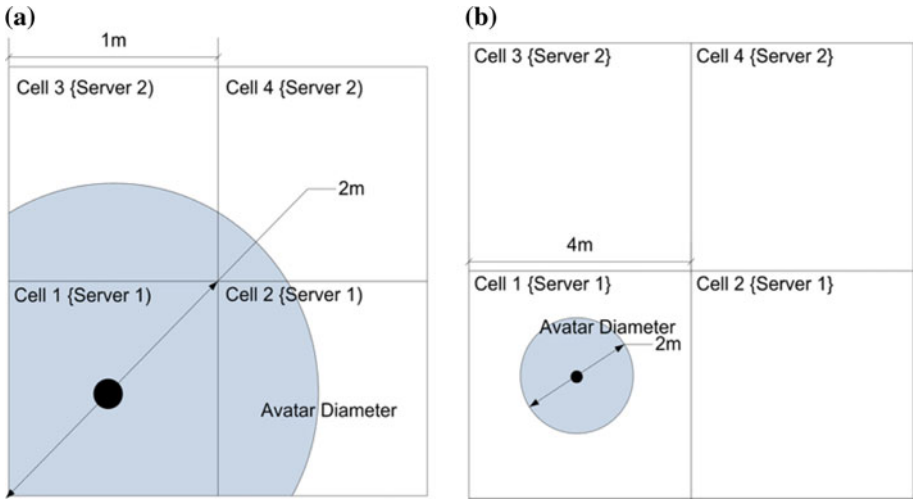


Fig. 2 Effect of cell size in relation to the *AoI*

(movement and interaction patterns) helps us identify and mark areas with high user concentration. We introduce and define these areas with the term hotspot. Hotspot areas, as defined in the proposed approach, add workload to the servers and should be therefore cautiously handled both during load distribution and rebalancing.

4 Distribution approach

This section presents the distribution approach that is based on the exploitation of the objects' attributes, described in the previous section, for achieving effective workload distribution (below the CPU threshold) and minimized communication cost among the servers of the DVE system. In particular, the basic concept of the algorithm described is based on the following observations:

- Firstly, in virtual environments, users tend to visit areas, where actions and interactions are allowed (http://playnoevil.com/serendipity/index.php?/archives/483-Linden-Labs-Second-Life-Server-Architecture-Questions.html).
- Furthermore, the majority of DVEs developed, which constitute simulations of real or imaginary worlds, are designed for supporting certain scenarios. Thus, the virtual world and the objects included are placed in ways that facilitate the users in performing these scenarios.

From the above, it becomes clear that the presence of objects within the virtual world is of high importance for the behavior and actions of the users that join the DVE system. Due to the fact that this behavior is driven by each user's individual preferences, interests, and characteristics, the prediction of users' behavior becomes challenging. However, based on the contextual purpose of objects within the virtual world and the attributes each of them has, they could be effectively exploited for designing behavioral patterns, which in turn can be used for optimizing the overall system performance. Based on these observations, the subsections that follow present the distribution approach and the actual algorithm.

4.1 Distribution process and object's attributes

The objective of load distribution in a DVE system is the efficient assignment of the virtual world and its entities to the available servers of the system in order to maintain a good overall performance. The efficient assignment is related both to managing the workload among the servers as well as to minimize the communication cost among them. The communication cost, used throughout this paper, is related to the messages that different servers need to exchange for maintaining the awareness of all connected users' views. Given the fact that the network and the availability of its resources can introduce delays that may affect the quality of the user experience, it is of vital importance that the communication cost of the total system is minimized.

Initially, all cells rely in one server. Each cell can comprise a number of objects. The workload for handling each cell depends both on the number of objects in this cell as well as on *DoI* and *LoI* values of these objects. More specifically, the number of objects increases the workload that each cell introduces to the server, while *DoI* and *LoI* "forecast" the workload and communication cost that each object will introduce to the server when users will interact or gather around it. Furthermore, *DoI*, *LoI*, and *SoI* could be considered as indicators for avatars' behavior (in terms of movement and interaction) within the virtual environment. The *DoI* and *LoI* values of an object could be used as prediction parameters. More specifically, when a number

of objects is within the *AoI* of an avatar, then the probability of the avatar to select the object that it will visit and interact with is strongly related to the *DoI* and *LoI* of these object. Finally, the *SoI* parameter of an object indicates "when" an interaction between this object and an avatar can be realized.

As mentioned above, apart from an effective workload distribution of the virtual world to the servers of the system, another important factor for the efficient partitioning is the minimization of the communication among participating servers. To address this issue, the distribution process takes into account the entry points and hotspot areas formulated around these entry points. More specifically, the algorithm, when distributing cells to the existing servers keeps the hotspot area, that is the entry point and the neighboring cells, on the same server. By keeping the hotspot areas together, the communication cost is reduced compared to the cost that the avatars' movement and interactions would produce, if these cells relied in different machines.

At this point, it should be mentioned that the rational of the algorithm is to mark the "predicted" crowded places within the virtual environment and assign them to the available servers so that workload will be balanced. Thus, the approach aims at assigning to each available server of the system at least one hotspot area.

4.2 DVE spatial distribution algorithm

As mentioned in the previous section, the distribution algorithm is a two-step process. In the first step, the hotspot areas of the virtual environment are located and formed. In the second step, these areas are assigned to the servers of the system.

Step 1 In the first step of the spatial distribution algorithm, the identification and formulation of the hotspot areas take place. The identification is based on the entry point cells of the virtual world and of their neighbors. More specifically, initially, the workload of all cells is calculated based on the number and attributes of the objects they contain. After the identification of the entry point cells, the neighbors of these cells are identified and are linked with the entry point cells and the hotspot area is formulated. As already mentioned, the aim is to assign at least one hotspot area to each of the available servers. For handling cases that the entry point cells are less than the available number of servers, the algorithm creates a "virtual" hotspot area. This creation is based on the selection of the cell with the maximum load (among the remaining cells). Similar to the entry point cells, a hotspot area is then formulated around the most loaded cell. The identification and formulation process of the virtual hotspot areas are depicted in Fig. 3.

Step 2 In the second step of the algorithm, the actual distribution of the environment to the servers takes place. The distribution follows a round robin approach starting from the entry point cells that are assigned one by one to the servers. The step that follows is the assignment of the entry point's neighbors to the server that the entry point is located following a descending order. When all hotspot areas' cells are assigned to the servers, the algorithm proceeds with the remaining cells. These cells are once again selected on a descending order (based on the cell load). If these cells are neighbors to one or more cells of an already assigned hotspot area, the algorithm selects the server with the minimum load among them and assigns this cell to this server. When this step is completed and for any remaining cells, the algorithm places them one by one to the servers, selecting each time the one with the minimum load.

At this point, it should be mentioned that in dynamic virtual environments the state of the virtual world is continuously changing as avatars join or leave the virtual space. It is, therefore, very difficult to find a partitioning scheme that could ensure a good performance throughout a long period of time, without the need of rebalancing the workload and communication cost among the servers and reassigning entities. However, the "long period of time" is scenario-specific for the DVE system. For example, in a virtual world that simulates a virtual battlefield, soldiers move and run, thus performing a large number of interactions. In this case, the changes in the state of the virtual world would be more frequent and severe than in the case of a virtual classroom, where users are seated, attending a lecture. Therefore, in the case of a virtual battlefield, a long period of time could be on the order of 10 min while in the case of the virtual classroom, it could be on the order of 30 min. The distribution approach presented in this paper proposes partitioning of the virtual environment, which aims at finding a good partitioning scheme that will ensure efficient performance of the system for longer time intervals after users' avatars enter the virtual environment as well as the minimization of the reassignments needed.

5 Rebalancing and the probability of interaction

The initial distribution of the virtual world can play an important role for the future reassignment needs of the system entities to the available servers. The dynamically changing state of the virtual environment, with users joining and leaving at will calls for future rebalancing needs. In the LOT approach (Lui and Chan 2002), the rebalancing is a two-step process: a workload balancing among the servers of the system and a communication refinement step for the further reduction in the communication cost. In the

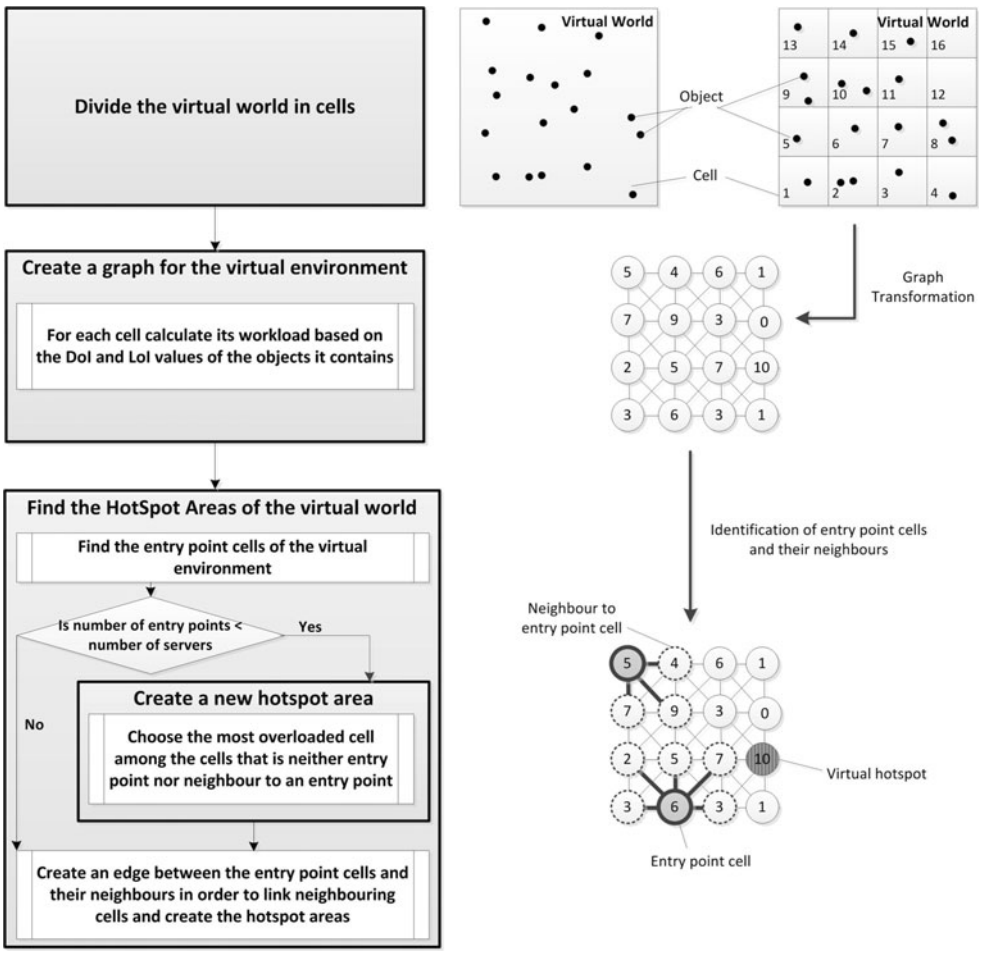


Fig. 3 Identification and formulation of *hotspot* areas

workload rebalancing or linear rebalancing, the algorithm creates a linear problem for which it aims at finding the optimal solution. However, not all systems provide a solution and furthermore how effective can a balanced workload distribution objective be?

In distributed systems, the servers available can significantly differ on their processing capabilities in terms of CPU and memory. Thus, a server of the system might be able to handle up to 100 concurrent users while another one maybe limited to 30. The heterogeneity of the servers that comprise the system is an important factor to be taken into account when considering the issues of load balancing and distribution, as there are cases that the workload among the servers is equally balanced, but for one server, this load could be easy to handle while for another one, it could be an overload. The equally balanced workload could only be effective in cases of homogenous systems, as all machines available are of the same type and with similar capabilities. For handling this issue, the proposed approach exploits performance thresholds, as proposed in (Morillo et al. 2005) different for each server based on its processing capabilities and resources.

Furthermore, the LOT approach (Lui and Chan 2002) performs rebalancing every fixed period of time (standard time intervals) in order to ensure balanced load distribution among the DVE system's servers. Though, there might be cases that the number of concurrent users in the system is low and the system servers can handle the generated load effectively. For these cases, time-based rebalancing would lead to reassignments of entities among the DVE's servers, when there is no actual need for that, thus affecting the state of all servers. In real DVE systems, the interventions and updates on the state of the DVE system's servers should be minimized in order to avoid possible impacts on the connected users' experience and smoothness of the overall DVE system's operation. These cases are avoided with the adoption of performance thresholds, as rebalancing takes place only when needed.

The rebalancing approach presented in this paper unifies in one step procedure the load balancing and communication refinement in order to achieve performance optimization and system consistency. The rebalancing exploits a new parameter denoted as Probability of Interaction (*PoI*) for each of the avatars. This parameter as well as the

rebalancing process is described in the paragraphs that follow.

5.1 Probability of interaction

An entity (object or avatar) can be viewed by a given avatar when it is located within the *AoI* of this avatar. Given the object attributes described in previous sections, *DoI* and *LoI* parameters can be used as indicators for identifying the objects that the avatar will tend to interact with from the set of feasible objects included in its *AoI*. Initially, the approach exploits the attributes of the objects and defines a probability of interaction for each one of them with the specific avatar. As we have already mentioned, the users of a DVE system tend to gather among objects based on their *DoI* and *LoI* values. Thus, the higher the value of these parameters, the higher the probability of an interaction between avatars and objects will be. 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 avatar with an object as the normalized value of those parameters (ranging from 1 to 10) with regard to the distance between the avatar and the object, as shown in (2).

$$PoI = \frac{w1 \times DoI + w2 \times LoI}{\text{dist}} \tag{2}$$

The objects that a virtual environment contains may significantly vary according to the scenario that each world simulates. In a virtual campus, the objects' attributes will have a different importance when compared to the objects of a virtual battlefield. Thus, in some cases, interactivity might be the most important factor while in others, the importance of an object might be the most meaningful parameter. To handle this diversity, in the *PoI* definition, attributes *DoI* and *LoI* are calculated based on weights *w1* and *w2*. In case of equal importance of *DoI* and *LoI* for objects of a virtual environment, both *w1* and *w2* are equal to 0.5.

Figure 4 presents an example for the different *PoI* values inside the *AoI* of an avatar. During the reassignment of avatars to the servers, the *PoI* parameter can be used as an estimate of the communication cost that will be introduced if a given avatar is moved from one server to another. In particular, for each "candidate for moving" avatar, the approach calculates the probability of interaction of this avatar with the entities of the server it is located (denoted as origin server) as well as the probability of interaction with the entities of the server it can be moved to (denoted as destination server). This probability of interaction is calculated as follows:

$$avatar_x\_s_i\_poi = \sum_{j=1}^{j=n} \left(PoI_{xj} = \frac{w1 \times DoI_j + w2 \times LoI_j}{dist_{xj}} \right) \tag{3}$$

The rebalancing process described in the subsection that follows uses the *PoI* parameter for identifying the entities that need to be assigned in order to improve the performance of the DVE system.

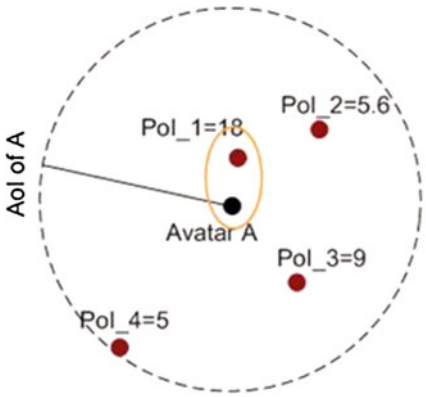
5.2 Rebalancing algorithm

The rebalancing process takes place when one of the system's servers exceeds a certain threshold. The threshold is defined by the system administrators and is related to the CPU utilization of a server for a certain period of time. As mentioned earlier, the results provided in (Morillo et al. 2007) show that if one of the system servers reaches 100% of CPU utilization, then the performance of the whole system is downgraded. Based on this, the application designers could set this threshold to a lower value on the order of 80–90%. The selection of a threshold value for the rebalancing process instead of defined time slots (as realized in Lui and Chan 2002) could avoid the application of algorithms when rebalancing is not really necessary and furthermore, preserve system performance if a server becomes overloaded before the predefined time that rebalancing takes place. The triggering of the rebalancing process can be realized with the SNMP protocol (http://tools.ietf.org/html/rfc1908), which constantly monitors the CPU utilization of all servers available.

When rebalancing is triggered, the algorithm examines the list of servers with which the overloaded server already communicates, in order to identify whether one of them could be used as candidate for the rebalancing. The identification of an already collaborating server is performed to avoid cases where the reassignment of an avatar to another server introduces communication cost among the two servers, which did not exist before the reassignment. When a neighboring server is identified, the algorithm creates a list with all border avatars of the overloaded server. As border avatars we define those avatars, whose *AoI* intersects with entities (objects and avatars) that are located in a different server. For each border avatar, the following values are calculated:

- The "forecasted" workload that the avatar will introduce to the "destination" server: this value provides an estimation of the workload that the avatar and its interactions will introduce to the "destination" server and is used in order to identify whether the destination server is able to handle it or not.
- The Probability of Interaction (*PoI*) of the border avatar with the entities of the "origin" server: this value is an

Fig. 4 Calculation of *PoI*



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

indicator of the communication cost that will be introduced if the border avatar is assigned to the destination server.

- The Probability of Interaction (*PoI*) of the border avatar with the entities of the "destination" server: this value is an indicator of the communication cost between the origin and the destination server that will be introduced if the border avatar remains to the origin server.

After calculating the above for all border avatars of the overloaded origin server, the algorithm creates a descending list based on the values of the probability of interaction to the destination server. The reason behind the selection of this value as the prime value is based on the fact that apart from workload de-loads, we need to move avatars that introduce or are probable to introduce the highest communication cost. When the list is prioritized, for each of the border avatars with the highest communication value, the algorithm makes a comparison with the existing probability of interaction value. If the destination value is higher than the origin, the capability of the destination server is examined in order to identify whether it could accept the workload or not. To define the capability of the server, another threshold is introduced, denoted as "operating," which indicates that a server is able to process additional workload. If the destination server performance is below the operating threshold, the avatar is moved to the destination server. As mentioned above, at the beginning of the rebalancing process, the algorithm checks among the list of servers with which it already communicates (exchanges messages) in order to avoid the establishment of additional unnecessary inter-server communication. However, for handling cases where the overloaded server does not exchange messages with other servers, the algorithm searches among the other servers and tries to find a candidate able to handle the additional workload.

Based on the above, the steps of the rebalancing algorithm are depicted in Fig. 5. The algorithm described above performs all the necessary steps for identifying and selecting the best possible solution for rebalancing, based both on the system's criteria (such as the CPU utilization) and on the criteria that the presence of avatars and objects introduces. At this point, it should be mentioned that the rebalancing process and in particular the selection of the avatars that will be re-assigned is based on the *PoI* parameter, which stands as an indicator for the system communication cost.

6 Validation

For validating the efficiency of the approaches, the load distribution along with the rebalancing algorithm has been applied to a virtual environment. The environment runs on three servers with the same capabilities in terms of computing power (homogenous system). For the validation of the approach, the following are presented:

- *The results of the algorithm for the spatial distribution of the virtual environment* These results present the spatial assignment of the virtual environment by exploiting the objects' attributes and demonstrate the distribution of cells and entities (objects) to the servers of the system after the application of the spatial distribution algorithm. In addition, the steps followed for the validation scenario are presented along with the distribution of workload among the servers when distribution is completed.
- *The results in terms of workload and communication cost of the system for the rebalancing approach* These results are provided to validate the rebalancing efficiency when one of the system servers reaches the utilization threshold. In this direction, for each of the scenario tested, the state of the virtual environment is presented when rebalancing is triggered (in terms of a snapshot of the virtual world). To demonstrate the rebalancing process, we provide the initial distribution of the virtual environment, the state of the virtual world when rebalancing is triggered, and finally the state and cost of the system when rebalancing is completed.

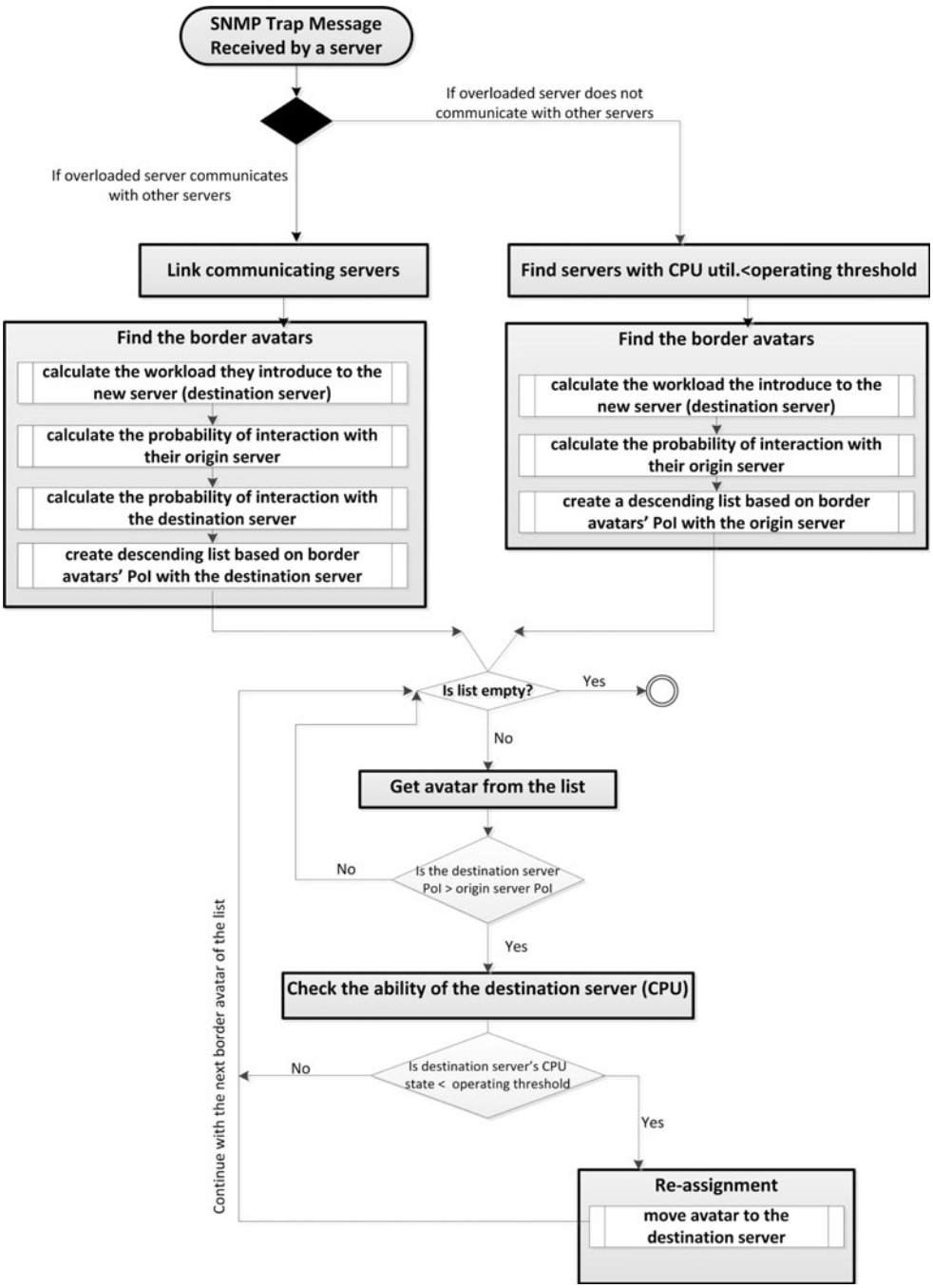


Fig. 5 Representation of the rebalancing algorithm

For the validation, we consider a small-scale virtual environment with dimensions 4×4. The virtual environment consists of 23 virtual objects and 28 avatars. The objects of the virtual environment are marked with polygons, while the values in these polygons represent the normalized value of *DoI* and *LoI* for each object. In these experiments, we consider a virtual environment that the user can interact with the virtual objects that are placed in the virtual scene for supporting specific objectives. This means that both *DoI* and *LoI* attributes of objects play an important role and are therefore of the same importance for the virtual environment examined. This means that their weight is equal and is set to 0.5 for each of them ($w1 = w2 = 0.5$). We also consider that the average workload that an object introduces to a server is on the order of 5 units while for the avatars, it is set to 10 units, the average avatar diameter D_A is 1, and the number of servers available equals to 3. Finally, for the CPU thresholds, we define that rebalancing will take place when a server exceeds 90% of utilization, while the CPU operating threshold is set to 65% (Table 1).

The application of the algorithm presents the case where all entry points share common neighbors. For this case, the

Table 1 Decision matrix for *LoI* calculation

Range of number of avatars visited object within period t	loi_x
$\frac{(loi_{(x-1)} \times S_i av\max}{loi_{\max}} + 1 \leq av\_visits \leq \frac{loi_x \times S_i av\max}{loi_{\max}}$	$x = 0$
.....	....
$\frac{(loi_{(x-1)} \times S_i av\max}{loi_{\max}} + 1 \leq av\_visits \leq \frac{loi_x \times S_i av\max}{loi_{\max}}$	$x = 10(x_{\max})$

process that takes place is the following: We note that the number of entry points available is 2 (C5 and C15), while the number of existing servers is 3. Thus, we need to create an additional "virtual" entry point. From the remaining cells, we select the one with the highest workload, which is C13. Since we have created at least one entry point to be assigned to each server, we start the formulation of the hotspot areas and the assignment of the hot spots cells to the available servers (using round robin). The step that follows is the calculation of the total number of neighboring cells for each hot spot as well as to give priority (when assigning the neighbors) to entry point with the smaller number of neighbors (so as to ensure that workload will be distributed and all entry points are assigned with neighbors). The results of this application of the algorithm are presented in Fig. 6a, while the steps followed are shown in Table 2.

After the spatial distribution of the virtual environment by exploiting objects' attributes, the experiment examines the effect of the rebalancing process. The triggering of the rebalancing step takes place when one of the servers exceeds the defined CPU threshold. In the case of the presented scenario, Server 3 is the one that triggers the event. Figure 6b presents the state of the virtual environment when the threshold is reached and rebalancing is called. At this moment, there are four border avatars with objects within their *AoI* located in different servers.

More specifically, as presented in Fig. 6b, *border_avatar*$_1$ has 2 objects in its *AoI* located in Server 3 and 1 object within its *AoI* located in Server 2, *border_avatar*$_2$ has 1 object in its *AoI* located in Server 3 and 1 object within its *AoI* located in Server 2, *border_avatar*$_3$ has 1 object in its *AoI* located in Server 3 and 1 object within its *AoI* located in Server 1, and finally, *border_avatar*$_4$ has 1 object in its *AoI* located in Server 3 and 1 object within its *AoI* located in Server 1.

As described in the rebalancing algorithm, all border avatars of the overloaded server, denoted as origin server, are selected. For each of them, the *PoI* values are calculated both for the origin (server 3) and the candidate (Server 1 and Server 2) servers. These *PoI* values are then sorted from high to low, as shown in Table 3.

Based on the comparison of the *PoI* between the destination and origin server in combination to the capability of the destination server to accept additional workload ($s_i\_CPU_{op}\_thr$), border avatars are reassigned among the system servers. It should be mentioned that in the column "Workload" of Table 3, the workload that each avatar introduces to the server is calculated. In the experiments conducted, for all scenarios tested, we have taken an average avatar workload of the order of 10 units. Thus, this column could be neglected from the sorting table. However, we keep this column in all experiments descriptions in order to ensure that the calculations are clear to the reader. The cost of the virtual environment in terms of workload for each of the existing servers and the communication cost among these in the states before and after rebalancing are presented in Table 4. As it can be noticed from the above table, the rebalancing process achieved the

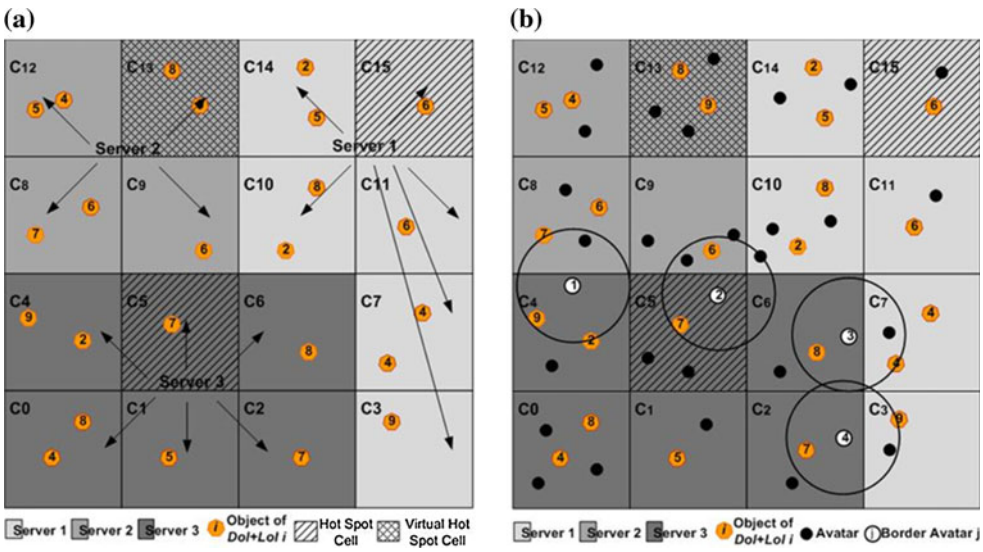


Fig. 6 Spatial assignment (**a**) and triggering state (**b**)

Table 2 Workload assignment for the spatial distribution

Steps	Server 1	Server 2	Server 3
1	C_{15}	C_{13}	C_5
2	C_{10}	C_8	C_0
3	C_{14}	C_{12}	C_4
4	C_{11}	C_9	C_6
5			C_2
6			C_1
7	C_7		
8	C_3		
Workload	46.0	45.0	50.0

Table 4 Rebalancing effect on system cost

	Before rebalancing state	After rebalancing state
Workload		
Server 1	136	156
Server 2	145	155
Server 3	180	150
Communication cost		
Server 1–server 2	21	12
Server 2–server 3	18	14
Server 1–server 3	25	14

distribution of workload in such a way that all connected servers are below the maximum threshold defined. The most important aspect, however, is the important reduction in the communication cost among the participating servers, which is of vital importance for the DVE system.

7 Experiments

After describing and validating the proposed object-oriented spatial distribution approach, we present the experiments conducted for a large-scale DVE. The results demonstrate that the main improvement was the reduced communication cost. Furthermore, the load distribution and rebalancing approach achieved balanced workload among the system's servers in terms of CPU usage, without ever reaching the prohibitive saturation point of 100% of CPU utilization for any individual server over the duration of the experiments. The experiments were conducted with the STEADiVE (Simulation Tool for Evaluating and Assessing Distributed Virtual Environments) tool (Bouras et al. 2009), which is a simulation tool implemented with Simul8 (Simul8 Simulation Software). The STEADiVE tool can be used by designers of DVE systems for simulating the performance of their approaches under different scenarios.

In order to examine the performance of the object-oriented spatial distribution approach, we compared it with that of the LOT approach (Lui and Chan 2002) that is proven to provide good results for large-scale DVEs. In short, the LOT approach follows a three-step process by first distributing the virtual environment into the available servers of the system using a Divide and Conquer technique. In the step that follows for a defined time interval, an algorithm checks the workload on the servers and performs all the necessary re-assignments of entities so that a nearly equally balanced workload is achieved. The third and last step of the approach encounters the exchange of some entities among the servers for the refinement of the communication cost.

7.1 Experiments' setup

The experiments conducted consider a DVE system comprising of 8 servers. The DVE system does not correspond to an actual DVE system (i.e., Second Life, WoW, or similar application). It is a DVE generated through simulation using the STEADiVE tool mentioned above. All of the servers available have the same computing power and are dedicated in serving the DVE system requests (no background applications are running). For comparison reasons, the setup of the DVE system adopts the values and characteristics of a large-scale DVE as defined and evaluated by the LOT approach (Lui and Chan 2002). In this direction, the experiments consider a large virtual world with a dimension of 25 × 25 units with the total number of avatars equals to 1,500. The radius of the *AoI* of each avatar is equal to 0.5 while the number of objects within the virtual environment equals to 900 and these objects are uniformly scattered within the virtual environment. Furthermore, the virtual environment comprises 18 entry

Table 3 Calculation and sorting of *PoI* values

	Workload	S_{1_poi}	S_{2_poi}	S_{3_poi} (origin)	Destinatioon server compatibiblity	Decision
$border\_avatar_4$	10	39,28		17,50	√	Move →
$border\_avatar_1$	10		34,83	22,75	√	Move →
$border\_avatar_2$	10	20	34,51	14,58	√	Move →
$border\_avatar_3$	10	33		20	x	Remain

points. For the OO Spatial approach, the *CPU_max* threshold is set to 80% while the *CPU_operating* threshold is set to 40%. For the LOT approach, rebalancing takes places every 10 min. The experiments run for a period of 180 min (3 h) and approximately 1,000 runs were executed. The results presented correspond to the average values obtained from these runs.

The main parameters monitored throughout the experiments were the workload of the available servers and the communication cost among them. Regarding the calculation of the communication cost, for simulation purposes, we consider an average message size of x units throughout the experiments conducted. Thus, the communication cost in all experiments and figures presented reflects the number of messages exchanged among the interconnected servers. Furthermore, another factor to be taken into account in DVEs is the latency of the system to users' requests. As in DVEs latency cannot be measured properly, the round-trip delay (RTT) or ASR (average response time) is used instead (Morillo et al. 2007). For the experiments conducted, the results obtained for the average ASR are compared to the 250 ms ASR threshold used in the literature.

7.2 Experimental results

Figure 7 presents the communication cost for the LOT and the object-oriented spatial distribution approach. The figure clearly demonstrates that the communication cost for the OO spatial approach is significantly reduced when compared to the communication cost introduced by the LOT approach. Given that DVEs are strongly dependent on the underlying network characteristics (i.e., delay, delay jitter, throughput), the reduction in the messages exchanged among the system's servers is of great value for the viability, scalability, and performance of the system. The higher communication cost of the LOT approach is due to the fact that avatar–object interactions have not been considered during the distribution. More specifically, in the avatar-based distribution of the LOT technique, the algorithm aims at minimizing the inter-server messages among avatars that rely on different partitions. Even though the avatar-to-avatar communication may be reduced, the approach fails to address the communication cost generated by avatar–object interaction.

Given the high concentration of objects in the experiments conducted compared to the avatars' number, the LOT approach presents significantly higher communication cost due to the omission of object entities.

Another major difference between the LOT and the OO spatial distribution approaches is the triggering of the rebalancing process. In real DVE systems, the interventions and updates on the state of the connected servers should be minimized in order to avoid possible impacts on the connected users' experience and smoothness of the overall system's operation. In the LOT approach, rebalancing takes place after a defined period of time (every 10 min in the experiments conducted). For the OO spatial distribution approach, rebalancing takes place only when one or more of the servers reach the CPU performance threshold. Figure 8 presents the trigger of the rebalancing for the two approaches over time. From the results, it can be seen that the LOT approach performs 18 rebalancing processes on the DVE system when the LOT approach performs only 7.

The results for the OO spatial approach show that the initial load distribution of the virtual environment based on object's attributes achieves a distribution scheme that is viable for longer time intervals. Furthermore, it can be seen that as time passes by the time intervals between the rebalancing for the OO spatial approach are prolonged. As mentioned in the previous section, objects' *LoI* factor changes dynamically based on users' behavior and preferences. Tracking of objects' *LoI* can provide useful information for the virtual world as it can highlight areas that need special resource handling. Thus, the prolongation of the time intervals between rebalancing could be explained by the fact that the DVE system learns over time users' behavior and uses this knowledge for future load distribution and rebalancing.

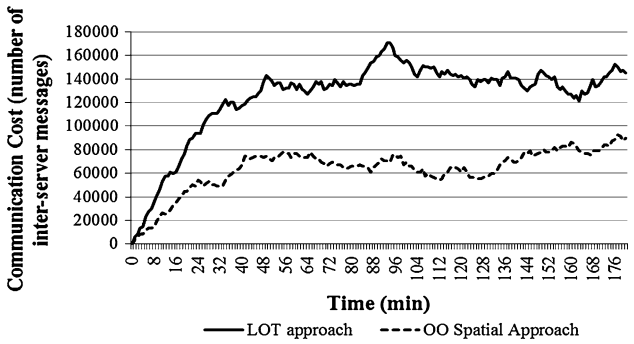


Fig. 7 Communication cost for the LOT and the OO spatial approach

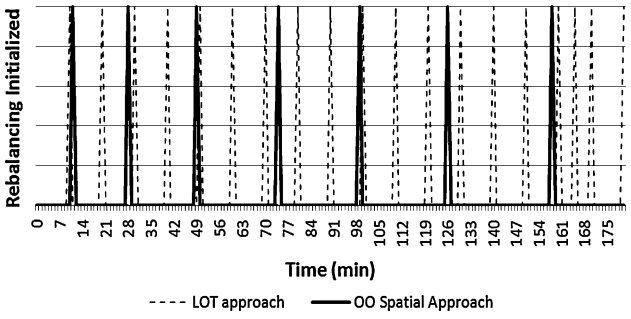


Fig. 8 Trigger of rebalancing for the LOT and the OO spatial approach

Table 5 Average ASRs for the LOT and the OO spatial approach

Average ASR (in ms.)	
OO spatial approach	LOT approach
219	245

As mentioned earlier, a factor to be taken into account in DVEs is the latency of the system to users' requests. As in DVEs latency cannot be measured properly, the round-trip delay (RTT) or ASR (average response time) is used instead (Morillo et al. 2007). The average ASRs in milliseconds obtained for the experiments conducted are presented in Table 5.

Based on the work presented by Henderson and Bhatti (2003), if the ASR is not greater than 250 ms, then users perceive that the system responds quickly. The results obtained from our experiments (Table 5) illustrate that both approaches achieve an average ASR below the 250 ms threshold. Furthermore, the results show that the OO spatial distribution approach achieves better results compared to the LOT one.

8 Conclusion

This paper identifies the importance of objects within DVE systems and describes a way that objects' attributes can be exploited for handling performance issues that DVE systems need to address. More specifically, the objects' attributes are introduced along with the impact they can have on users' behavior within the virtual environment. These attributes are then used for assisting both the distribution of the virtual world and the rebalancing process when rebalancing is needed. Throughout this paper, objects' attributes could be proven of high importance for predicting users' behavior within the virtual world and for "forecasting" the dynamically changing needs of the DVE system.

The object-oriented spatial distribution approach presented is based on the fact that an effective initial partitioning of the virtual environment could make the system more tolerant to future changes as it can be viable for longer time intervals, so that rebalancing needs are reduced. For the distribution, the approach exploits the objects' presence and virtual world characteristics in order to identify and forecast the most demanding regions of the environment, in terms of resources needed, and to assign them to the servers available. However, independent of the efficiency of the initial partitioning rebalancing is always needed due to the dynamic nature of DVEs. In this direction, the paper presents a rebalancing approach that combines scenario-specific aspects of entities' presence to identify their optimal assignment to the servers of the system.

In order to evaluate the efficiency of the proposed approach, experiments were conducted in large-scale environments and the approach was compared to an avatar-based technique. The results of the experiments clearly showed that the major contribution of the approach is significant reduction in communication cost. Given the fact that DVEs depend strongly on the underlying network characteristics, the reduction in the messages exchanged among the system's servers is of increased value for the viability, scalability, performance, and overall communication efficiency of the DVE system. Furthermore, the proposed approach achieved an effective load distribution scheme for longer time intervals without reaching the saturation point of 100% of CPU utilization.

References

EVE Online. http://www.eveonline.com. Accessed 29 May 2011

Second Life. http://secondlife.com/. Accessed 29 May 2011

Simul8 Simulation Software. http://www.simul8.com. Accessed 29 May 2011

Beatrice N, Antonio S, Rynson W, Frederick L (2002) A multi-server architecture for distributed virtual walkthrough. In: Proceedings of the ACM symposium on virtual reality software and technology, Hong Kong, China

Bouras C, Giannaka E, Tsiatsos T (2009) A framework model for DVEs using SIMUL8. In: Proceedings of the second international conference on simulation tools and techniques (SIMUTOOLS), Rome, Italy

Chen J, Wu B, Delap M, Knuttson B, Lu H, Amza C (2005) Locality aware dynamic load management for massively multiplayer games. In: Proceedings of principles and practice of parallel programming (PPoPP). pp 289–300

Chertov R, Fahmy S (2006) Optimistic load balancing in a distributed virtual environment. In: Proceedings of the 16th ACM international workshop on network and operating systems support for digital audio and video (NOSSDAV), pp 74–79

De Vleeschauwer B, Van Den Bossche B, Verdickt T, De Turck F, Dhoedt B, Demeester P (2005) Dynamic microcell assignment for massively multiplayer online gaming. In: Proceedings of Netgames 2005, New York, USA

DIS Steering Committee (1998) IEEE Standard for distributed interactive simulation – Application protocols doi: 10.1109/IEEESTD.1998.88572

Frecon E, Smith G, Steed A, Stenius M, Stahl O (2001) An overview of the COVEN platform. Presence Teleoper Virtual Environ 10(1):109–127

Henderson T, Bhatti S (2003) Networked Games: A QoS-sensitive application for QoS-insensitive users. In: Proceedings of the ACM Int'l conference applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03). pp 141–147

Hu SY, Chen JF, Chen TH (2006) Von: a scalable peer-to-peer network for virtual environments. IEEE Netw 20(4):22–31

Liu G, Maguire G (1996) A class of mobile motion prediction algorithms for wireless mobile computing and communication. Mob Netw Appl arch 1(2):113–121

Lui JC, Chan MF (2002) An efficient partitioning algorithm for distributed virtual environment systems. IEEE Trans Parallel Distrib Syst 13(3):193–211

Macedonia M, Zyda M, Pratt D, Brutzman D, Bar-ham P (1995) Exploiting reality with multicast groups: A network architecture for large-scale virtual environments. In: Proceedings of virtual reality annual international symposium (VRAIS)

McCoy A, Delaney D, McLoone S, Ward T (2004) Towards statistical client prediction—analysis of user behavior in distributed interactive media. In: Proceedings of the international conference on computer games: artificial intelligence design and education (CGAIDE), Reading, U.K

Morillo P, Fernandez M, Orduna JM (2003a) An evolutive approach to the partitioning problem in distributed virtual environment systems. In: Proceedings of XIV Jornadas de Paralelismo. Madrid, Spain, pp 299–304

Morillo P, Orduna JM, Duato J (2003b) On the characterization of distributed virtual environment systems. In: Proceedings of European conference on parallel processing (Euro-Par). Klagenfurt, Austria

Morillo P, Orduna JM, Fernandez M, Duato J (2005) Improving the performance of distributed virtual environment systems. IEEE Trans Parallel Distrib Syst 16(7):637–649

Morillo P, Rueda S, Orduna JM, Duato J (2007) A Latency-Aware partitioning method for distributed virtual environment systems. IEEE Trans Parallel Distrib Syst 18(9):1215–1226

SNMP v2. http://tools.ietf.org/html/rfc1908. Accessed 29 May 2011

PlayNoEvil Game Security News and Analysis. http://playnoevil.com/serendipity/index.php?/archives/483-Linden-Labs-Second-Life-Server-Architecture-Questions.html. Accessed 29 May 2011

Schroeder R, Heldal I, Tromp J (2006) The usability of virtual environments and methods for the analysis of interaction, Journal. Presence Teleoper Virtual Environ (MIT Press) 15(6):655–667

Shirmohammadi S, Kazem I, Ahmed DT, El-Badaoui M, De Oliveira J (2008) A visibility-driven approach for zone management in simulations. Simulation 84(5):215–229

Tam PT (1998) Communication cost optimization and analysis in distributed virtual environment. Technical report, department of computer science and engineering, The Chinese University of Hong Kong

Varvello M, Picconi F, Diot C, Biersack E (2008) Is there life in second life?. In: Proceedings of the ACM CoNEXT conference, Madrid, Spain, pp. 1-12

Verdickt T, De Vleeschauwer B, Van Den Bossche B, De Turck F, Dhoedt B, Demeester P (2007) Adaptive microcell assignment in massively multiplayer online games. In: Proceedings of the 10th international conference on computer games; AI, animation, mobile, educational and serious games (CGAMES), Louisville, Kentucky, USA, pp 92–99

World of Warcraft. http://www.worldofwarcraft.com/. Accessed 29 May 2011