Simulation Modelling Practice and Theory 25 (2012) 1–16

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

A simulation modelling tool for Distributed Virtual Environments

Christos Bouras [a,b,*], Eri Giannaka [a,c], Thrasyvoulos Tsiatsos [d]

[a] Computer Engineering and Informatics Department, University of Patras, GR-26500 Rion, Patras, Greece
[b] Research Academic Computer Technology Institute, N. Kazantzaki Str., Patras University, GR-26500 Rion, Patras, Greece
[c] Athens Information Technology, 0.8 km Markopoulou Ave., 19002 Peania, Attika, Greece
[d] Department of Informatics, Aristotle University of Thessaloniki, PO Box 114, GR-54124 Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 2 November 2010
Received in revised form 16 December 2011
Accepted 16 February 2012

Keywords:
Distributed Virtual Environments
Performance monitoring
Simulation modelling
Simulator

ABSTRACT

Distributed Virtual Environments simulate the behaviour and activities of a great number of users interacting in a virtual world over a wide area network. The size of the virtual worlds and the tremendous number of users that these environments are called to support require additional bandwidth and computational resources. For supporting large-scale Distributed Virtual Environments, extended infrastructure is needed in terms of both hardware and software. However, both researchers and application designers do not always have access to such an extended infrastructure and the assessment and evaluation of developed performance improvement techniques becomes extremely difficult. To address this issue, this paper presents a simulation modelling tool, called STEADiVE for networked servers Distributed Virtual Environments that could be used by designers for evaluating the performance of their approaches under different scenarios and system settings. The validation of the simulation modelling tool has showed that it achieves high accuracy in representing a real DVE system. STEADiVE comes to fill in the gap in the area of simulation tools for these systems.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Distributed Virtual Environments (DVEs) have become a major trend in distributed applications [6]. DVEs simulate real or imaginary worlds by incorporating rich media and graphics. A large number of platforms and applications were designed and developed to support large-scale 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, where platforms such as Second Life [10] and World of Warcraft [11] have become extremely popular. To handle these demanding applications, existing approaches fall usually into one of the following architectures: (a) networked servers architectures and (b) peer-to-peer architectures. A lot of work has been performed towards both these directions, including algorithms for the partitioning problem [1,7], load balancing techniques [4], awareness methods [6] and other optimization techniques [5].

The requirements and characteristics of each DVE system may vary significantly among virtual worlds for different simulated scenarios. For example, in case of an educational DVE, the consistency of the world would not be significantly affected if a number of position messages (i.e., messages sent each time a user changes his/her position) were lost. However, if position messages were lost while in a virtual battlefield, where soldiers move and run, then the sense of realism, users' awareness and performance would be significantly impacted. Therefore, the decision on the techniques and approaches used is

* Corresponding author at: Research Academic Computer Technology Institute, N. Kazantzaki Str., Patras University, GR-26500 Rion, Patras, Greece. Tel.: +30 2610 960375/996951; fax: +30 2610 969016.
E-mail address: bouras@cti.gr (C. Bouras).

1569-190X/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2012.02.006

usually related to the objectives, scope and context that each virtual world aims at supporting along with its special characteristics.

However, running and measuring the effectiveness of a large-scale DVE system is a difficult problem. On the one hand, one cannot possibly find thousands of real users to participate. On the other hand, an extended infrastructure is needed in terms of both hardware and software to support large-scale DVEs and both researchers and application designers do not always have access to such an extended infrastructure. To address this issue, this paper presents a simulation modelling tool for networked servers DVEs that comes to fill in a gap in the area of simulation tools for this kind of system. The main goal of the simulation model presented in this paper is to provide a simulation framework that can be used by application designers in order to select and evaluate the appropriate scheme, algorithm and technique for the virtual world scenario they are called to simulate, without the need for extensive software and hardware infrastructures. The simulation model allows for full customization for both generic and scenario-specific DVE systems. The formulation of various DVE systems in combination with a number of existing performance improvement algorithms and techniques that the model incorporates can provide insight and valuable information for the behaviour and performance of the DVE under different approaches and system settings. More specifically, the simulation model has been designed in order to include most of the entities, processes, parameters and techniques used in the majority of both generic and scenario-specific DVE systems. The design and implementation of the simulation model derives from a definition for the issue of "optimized" performance for DVE systems as an operational management problem. Therefore, one of the innovative aspects of the simulation model is that it is based on "translating" the DVE system's requirements to the concepts of operational management. Thus, the design and implementation of the framework is conducted with SIMUL8 [12], an integrated environment for working with simulation models, widely used in operational management. The simulation model has been validated through a comparison with the results produced by a real DVE system. The results of the validation showed that the simulation model achieves high level of accuracy in representing and modelling the behaviour and performance of a real DVE system.

The rest of the paper is structured as follows: Section 2 presents the motivation for the work presented as well as related work in the area of simulation tools for evaluating DVEs' performance, while Section 3 describes the main concepts, entities, processes and characteristics of generic DVEs along with the performance issues that these systems need to address. Section 4 presents the simulation framework in terms of its basic entities and their "mapping" to SIMUL8 objects. Section 5 presents the STEADiVE simulation model in terms of the algorithms and techniques integrated for simulating DVEs' behaviour along with the main results produced by the model. Section 6 presents the model validation for proving the model's accuracy and validity and Section 7 presents an indicative experiment conducted in order to demonstrate the way that the simulation tool can be used for assessing different techniques. Finally, Section 8 concludes the paper and Section 9 presents some planned next steps.

2. Motivation and related work

One of the basic problems when designing and implementing algorithms, methods and techniques for large-scale DVEs is the way that their efficiency could be examined. In most cases, the evaluation is based on theoretical models that often fail to meet the conditions met in real DVE systems. More specifically, an extended infrastructure is needed in terms of both hardware and software to support large-scale DVEs. As both researchers and application designers do not always have access to such an extended infrastructure, the assessment and evaluation of developed techniques becomes extremely difficult.

The problems and challenges that a DVE system needs to address as well as the algorithms and techniques for performance improvement are strongly related to the architectural solution that the system adopts. As mentioned above, the architectural approaches for large-scale DVEs fall usually into one of the following architectures: (a) networked servers architectures and (b) peer-to-peer architectures. Regarding peer-to-peer (P2P) DVE systems, there is a range of simulators available that simulate the behaviour of these systems and allow for performance monitoring of different techniques. In [16] the authors present a simulator for P2P DVE systems that reproduces the behaviour of a P2P DVE system and is capable of performing the evaluation, parameterization and result acquisition of DVEs. The simulator is composed of two different kind of applications, written in C++ that implement the clients and the central manager to whom the clients should connect to in order to join the system. Both type of applications use different threads for managing the connections that should be established, while the communications are implemented by means of sockets. In the direction of P2P DVE systems Adam [17] is another simulator developed for monitoring and assessing the performance and behaviour of this type of systems. Adam is built on top of the discrete event simulator OMNet++ [18] in conjunction with the MRIP engine Akaroa [19] and comprises a large set of measuring tools to compare the effect of different algorithms and to allow for optimization of these algorithms. Finally, VAST [20] is a light-weight network library for simulating P2P DVE systems and is also designed to be an experimental middleware consisting of three major components: network-layer, protocol and simulator. This allows new algorithms or simulators to be tried and tested for assessing their performance.

However, for the cases of networked servers DVE systems the simulation tools for representing the DVE and for monitoring and assessing its performance are rather limited. In most cases, both application designers and researchers adopt specialized methods [4] for evaluating different techniques. More specifically, most of these simulations aim at finding the effect of a specific algorithmic change or of an algorithmic parameter in an otherwise monolithic application and are, in their vast majority, completely non-transferable to other scenarios or other algorithms. A more unified approach for a simulation

modelling tool is mentioned in [7], where the authors shortly describe a simulation tool they developed. The tool is written in C++ and models the behaviour of a generic networked-servers DVE system. In particular, the tool comprises a set of multi-threaded servers and each thread uses blocking sockets for communicating with a client computer. One of the threads manages the communication with the server assigned to that client, while another thread manages user information (current position, network latency, etc.). This tool has been used by the authors for simulating the performance of large-scale networked DVE systems, where CPU usage, communication cost and the system's average response time to users' requests are monitored. However, the C++ tool is not freely available and can provide information only for generic and not application-specific DVE systems. Given the fact that the design and implementation is application or technique-specific, the reusability of such a tool for different environments and algorithms is not always successful as either quantitative or qualitative parameters are not taken into account. Finally, ScienceSim [14] is a tool with a number of services for extending OpenSimulator [13]. More specifically, OpenSimulator is an open source multi-platform, multi-user 3D application server that provides virtual worlds similar to those of Second Life. ScienceSim, among the services that it provides for extending OpenSimulator also provides simulators instrumented to collect detailed statistics about operational parameters. Furthermore, it provides a series of stress-tests for monitoring and evaluating the performance of a virtual world. However, even though ScienceSim and OpenSimulator can provide helpful information and statistics based on real DVE systems, the incorporation of algorithms and techniques requires additional programming, while infrastructure in terms of mainly hardware is also needed, especially for simulating large-scale virtual environments.

Based on the above, we could conclude that there is no complete simulation model widely available in order to simulate the processes, algorithms and techniques of a wide area of DVE systems, without the need for extended infrastructure in terms of both software and hardware. In this direction, and for overcoming this important limitation, a simulation model for accessing networked servers DVEs' performance has been designed and implemented. The proposed simulation model is an extension of Simulation Tool for Evaluating and Assessing Distributed Virtual Environments (STEADiVE) [8]. In contrast to the initial version that took into account generic DVE parameters and entities, the version of STEADiVE presented in this paper has been extended in order to incorporate DVE-specific parameters and entities so as to simulate a wider range of scenarios, algorithms and techniques. More specifically, the presented version of STEADiVE model incorporates both avatars and objects as the main entities of a virtual world, takes into account the most common distribution parameters for these entities and includes some of the partitioning and load balancing algorithms that have proven to achieve good results in bibliography the last few years. Furthermore, the updated version provides an intuitive interface that allows designers and stakeholders to fully customize and parameterize each DVE system along with the system's special characteristics. As mentioned above, the concept of STEADiVE is based on translating DVE system requirements to the concepts of operational management. For being consistent with this objective, the design and implementation of the simulation model has been realized with SIMUL8, an integrated environment for working with simulation models, mainly focused and widely used in operational management. One of the main difficulties and challenges that had to be faced during the design and implementation process was the incorporation of detailed aspects of DVEs, such as neighbouring avatars concept, entities distribution in the virtual world, synchronization messages, computer resources and routing techniques, in the set of options provided by SIMUL8, as these options are more general and are directed to classic operational management problems. Having overcome this difficulty, the simulation tool implemented incorporates all basic entities, processes, factors and parameters that affect system's performance, and through an easy and comprehensible interface allows for the customization and evaluation of diverse techniques, easily tailored to the specific needs of each DVE. As mentioned above, the simulation tool could be used by designers and stakeholders of DVE systems for simulating the performance of their approaches under different scenarios and system settings.

At this point it should be mentioned that due to the fact that STEADiVE constitutes the first complete environment for simulating the behaviour of different DVEs and given the fact that existing work in the field is either more abstract or in another DVE direction (i.e. P2P environments) the comparison of the model with other tools was not feasible. However, a comparison of STEADiVE's results has been conducted with the results provided by a real DVE system, which is of added value for the validity of the model's output. This comparison is presented in detail in the model validation section.

3. Distributed Virtual Environments

This section presents the main entities, characteristics and events that take place in a generic DVE system. The main purpose of this section is to provide an insight on the processes that need to be taken into account for the design and implementation of the simulation model.

3.1. Main concepts and processes

A virtual environment could be considered as a simulation generated by a computer, which can simulate either an imaginary or real world. Even though these environments can be two-dimensional, the term is mainly related to three-dimensional environments that aim at providing to the users a high sense of realism by incorporating realistic 3D graphics. In DVEs the simulated world runs not on one computer system but on several that are connected over a network, as presented in Fig. 1 [8]. Users that connect to these systems are able to interact in real time, sharing the same view of the virtual world.

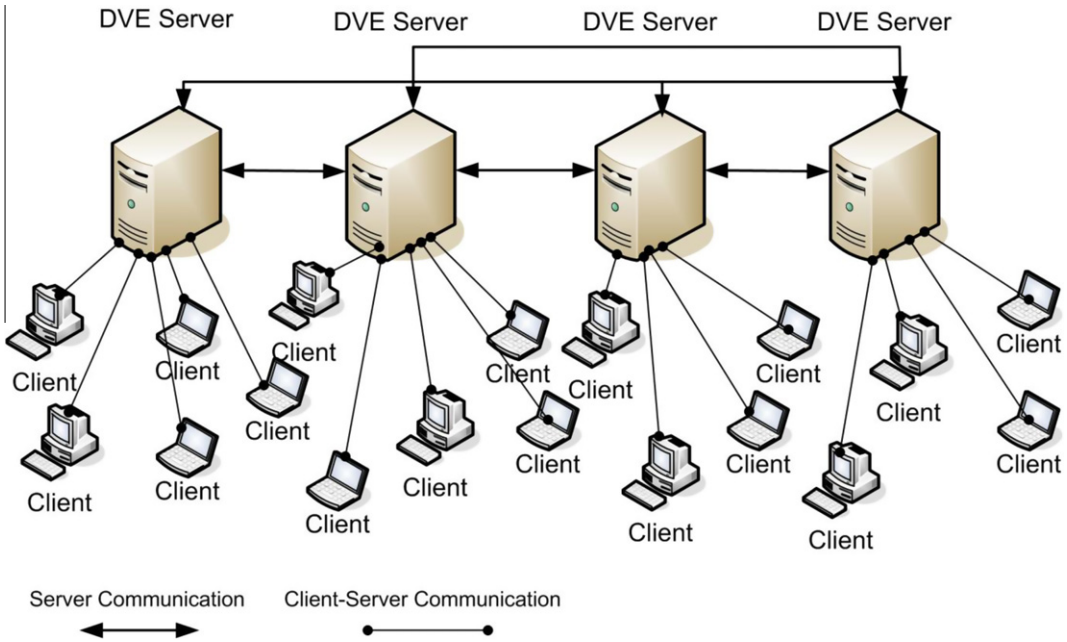

Fig. 1. Multi-server DVE architecture.

DVEs aim at supporting a tremendous number of concurrent users, scattered around the globe. The participants constitute active parts of the DVE, usually represented by human-like entities, called avatars for enhancing the awareness [3]. The state and behaviour of each avatar is controlled by the user through the client computer. Connected users can view the virtual world on their computer (client), thus having their own local copy of the virtual scene. Apart from the users, DVEs are, on their vast majority, also comprised by non-autonomous entities, called objects, which constitute the graphical representation of entities that are placed within the virtual world for supporting the context of the simulated scenario (e.g. trees, books, chairs, weapons, etc.). According to the nature of the DVE and the scenario it simulates, these objects could be static or moving, interactive or not.

In the majority of existing DVE systems, users have the ability to navigate in the virtual world, thus changing their position coordinates, interact with the objects of the virtual environment, thus changing some of their attributes (such as location, shape, and colour), interact and communicate with other participating users. To achieve high-sense of realism and maintain consistency, it is of critical importance that all connected users are always aware both of the presence of other entities (users or objects) as well as of any actions performed.

Each time that a user connects to the DVE system a request is sent to a central server, which is denoted as ConMan Server. This server performs two main tasks: (a) accepts and authorizes the connection requests from the users and redirects them to the appropriate server, (b) monitors the performance of the working servers over a period of time and acts when it is identified that one or more servers need to be unloaded. When users' avatars enter the virtual world, and from the moment they are assigned to a server, they start to navigate, interact and perform actions, thus sending messages/requests to the servers. Each of the servers of the DVE system is responsible for processing and serving these requests/messages initialized by the users, perform all the necessary updates to the virtual scene and notify all concerned users about the updates. When users connect to the DVE system, they are assigned to one of the available servers.

This assignment is based on the algorithms and techniques that each DVE adopts for handling its resources and for achieving load balancing among the servers. Throughout the users' presence in the system, the server that is responsible accepts the messages produced by all avatars that it handles, processes these messages and updates the state of the virtual world accordingly. Then, it propagates those changes to the avatars concerned thus, modifying and synchronizing their view of the virtual world, as presented in Fig. 2 [8].

3.2. Performance issues in DVEs

DVEs are characterized by a dynamically changing state, with users joining, navigating, interacting, communicating, and leaving the virtual world randomly. These changes, which are not easy to predict, may lead to situations where some regions of the virtual world are overcrowded while others are under-populated. In these cases, load rebalancing is needed for maintaining the world's consistency and effective performance. The frequency of rebalancing or redistribution of workload may affect the overall performance of the DVE system as well as the user's experience. Therefore, redistribution is necessary so

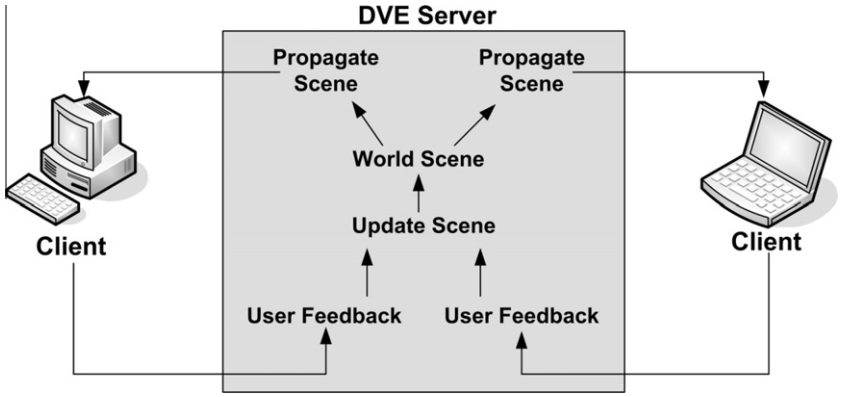

Fig. 2. DVE client–server communication.

that inconsistencies and performance degradation are avoided. The load balancing scheme that each DVE system adopts could assist in preventing frequent redistributions. Therefore, load balancing techniques have a threefold objective related to: (a) the effective distribution of workload among the system servers for optimum resource utilization, (b) the minimization of the communication cost among the system's servers in order to make the DVE system more viable and tolerant to changes, and (c) minimization of the system latency to users' requests.

- **Load Distribution**: In distributed systems, the available servers can significantly differ in their processing capabilities in terms of CPU and memory. For example, while a system server might be able to handle up to 100 concurrent users, another one may be limited to 30. The heterogeneity of the servers that comprise the system is an important factor that needs to be accounted for when discussing load balancing and distribution. Load distribution is essential not only for the optimal support of the DVE system and resource exploitation but also for avoiding system saturation. More specifically, DVE systems reach saturation when any of the available servers reaches 100% of CPU utilization [7]. This leads to a dramatic decrease of the overall system performance and severe awareness damage. On this basis, load distribution techniques should focus on making the system more resistant to continuous changing states over time.
- **Communication Cost**: Communication cost or inter-server communication cost is defined by the number and size of the messages exchanged among the servers of the system for synchronization and in order to support a consistent view among all users. Therefore, the communication cost among connected servers is an important parameter that needs to be taken into account when addressing load distribution [4]. The effect that communication cost can have in a system is: (a) significant increase of network load, (b) introduction of delays in message delivery, and/or (c) deterioration of the system performance and the user's experience. Therefore, the management of communication cost is critical, as system developers have no means to determine or even influence either the network bandwidth or capacity.
- **Average System Response (ASR)**: The latency of the system to users' requests is an important factor to be taken into account in DVE systems [6]. As in DVEs latency cannot be measured properly, the round-trip delay (RTT) or ASR (Average Response Time) is used instead [6]. Based on the work presented in [9], if the ASR is not greater than 250 ms, then users perceive that the system responds quickly. Therefore, load balancing techniques should ensure that the ASR provided is below this threshold.

Given these challenges, load distribution, communication cost and latency comprise three of the most critical factors for the performance of DVE systems. Furthermore, these factors are the ones monitored and used in the majority of existing work in bibliography [4–7] in order to assess and evaluate the behaviour and performance of DVE systems. Therefore, load distribution, communication cost and latency are the main factors that should be accounted for in resource management schemes and, in turn, in the simulation model designed. Based on the above a networked servers DVE system needs to identify the optimum solution for the following problem:

> *In a system including a certain number of servers, with predefined processing power, the goal is to identify the optimal assignment of resources to serve as many users as possible, minimizing at the same time the communication cost. This needs to be achieved with guaranteed efficiency based on each application's special requirements.*

In this context, the *optimal assignment of resources* refers to the number of the system's servers that need to be activated for handling the workload introduced by the users at a given time interval. On the other hand, the *guaranteed efficiency* refers to the tolerance of the system regarding realism/awareness, latency and performance. Finally, the *minimization of the communication cost* is related to the identification of such a distribution scheme which ensures that the messages exchanged among the active servers of the system are only the necessary ones. Thus, the selection of an effective load distribution scheme is derived from the intersection of the above parameters. This problem falls into the area of operational manage-

ment, where the efficiency is defined by the boundaries set for the CPU usage. It is worth mentioning that the importance of each parameter may vary among different DVE applications, with different characteristics and techniques used. Therefore, the optimum solution is DVE-specific.

4. Simulation framework

As mentioned above, the approach presented faces the problem stated as an operational management one. In this direction, a discrete event simulation model is developed using SIMUL8 (version 14). SIMUL8 [12] is an integrated environment for working with simulation models and its use is mainly focused to operational management problems. It is based on a powerful language and exhibits visualization capabilities that allow the creation of accurate, flexible, and detailed simulations in a reasonable time. It also has several features (trials, warm-up period, random sampling, etc.) allowing one to conduct statistical analysis of the simulation output. The simulation model is developed in order to provide insights into the workflow processes of DVE systems and to estimate system performance measures.

4.1. SIMUL8. Objects

This section presents briefly some of the basic objects that SIMUL8 provides for designing and creating models, whose behaviour can be simulated.

Work Centers: A Work Centre is a place where work takes place on Work Items. Work done at work centers usually takes up time and sometimes requires the availability of resources.

Storage Bins (Queues): A storage bin is a place where work to be done can wait until appropriate resources or work centers are available.

Work Entry Points: A work entry point is a place where work to be done appears in the model for the first time.

Work Exit Points: A Work Exit Point is a place where work that is complete (or otherwise "finished") leaves the model. At the point in time when each work item leaves, data is recorded about how long it has spent in the model (from the time when it entered through a "Work Entry Point").

Work Items: A Work Item is the work which is done in the organization being simulated. Work Items flow through the simulation, being stored in Storage Areas, and acted upon by work Centers. Work Items can be allowed to "expire" while in a storage bin.

Components: Components consist of one or more existing objects (either the standard ones or other Components) that are tailored in some way and are then saved as a single new object for future use.

4.2. DVE simulation entities

This section presents the main entities of the simulation model. At this point it should be mentioned that the entities of the DVE system are mapped to the simulation objects provided by SIMUL8 tool, as presented in Table 1.

4.2.1. DVE generic entities

This sub-section presents the basic entities found in the majority of DVE systems as well as the way they are represented in the STEADiVE simulation model.

Virtual World Entry Point: This entity corresponds to the *Work Entry Point* of SIMUL8. For the DVEs' simulation, the *Work Entry Point* represents the point where users' avatars enter the system. Each *Entry Point* is characterized by the distribution used for initializing and sending messages as well as the inter-arrival times between messages.

Messages: In the DVE simulation model there are two types of messages taken into account and are presented as *Work Items*. The first type is called "avatar" and the second one is called "request". The avatar messages represent the actual avatars that enter the system, which means that for each avatar there is an avatar message initialized. Each avatar message is labelled with seven labels that are: a unique identifier (avatar_id), the time that the avatar spends in the system (avatar_life), a timestamp with the moment that the avatar entered the virtual world (avatar_entry_time), three labels with the avatars'

Table 1
Mapping of DVE entities to SIMUL8 objects.

DVE generic entities	SIMUL8 objects
Avatar	Work Item
Virtual World Entry Point	Work Entry Point
Avatar Messages	Work Items
ConMan Server	Component
DVE Servers	Components
DVE-specific entities	**SIMUL8 objects**
Cell	Work Item
Object	Work Item

coordinates (avatar_x, avatar_y and avatar_z) and a label that represents the server that handles this avatar (server_id). The request messages represent the messages sent by each avatar and are labelled with a parent_id label (the value of this label is the avatar_id of the avatar that sent the message). For simplification purposes, we consider that all messages sent by the users' avatars are of the same type and require the same resources. This simplification is mainly related to the fact that in the simulation model we consider a general approach of a DVE system, where message types (e.g. position messages, object modification messages, chat messages) and attributes are not distinguished. However, it should be mentioned, that the simulation tool provides the necessary functionality to tailor the messages exchanged by using additional labels attached to each message type.

ConMan Entity: The ConMan Server entity used in the DVE model is a combination of a *Work Centre* and a *Storage Bin* object (of SIMUL8 library). The ConMan Server is connected to the Virtual *World Entry Point* and the messages that arrive, which represent avatars, first move to the ConMan *Queue* and then to the ConMan *Work Centre*, where they are processed. The processing implies that ConMan first labels the messages with a unique identifier (avatar_id) and sets the life of each of these messages (avatar_life). The life of each message represents the average time that a user spends in the DVE system.

Server Entity: The Server entity used in the DVE model is a *Component* of SIMUL8, and constitutes a combination of three Queues and Work Centres and two Work Exit Points, as presented in Fig. 3. Each of these objects serves different tasks within the model. Work done at *Work Centre*s usually takes up time and sometimes requires the availability of resources. For the DVE simulation, we consider that the time it takes to process each message depends on the server's processing capabilities and this parameter could be adjusted by the system designers based on the infrastructure they plan to use.

4.2.2. DVE-specific entities

Apart from the basic DVE entities used in the majority of DVE systems, there are also DVE-specific entities that are used by alternative DVE management techniques. These entities are included and supported by STEADiVE simulation model and could be either included in the model for the evaluation of the techniques that adopt them, or ignored, without affecting the representation of the simulated technique.

Messages: As in the case of avatars' representation, in the simulation model two special types of messages are defined and described as *Work Items*. The first type is called cell and the second one object. Cell messages represent the actual cells that the virtual world is divided in. These *Work Items* are created at the initialization of the simulation. Cell *Work Items* are directly stored in a *Storage Bin* (named Virtual World Area) and remain there unaffected throughout the simulation duration. Each Cell *Work Item* is dynamically labelled with three values: the cell unique id and other two that describe the coordinates of the cell so as to define the cell's location in the virtual world. The second type is called Object. Object *Work Items* represent the objects that the virtual world comprises and are created at the initialization of the simulation. Object *Work Items* are directly stored in a *Storage Bin* (Object Repository) and remain there throughout the duration of the simulation. Each Object *Work Item* is dynamically labelled with six values: object unique id, three labels that describe the object's coordinates in the virtual world and the cell id that the object is located and the server_id that represents the server that handles this object. The values of these labels are modified throughout the simulation so as to simulate the changes of the virtual world.

5. STEADiVE simulation model

This section describes the sequence and workflow followed throughout the DVE simulation model along with the logic that runs on the back.

5.1. DVE workflow

The detailed simulation model of the DVE system is presented in Fig. 3. In this figure, the *Components* (Servers) are left open so that the sequence of events that take place through a DVE session are clearer to the reader. The first step for the

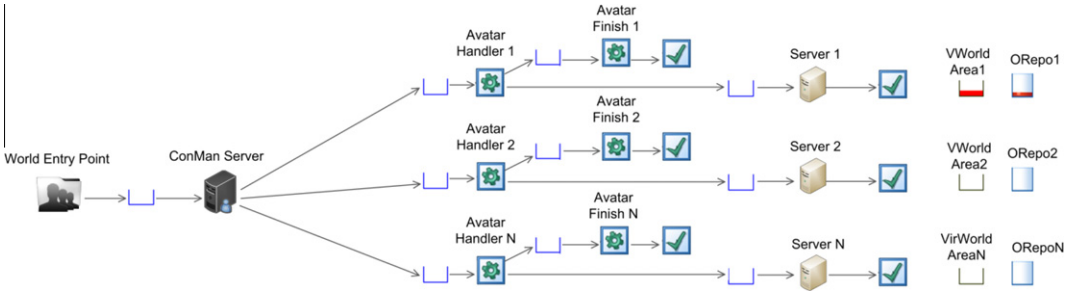

Fig. 3. Detailed DVE simulation model.

simulation process is the formulation of the virtual world. The formulation is realized with the creation of cells. Depending on the number of cells upon which the virtual world is divided the necessary "cell messages" are created and are inserted in the Virtual World Area Storage Bin. The step that follows is the representation of objects. Depending on the number of objects that the virtual world comprises the necessary "object messages" are created and are inserted in the Object Repository Storage Bin. Each "object message" is labelled with values for *x*, *y* and *z* axis in order to locate itself within the virtual world. This labelling is then used by the simulation model in order to calculate the interaction among users' avatars and objects as well as the communication cost that these interactions could introduce among the system servers.

Users enter the DVE system through the Virtual World Entry Point and for each of the connected users an "avatar" message is initialized. The distribution followed for avatars entrance in the system, could be set in the simulation model according to each application's characteristics. Avatar connection messages are forwarded to the ConMan server that labels each of these messages with a unique identifier (avatar_id) as well as an "avatar life" value. Given the fact that the avatar message represents the actual users that participate in the DVE, the avatar life attribute is used for defining the approximate period of time (simulation time) that an avatar will spend in the DVE system. This attribute could be set according either to existing data for users' presence in the virtual world or to approximate values. After labelling avatar messages, the ConMan Server forwards the messages to one of the available servers of the DVE system. The selection of the destination is a process that may vary among real DVE systems and is related to the technique adopted for the partitioning of the virtual world, for load and resource management. The resource assignment techniques that are used in the majority of existing applications are simulated in STEADiVE and are presented in detail in the section that follows.

When avatar messages are forwarded to the appropriate server, they enter a queue, waiting to be added to the server's list. The server's list is represented with a *Work Centre* object named "S_{num}_Avat_Handler". Upon receipt of an avatar message the server updates its list of existing avatars and forwards the message to a Storage Bin object. This Storage Bin object for the simulation model represents the list where all active avatars exist. The avatars stay in this bin for as long as their "avatar life" attribute defines and then they (expire). Expired avatar messages are collected by a work center called "S_{num}_Avat_Finish", which acts as the information point of the users, who have been served and have exited the system.

As mentioned above, avatars in a DVE system have the ability to navigate in the virtual world, interact with both objects and other users' avatars and communicate, thus generating messages. For simulating this behaviour, the simulation model generates request messages, which represent the messages realized by connected users. In particular, for as long as the avatar messages are present (while in the storage bin and before they expire), "request messages" are initialized. These messages are labelled with a "parent_id", which represents the avatar that sent the message. The requests send by the users in an actual DVE system cannot be known, neither to their number or frequency, as they depend on each user's individual behaviour and the nature of the simulated virtual world. In particular, in a battle simulation users would generate a tremendous number of users while running or fighting, while in a class simulation, the messages would be less due to the fact that avatars would be seated and would attend the lecture. Therefore, the inter-arrival time of the request messages differs from one DVE to another and should be properly set by the designers. In any case and for any inter-arrival period, the simulation model, through the Time Check Logic of SIMUL8 and Visual Logic, implements the necessary functionality that checks the number of avatars in the storage bin every a fixed time interval and generates a request for each of these avatars. The request messages are placed in the "S_{num}_Queue" waiting for the server to process them.

The actual processing unit of the server is represented by the Work Centre titled "S_{num}_Server", where all request messages are processed. Each server of the DVE system is characterized by its processing power, which in the simulation model is represented by the time it takes to the server to process a message. For monitoring servers' performance a utilization parameter was defined to the simulation model for each of the servers (denoted as S_{num}_CPU_Utilization).

5.2. Integration of algorithms and techniques

This section presents the DVE techniques that have been developed and represented in the STEADiVE simulation model along with some basic parameters that are necessary for the detailed system representation.

5.2.1. Entities' distribution

The entities' distribution types simulated in STEADiVE are the ones mainly used in literature and are the following [4]:

- **Uniform**: A uniform distribution is the one for which the probability of occurrence is the same for all values of *X*. If a uniform distribution is divided into equally spaced intervals, there will be an equal number of members of the population in each interval. In the case of the virtual environment, the uniform distribution implies that the entities of the platform are uniformly placed within the virtual space.
- **Skewed**: A skewed distribution is the non-symmetrical one. For skewed distributions, it is quite common to have one tail of the distribution considerably longer or drawn out relative to the other tail. A "skewed right" distribution is one in which the tail is on the right side. A "skewed left" distribution is one in which the tail is on the left side. In the case of the virtual environment, skewness implies that there is a certain area where entities are concentrated while the rest of the space is sparsely populated.

- **Clustered**: This type of distribution implies that the population is mainly concentrated around certain points. The number of the concentration points defines the number of the clusters available. In the case of the virtual environment, the clustered distribution implies that the majority of the existing entities are placed around the available clusters, while the rest of the space is sparsely populated.

During the creation of an object or the arrival of an avatar, their coordinates in the virtual environment are calculated based on the distribution that has been selected.

5.2.2. Resource assignment techniques

As mentioned above, one of the main factors for DVEs' performance is the resource assignment technique that each approach adopts. The resource assignment techniques simulated in STEADiVE are the ones used in the majority of existing DVE systems and are the following:

- **Linear Classic**: This approach is a representation of the LOT technique [4] and is proven to achieve good results for DVE systems. In this approach all servers are assigned with users and load balancing takes place every a fixed time interval. The routing technique used for assigning incoming users to the servers of the system adopts a circular approach. In the circular approach entities are forwarded in a circular way to the available servers of the system. Finally, rebalancing involves the re-assignment of existing workload to all connected servers of the system.
- **Linear Dynamic**: This approach is a modification of the Linear Classic one and the main change involves the adaptation of a dynamic approach for the activation of the servers. More specifically, for the dynamic approaches, servers are used on an on-demand basis and are activated or deactivated according to the needs and demands of the system, as they change over time [2]. For the dynamic approaches when a defined CPU threshold is reached, a new server is activated and the rebalancing does not take place every fixed period of time, but instead, it happens when one of the already connected servers reaches the threshold. At this point it should be mentioned that the rebalancing, once again, concerns all connected servers, while the routing technique used is the one of equal probability, as entities are forwarded with an equal probability profile to the servers of the system.
- **Equally Balanced Dynamic**: In this approach the dynamic scheme is fully adopted for the assignment of workload to the servers of the system. When a defined CPU threshold is reached, a new server is activated. The main difference of this approach to the Linear Dynamic one is that when rebalancing takes place, the existing workload is not distributed among all connected servers of the system. Instead it is re-distributed between the overloaded and the newly activated one. As for the routing technique, the approach adopts the equal probability, in terms that incoming users, and thus workload, is distributed equally among existing servers.
- **Spatial Dynamic**: This approach is based on a spatial partitioning of the virtual world [15]. In particular, each connected server is assigned with a certain area of the virtual world (in terms of virtual world cells) and consequently with all the entities, users and objects that enter this area. When one of the servers reaches the CPU threshold then rebalancing takes place between the overloaded one and the newly assigned. During rebalancing, the overloaded area, in terms of virtual space is divided, following a quad tree structure. Regarding the routing technique, given the fact that this approach is space oriented, we consider that the routing follows the same principle. For example, if a server, which handled area A, was visited by incoming users with a probability of 50%, then during the rebalancing of this server, the visiting probability would fall to 25% for this server and 25% for the newly assigned server.

Resource assignment techniques vary, among others, in the way that rebalancing is triggered. More specifically, some of the techniques perform rebalancing after a fixed period of time (Linear Classic) while dynamic techniques (Linear Dynamic, Equally Balanced Dynamic and Spatial Dynamic) initialize rebalancing by adopting the concept of thresholds. For simulating the rebalancing triggering process, there are functions developed in the Time Check Logic and Visual Logic of SIMUL8. Especially for the dynamic approaches the concept of CPU thresholds is adopted. Through the Time Check Logic of SIMUL8 the model checks on fixed time intervals the CPU utilization and in the cases that one or more that the servers are either overloaded or underused (a threshold is reached), the ConMan Server modifies the resource assignment technique for load balancing among the system's servers.

5.3. Results produced

SIMUL8 provides the ability to monitor, store and analyse a number of parameters throughout the simulation. In STEADiVE simulation model the basic parameters recorded are related to the server workload, the inter-server communication cost and the average system response time to users' requests.

- **Server Workload**: For monitoring and analysing the server workload for the STEADiVE model, the model implements though the Time Check Logic of SIMUL8 a function (Cpu_Avatars_Collection_Results) that is called at regular time intervals (every 60 s). This function gathers the CPU Utilization of all servers and stores the results in an excel file for further elaboration and analysis.

- **Communication Cost**: the calculation of the communication cost in the simulation model is realized as follows: every time that a "request message" is sent by the avatars of the system a function checks the parent_id of this message. The step that follows is the creation of a list with all entities that exist within the area of interest of the avatar (by comparing their coordinates) and for each of the entities their server_id is extracted. If the server_id of the entities that lie in the area of interest of the avatar is different than the server_id of the avatar that sent the request message then the message is stored in an excel file along with the id of the request message, the parent_id of the message, the server_id of the avatar that issued the message, the server_id/ids of the entities that received the message and a timestamp for the moment that the message was sent. When the simulation is completed the produced file is analysed and the total communication cost among the system servers is calculated.
- **Average System Response (ASR)**: the calculation of ASR in the STEADiVE model is realized as follows: every time that a server processes a message (the message leaves the Server Queue and inserts the Exit Request Finished) the server checks the parent_id of the message. If the parent_id of the message corresponds to an entity that this server does not handle then in an excel file the following values are stored: the id of the request message, its parent_id, the server_id of the avatar that sent the message and a timestamp for the moment that the message exited the Work Center of the server. When the simulation completes the produced file is combined with the excel file produced for the calculation of the communication cost and through a matching of the ids of the request messages and the difference between the values of the two timestamps (value when message was processed and value when the message was sent) the ASR is calculated.

These parameters are in accordance to the main factors (load distribution, communication cost and ASR) that affect DVE performance, as monitored and used in bibliography [4–7].

6. Model validation

The step that follows the design and implementation of STEADiVE simulation model is the actual validation of the model. The validation is necessary in order to ensure that the results produced by the simulation model are accurate, in terms that the model provides similar results to the ones produced by a real DVE system. For the model validation a small-scale experiment was conducted with a DVE system comprising three computers. One of the three computers was acting both as a ConMan server and as a standard server. The specifications of this computer were an Intel Core2 Duo T9300 processor at 2.50 GHz with 4 GB RAM, while the operating system was Windows 7 (32bit version). The specifications of the other two computers of the DVE system were Intel Core2 Duo T9900 processor at 2.80 GHz with 4 GB RAM, while the operating systems were Ubuntu Maverick 10.10 (32bit version). The three computers were on the same LAN, interconnected with a CISCO Catalyst 3500XL switch (100 Mbps).

As mentioned above, for the model validation the methodology followed was to monitor the performance of the real DVE system in terms of CPU usage, inter-server communication cost and ASR of users' requests, since these are the main parameters taken into account for the evaluation of DVE's performance. These parameters were monitored for a virtual world scenario that was implemented for the real DVE system and the same scenario was then simulated to the STEADiVE model. For the scenario presented, approximately 100 runs were executed and the results correspond to the average values obtained from these runs.

6.1. Virtual world scenario description

For the validation scenario a virtual world was created with dimensions 600×600 units that is distributed among the servers with the Linear Classic partitioning technique. Thus, each partition (server) handles an area of 200×200 units. The scenario considers that users' avatars enter the virtual world and move randomly. The duration of the experiments is set to 5 min and considers 300 avatars that enter the virtual world in groups of 75 avatars every 90 s.

The creation of the virtual world for the real DVE system was implemented with OpenSimulator [13], an open source multi-platform, multi-user 3D application server. Fig. 4 presents a screenshot of the virtual world created with OpenSimulator. The users' entrance, movement and interactions in the virtual world were implemented with a modified version of a stress-test provided by ScienceSim [14]. ScienceSim is a tool with a number of services for extending OpenSimulator as well as a series of stress-tests available for monitoring and evaluating the performance of a virtual world.

6.2. Validation results

This section presents the results of the small-scale experiments conducted for validating the STEADiVE model. Fig. 5 presents the percentage of the CPU usage for the 5 min duration of the experiments for both the real DVE system and the STEADiVE model for one of the system servers. More specifically, given the fact that the partitioning approach selected is the Linear Classic one, users' avatars and the respective load is equally distributed among the system servers. Therefore, the servers present very similar results for the percentage of the CPU usage. The selection of one of the servers (instead of the three of them) in Fig. 5 is representative for the rest of the servers and has been selected for assisting in the better understanding of the plot.

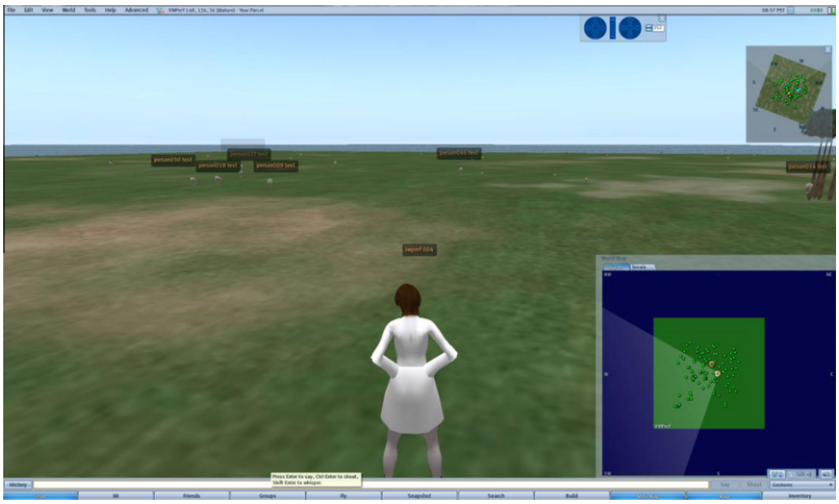

Fig. 4. Virtual world for the validation scenario.

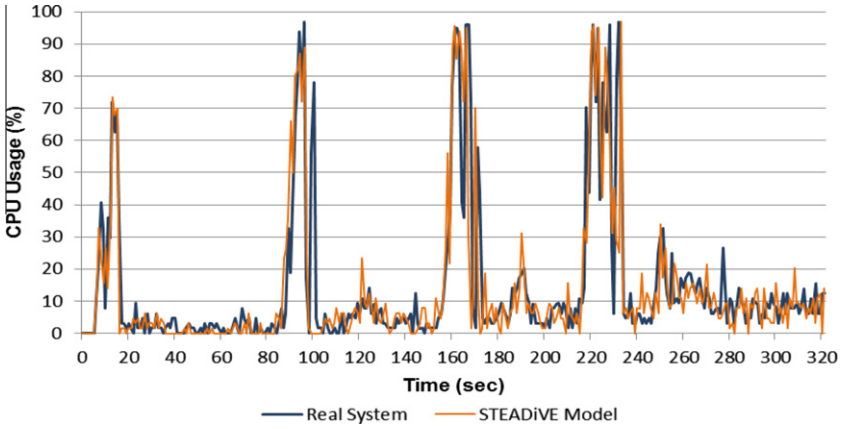

Fig. 5. Average CPU usage per system server.

As it can be seen from Fig. 5, the percentage of the CPU usage is increased as the batches of 25 users enter the virtual world in each server every 90 s and then decreases to average values. This change in the CPU usage is also captured accurately by the scenario simulated with the STEADiVE model.

More specifically, the results show that the model is able not only to successfully simulate the variations in the performance of the system as users enter and move within the system, but achieves values for the CPU usage very close to those of the real DVE system. It is worth noting that the variations and differences in values occurring between the real system and the simulation model are expected given the random movement of users in the system, which are different for every execution of the experiment without however significant variations among the runs executed. Thus, the results extracted for the accuracy of the monitored CPU usage show that the STEADiVE model provides secure and valid results for the simulated systems.

The second parameter monitored was the communication cost of both the real DVE system and the system simulated with STEADiVE. As already mentioned, the communication cost is the number of messages exchanged among system servers in order to maintain consistency. Fig. 6 presents the results of the experiments for the average communication cost of the real system and the STEADiVE model. The communication cost presents an incremental trend as the batches of users enter the virtual world, move and interact with each other. The increases are more dramatic each time that new batches of users enter the virtual environment and then the cost continues to increase until the end of the experiment. The results show that the STEADiVE model manages to capture with high proximity the number of messages exchanged among the system servers thus providing values for the communication cost very close to the actual values of a real DVE system. More specifically, the average deviation of the communication cost produced by the STEADiVE model in comparison to the real system is on the order of 2.46%, a very low percentage for the simulation tool that does not affect the overall view and assessment of the system's performance.

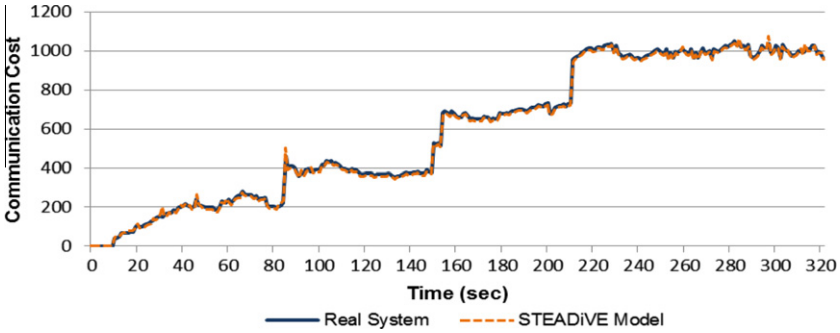

Fig. 6. Communication cost for the validation scenario.

The last parameter that needs to be monitored in order to have a full picture of the system's performance is the ASR parameter. Table 2 presents the average ASR values obtained from the experiments conducted for both the real system and the STEADiVE model.

The results show that once again the STEADiVE model calculates accurately the ASR parameter for the simulated scenario, providing a very small deviation from the results obtained by the real DVE system, on the order of 1.276%.

6.3. Discussion of the validation results

The results obtained from the validation scenario clearly demonstrate that STEADiVE simulation model manages to accurately represent a real DVE system and provide accurate results for the three basic parameters monitored and used in order to evaluate a DVE's performance. More specifically, the STEADiVE model, through the detailed representation of all processes that take place within a virtual environment manages to provide accurate results for the CPU usage, the communication cost and the ASR of the simulated virtual world.

The detailed representation of different DVE systems, the inclusion of the majority of processes that take place in a virtual environment, the incorporation of the majority of factors that can divert (i.e. avatars and objects distribution, avatars' incoming distribution, load balancing technique, etc.) among DVEs, make the STEADiVE model a powerful tool that allows designers and researchers to easily simulate, monitor and assess different aspects of a system's performance and behaviour.

Furthermore, the model also supports flexible customization of parameters as well as the inclusion of additional parameters that may be included in DVE-specific systems (i.e. different approaches for interest management, group communication, etc.). This extensibility and customization makes the STEADiVE model a tool that can support a wide range of DVE systems.

7. Using STEADiVE for large-scale DVEs

After the model validation that has shown the high level of accuracy of the results obtained with STEADiVE in comparison to the results produced of a real DVE system, this section presents the way that the model can be used for monitoring and accessing the behaviour and performance of large-scale DVE system. More specifically, this section presents an indicative simulation conducted with STEADiVE for a large-scale DVE system, where different resource assignment schemes have been adopted for different virtual world scenarios. This is only an example from the series of experiments that can be conducted with the simulation model, as different experiments can be conducted for monitoring the effect of different parameters on DVE systems. A more detailed and analytical presentation of how different parameters can affect a DVE system's behaviour and performance has been presented in [15]. More specifically, in [15] different assignment techniques have been simulated with STEADiVE and the results of their behaviour have been in detail presented. However, the main objective of this section is to demonstrate how different DVE characteristics and algorithmic approaches can be integrated in STEADiVE simulation tool, rather than comparing the different approaches and the effect they have on a DVE behaviour and performance, which was the scope of the work presented in [15]. In this direction, this section presents the setup of the experiments conducted along with the values of the parameters taken into account and the results produced by STEADiVE simulation model.

Table 2
Average ASRs and deviation.

Average ASR (in ms)		Average deviation
Real system	STEADiVE model	
183	179	~1.276%

Table 3
Experiments' parameters.

	Experiments' values	STEADiVE model options
General DVE system parameters		
DVE setup settings		
Servers number	8	Any value can be set
Mean user lifetime	40 min	Any value can be set
Experiments' duration	180 min (3 h)	Any value can be set
User's incoming distribution	Exponential	Average | Fixed | Normal | Uniform | Geometric | Poisson | Exponential|... (already provided by SIMUL8)
User's distribution in space	Uniform	Uniform | Skewed | Clustered
Objects' number	500	Any value can be set
Objects' distribution in space	Uniform	Uniform | Skewed | Clustered
Resource assignment scheme specifics		
Spatial dynamic approach		
Rebalancing triggering	Threshold based	Threshold based | Fixed time interval
CPU max threshold	80%	0–100%
CPU capable threshold	40%	0–100%
CPU min threshold	10%	0–100%
CPU check time	1sec	Any value can be set
Linear classic approach	Experiments' values	STEADiVE model options
Rebalancing triggering	Fixed time interval	Threshold based | Fixed time interval
Rebalancing time	5 min	
Scenario specifics		
Average case scenario		
User inter-arrival time	6 s	Any value can be set
User inter-request rate	0.2 s	Any value can be set
Worst case scenario	Experiments' values	STEADiVE model options
User inter-arrival time	3 s	Any value can be set
User inter-request rate	0.1 s	Any value can be set

Table 4
Servers' CPU usage for the linear classic and the spatial dynamic approach.

Scenarios	S1	S2	S3	S4	S5	S6	S7	S8
Spatial dynamic approach								
CPU usage for average case scenario	38	39.5	36	36				
CPU usage for worst case scenario	85	74	71	69	73	71	71	71
Linear classic approach								
CPU usage for average case scenario	17	18.5	21	18	17.5	19	18.5	21
CPU usage for worst case scenario	75	71	74	72	77	71	75	72

7.1. Experiments' setup

The experiments conducted consider a DVE system comprising eight servers. All of the servers available have the same computing power and are dedicated in serving the DVE system. The experiments aim to investigate the effect and performance of two different resource assignment techniques when user-driven parameters change. By user-driven parameters we mean metrics that are totally dependent on users' actions, preferences and behaviour and cannot, therefore, be a priori known to the system. The user-driven parameters that the series of experiments takes into account are the inter-arrival time and the users' activity profile. With inter-arrival time we define the rate that users join the virtual world, while the activity profile corresponds to the rate of movements and interactions that users perform in a virtual environment over time. For the experiments an average and a worst case scenario of user activity and incoming rate have been selected. In order to select the values for the user-driven metrics and considering the fact that no data from a running virtual environment were available, existing work and findings for the World of Warcraft world have been selected [21]. Regarding the resource assignment techniques that are examined in the experiments, the Linear Classic and Spatial Dynamic approach have been selected. For the dynamic approach that adopts the concept of thresholds and for the experiments conducted values for the different CPU threshold were set. Table 3 presents the general system parameters of the experiments, the resource assignment technique specifics as well as the specifics of the scenarios that will be simulated with STEADiVE. For each parameter the table includes the value of the presented experiments as well as the options provided by STEADiVE for each parameter. It becomes clear

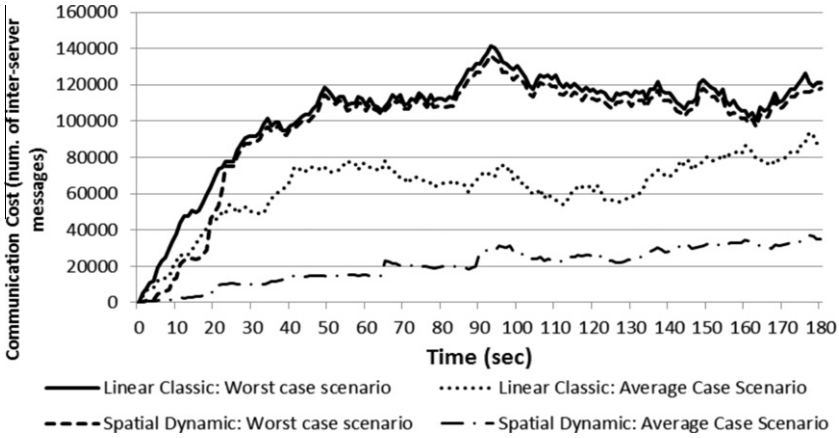

Fig. 7. Communication cost over time for the different scenarios.

Table 5
Average ASRs for the different approaches.

Scenario	Average ASR (in ms)	
	Spatial approach	Linear classic approach
Worst case scenario	252	258
Average scenario	214	236

that a selection of a different value and option for each of the parameters can lead to different DVE systems and therefore results.

For the scenarios presented, approximately 1000 runs were executed and the results correspond to the average values obtained from these runs.

7.2. Experimental results

The results of the CPU utilization of all connected servers are presented in Table 4. Columns 1 (under the heading "Scenarios") of the table present the approach used while the rest of the columns present the CPU utilization for each server *Si* ($i = 1–8$) of the DVE system. The cells of the table that are left blank indicate that the spatial dynamic approach did not activate the corresponding servers.

The results of Table 4 show that for the DVE system and the average case scenario, there is no need to activate all eight servers in order to support the users. More specifically, the Spatial Dynamic approach demonstrates that when adopted for the specific DVE system and scenario can effectively support the system's load with four servers while achieving balanced workload. Regarding the worst case scenario, Table 4 shows that the results obtain for the workload distribution are very similar for the two resource assignment techniques. More specifically, for this heavy load case, the Spatial Dynamic approach needs to activate all servers available in order to support the users that enter the DVE system and the heavy load they introduce. In terms of workload distribution among the system's servers, it can be seen that both approaches achieve relatively balanced workload without reaching the prohibitive threshold of 100% of CPU utilization.

Fig. 7 presents the communication cost for the Linear Classic and the Spatial Dynamic approach for both scenarios examined. The figure clearly demonstrates that the communication cost for the Spatial Dynamic approach is significantly reduced when compared to the communication cost introduced by the Linear Classic approach for the average case scenario of user presence and participation. This is explained by the fact that in the case of multiple running servers, even though there is no actual need, a greater number of messages need to be exchanged among servers for maintaining consistency. Thus in the Linear Classic approach communication messages are exchanged among eight servers, when in the Spatial approach only four servers are concerned. Given that DVEs are strongly dependent on the underlying network characteristics (i.e. delay, delay jitter, throughput), the reduction of the messages exchanged among the system's servers is of great value for the viability, scalability and performance of the system. Regarding the worst case scenario, Fig. 7 shows that both approaches converge when it comes to communication cost. This can be explained by the fact that the Spatial Dynamic approach needs to activate all available servers and thus, more messages need to be sent among the system's servers.

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 [6]. The

average ASRs in milliseconds obtained for the experiments conducted are presented in Table 5. Based on [9], if the ASR is not greater than 250 ms, then users perceive that the system responds quickly.

For the average case scenario the results obtained from the experiments illustrate that both approaches achieve an average ASR below the 250 ms threshold. Furthermore, the results show that the Spatial Dynamic approach achieves better results when compared to the Linear Classic one. The difference in the average ASRs of the two approaches can be explained by the fact that in cases where the system load is not very heavy and less servers can support the DVE system, then a greater number of neighbour avatars of a given avatar *i* are assigned to the same server as *i*. When the neighbour avatars of *i* are assigned to the same server, then the messages sent by *i* do not have to travel to another server in order to reach their destinations, and the average ASR is lower.

Finally, for the worst case scenario, both approaches exceed the 250 ms threshold. However, the average values obtained for both approaches are very close to the defined threshold and it is not expected that it will affect user's experience.

7.3. Discussion of the results

This section presented a series of experiments that demonstrate the way that STEADiVE tool can be used for monitoring and evaluating DVE systems with different settings and characteristics. For the series of experiments conducted, one could conclude that for DVE systems with an average user presence and participation, the Spatial Dynamic approach would perform better, when compared to the Linear Classic approach. More specifically, the CPU utilization and server usage achieved with the Spatial Dynamic approach combined with the significantly reduced communication cost and lower ASR could improve the performance of the overall DVE system. Therefore, the Spatial approach would be more effective to adopt in the case of a DVE with similar characteristics as those defined in the average case scenario. In the case of the worst case scenario, both approaches seem to achieve similar results. Therefore, any of the two techniques could be used for supporting the DVE system.

8. Conclusion

This paper presented a simulation model for networked servers DVEs, called STEADiVE that was implemented using SIMUL8 simulation tool. The entities, processes and characteristics of both generic and scenario-specific DVEs were translated into SIMUL8 objects and the necessary logic was programmed and added for covering all important aspects of these dynamic systems. The motivation behind this implementation came from the observation that one of the main issues in DVE systems, that is the resource management, could be formed into an operational management problem. In this direction, SIMUL8 was selected for designing the DVE simulation model, as it is one of the most popular and widely used tools, both in the industrial and academic area for operational management.

Furthermore, given the fact that DVEs can widely vary on the scenarios they simulate, the resources they use, the techniques and algorithms they adopt and the number of users they are called to support, it is important to have a framework that could be easily tailored for meeting the specific needs of each DVE application. The validation of the model through its comparison with a real DVE system has shown that STEADiVE is practical and achieves high level of accuracy in the simulation of the DVE system and the results it provides. Therefore, STEADiVE simulation model could be used by application designers for selecting the appropriate model for the scenario they are called to simulate as well as for evaluating optimization techniques and algorithms. STEADiVE comes to fill in the gap in the area of simulation tools for DVE systems allowing for their detailed representation as well as performance and behaviour monitoring under different approaches and system settings.

9. Future work

As often mentioned throughout this paper, the diversity of DVE systems, their specific attributes, characteristics and the scope of the scenario they simulate call for diverse techniques and approaches. In this direction some of the planned next steps include the simulation and evaluation of various existing techniques and algorithms for different types of DVE systems so as to form some guidelines on the pros and cons of each approach. Furthermore, given the high degree of customization that the simulation model supports, there is the ability for the integration of additional DVE-specific attributes and processes that consume resources, such as data replication and interest management. These processes can be added to STEADiVE model as "plugins" for further extending its capabilities. Finally, a step to be carried out is the assessment of the effect that different parameters and factors have on a DVE system as well as the experimentation with different performance optimization methods.

References

[1] C. Bouras, E. Giannaka, Th. Tsiatsos, Partitioning of distributed virtual environments based on objects' attributes, in: 11th IEEE International Symposium on Distributed Simulation and Real Time Applications, 2007, pp. 72–75.
[2] C. Bouras, E. Giannaka, Th. Tsiatsos, Performance evaluation of a dynamic approach for networked servers distributed virtual environments, in: 12th Communications and Networking Simulation Symposium, 2009.
[3] C. Joslin, I. Pandzic, N. Thalmann, Trends in networked collaborative virtual environments, Comput. Commun. 26 (2003) 430–437.
[4] J. Lui, M.F. Chan, An efficient partitioning algorithm for distributed virtual environment systems, IEEE Trans. Parallel Distrib. Syst. 3 (2002) 193–211.
[5] R.M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, P.T. Barham, Exploiting reality with multicast groups, IEEE Comput. Graph. Appl. 15 (1995) 38–45.

[6] P. Morillo, S. Rueda, M.J. Orduna, J. Duato, A latency-aware partitioning method for distributed virtual environment systems, IEEE Trans. Parallel Distrib. Syst. 18 (2007) 1215–1226.
[7] P. Morillo, S. Rueda, M.J. Orduna, M. Fernández, J. Duato, Improving the performance of distributed virtual environment systems, IEEE Trans. Parallel Distrib. Syst. 16 (2005) 637–649.
[8] C. Bouras, E. Giannaka, Th. Tsiatsos, A framework model for DVEs using SIMUL8, in: Second International Conference on Simulation Tools and Techniques – SIMUTools 2009, Rome, Italy, 2009.
[9] T. Henderson, S. Bhatti, Networked games: a QoS-sensitive application for QoS-insensitive users, in: Proceedings of the ACM Int'l Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm (SIGCOMM '03), 2003, pp. 141–147.
[10] Second Life, <http://secondlife.com/> (accessed 29.04.11).
[11] World of Warcraft, <http://www.worldofwarcraft.com/> (accessed 29.04.11).
[12] SIMUL8, <http://www.simul8.com> (accessed 29.04.11).
[13] OpenSimulator: <http://www.opensimulator.org/> (accessed 29.04.11).
[14] ScienceSim: <http://www.sciencesim.com/> (accessed 29.04.11).
[15] C. Bouras, E. Giannaka, Th. Tsiatsos, A dynamic management scheme for DVEs, J. Network Comput. Appl. (JNCA), Elsevier Science 34 (1) (2011) 89–101.
[16] S. Rueda, P. Morillo, J.M. Orduna, A peer-to-peer platform for simulating distributed virtual environments, in: 13th International Conference on Parallel and Distributed Systems – Volume 2 (ICPADS'07), vol. 2, 2007, pp. 1–8.
[17] J. Sablatnig, S. Grottke, A. Köpke, J. Chen, R. Seiler, A. Wolisz, Adam-A testbed for distributed virtual environments, in: The 10th International Workshop on Multimedia Network Systems and Applications (MNSA-2008), Beijing, June 2008.
[18] A. Varga, The OMNet++ discrete event simulation system, In: Proceedings of the European Simulation Multiconference (ESM 2001), June 2001.
[19] G.C. Ewing, K. Pawlikowski, D. McNickle, Akaroa2: exploiting network computing by distributing stochastic simulation. In: Proc. European Simulation Multiconference ESM'99, International Society for Computer Simulation, June 1999, pp. 175–181.
[20] VAST, <http://www.vast.sourceforge.net/> (accessed 29.04.11).
[21] X. Zhuang, A. Bharambe, J. Pang, S. Seshan, Player Dynamics in Massively Multiplayer Online Games. Carnegie Mellon University School of Computer Science, October 2007, pp. 1–26. <http://reportsarchive.adm.cs.cmu.edu/anon/2007/CMU-CS-07-158.pdf>.