A Framework Model for DVEs using SIMUL8

Christos Bouras
Research Academic Computer Technology Institute and Computer Engineering and Informatics Department, Greece
N. Kazatzaki Str. GR-26500
+30 2610 960375
bouras@cti.gr

Eri Giannaka
Research Academic Computer Technology Institute and Computer Engineering and Informatics Department, Greece
N. Kazatzaki Str. GR-26500
+30 2610 960380
giannaka@cti.gr

Thrasyvoulos Tsiatsos
Research Academic Computer Technology Institute and Informatics Dept., Aristotle University of Thessaloniki, Greece
N. Kazatzaki Str. GR-26500
+30 2310 998990
tsiatsos@csd.auth.gr

ABSTRACT
Distributed Virtual Environment systems simulate the behaviour and activities of a great number of users interacting in a virtual world over a wide area network. The sizes of the virtual worlds and the tremendous number of users that DVEs are called to support require additional bandwidth and computational resources. For handling these growing requirements a lot of work has been done both to the direction of alternative architectural solutions as well as to techniques and algorithms for handling the limitations of these environments. For supporting large-scale DVEs, extended infrastructure is needed in terms of both hardware and software. However, both researchers and application designers do not always have access to such extended infrastructure and the assessment and evaluation of developed techniques becomes extremely difficult. To this direction, this paper presents a simulation modelling tool for networked servers DVEs that could be used by designers for simulating the performance of their approaches under different scenarios.

Categories and Subject Descriptors
I.6.3 [**Applications**], I.6.5 [**Model Development**], I.6.7 [**Simulation Support Systems**]: Environments

General Terms
Management, Measurement, Performance, Design, Experimentation.

Keywords
Distributed virtual environments, multiple networked servers architecture, performance evaluation, resource management, simulation applications.

1. INTRODUCTION
Distributed Virtual Environments (DVEs) have become a major trend in distributed applications [9]. Virtual environments simulate real or imaginary scenarios and are, nowadays, characterized by rich graphics and a wide range of integrated services. Many platforms, which adopt virtual reality technology, are implemented and DVEs are constantly enhanced with additional functionality, such as text and audio chat, streaming media support, application sharing, etc. These highly interactive systems simulate a virtual world, where multiple users share the same scenario and the same view of the virtual scene [9]. Each user is represented by an entity, called, avatar and has the ability to navigate within the virtual world and interact both with the other users of the system as well as with the system itself.

The advances in hardware and software technology, as well as the wide expansion of high-speed internet access allowed for the support of large-scale DVEs, which are currently used in various fields, such as learning and training, collaborative work, military applications and entertainment. The term large-scale is two-fold and refers both to the size of the virtual environment as well as to the tremendous number of users it is called to support [2]. For handling these demanding applications, existing approaches, from architectural point of view, fall usually into one of the following architectures: a) networked servers architectures and b) peer-to-peer architectures. To both directions there is lot of work done, including algorithms for the partitioning problem ([3], [10]), load balancing techniques [5], awareness methods [9] and other techniques ([13], [7]), while a great number of platforms has been designed and implemented ([1], [4], [8]).

As mentioned above, due to the great number of users that DVEs aim to support, the rich graphics and the need for a high level of realism, there is a constant trade-off between system performance and fault tolerance. The decision for the techniques and approaches used for dealing with this trade-off is usually related to the scope and the special characteristics of the scenario that each virtual world simulates. To this direction, this paper presents a framework model for networked servers DVEs, which can be used by application designers for selecting the appropriate model, algorithm and technique for the scenario they are called to simulate. The framework model proposed takes into account a number of generic parameters, which can be set on demand by the designers and stake-holders and it is based on "translating" system requirements to the concepts of operational management. The design and implementation of the framework is conducted with Simul8 [11], an integrated environment for working with simulation models, widely used in operational management.

SIMUTools 2009 March 2-6, Rome, Italy

The rest of the paper is structured as follows: Section 2 presents the motivation for the work presented as well as related work to the area of simulation tools for evaluating DVEs' performance, while Section 3 describes the main concepts, entities, processes and characteristics of generic Dynamic DVEs. Section 4 presents the distributed framework approach, in terms of the flow of events that take place and Section 5 presents the simulation framework in terms of its basic entities and their "mapping" to Simul8 objects. Section 6 presents the actual DVE simulation model, in terms of the issues and decisions implemented for incorporating all aspects of DVE systems as well as the parameters created for providing tailored solutions, according to the special needs and characteristics of DVEs. Section 7 presents an indicative experiment conducted for testing the model's accuracy and validity. Finally, Section 8 concludes the paper and 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 of the cases, the evaluation is based on theoretical models, which often fail to meet the circumstances and situations met in real DVEs. In particular, for supporting large-scale DVEs, extended infrastructure is needed in terms of both hardware and software. Due to the fact that both researchers and application designers do not always have access to such extended infrastructure, the assessment and evaluation of developed techniques becomes extremely difficult. In most of the cases, both application designers and researchers adopt specialized methods ([5]) for evaluating different techniques, while in other cases simulation tools have been developed from scratch [9]. However, given the fact that the design and implementation is application or technique-specific, the reusability of these tools for different architectures and algorithms is not always successful as either quantitative or qualitative parameters are not taken into account.

To this direction and for overcoming this important limitation, a simulation framework for assessing DVEs' performance is designed and implemented. The modelling framework takes into account a number of generic parameters, which can be set on demand by the DVE designers and stake-holders and it is based on transforming system requirements to the concepts of operational management. For being consistent to this objective, the design and implementation of the framework has been realized with Simul8 [11], an integrated environment for working with simulation models, mainly focused and widely used in operational management. One of the main difficulties that needed to be faced during the design and implementation process was the incorporation of detailed aspects of DVEs, such as neighbouring avatars concept, synchronization messages, computer resources and routing techniques, in the set of options provided by Simul8, which are more general and directed to classic operational management problems.

Having overcome this difficulty, the simulation tool implemented manages to incorporate all basic factors and parameters that affect system's performance, and through an easy and comprehensible interface allows for the evaluation of diverse techniques, easily tailored to the specific needs of each DVE. As mentioned above, the simulation tool could be used by designers of DVE systems for simulating the performance of their approaches under different scenarios and system setups.

3. DISTRIBUTED VIRTUAL ENVIRONMENTS

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

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 for creating an immersive experience. In DVEs the simulated world runs not on one computer system but on several, which are connected over a network, as presented in Figure 1. Users that connect to these systems are able to interact in real time, sharing the same view of the virtual world.

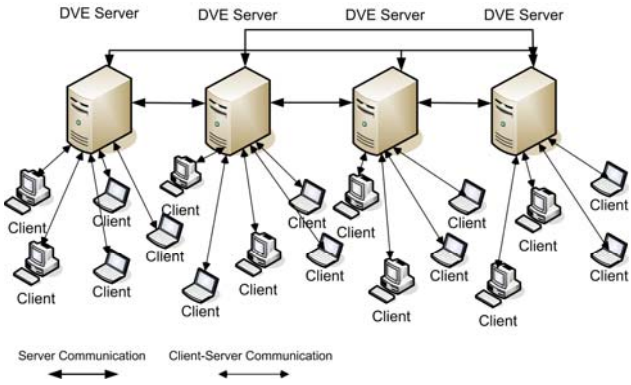

Figure 1: Multi-server DVE architecture

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 [6]. 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 each time 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, colour, etc), interact and communicate with other participating users. For achieving high-sense of realism and maintaining consistency, it is of critical importance that all connected users

are always aware both of the presence of other users as well as of any actions performed.

When a user connects to the DVE system, s/he is 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, who 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 sends those changes to the avatars concerned thus, modifying and synchronizing their view of the virtual world, as presented in Figure 2.

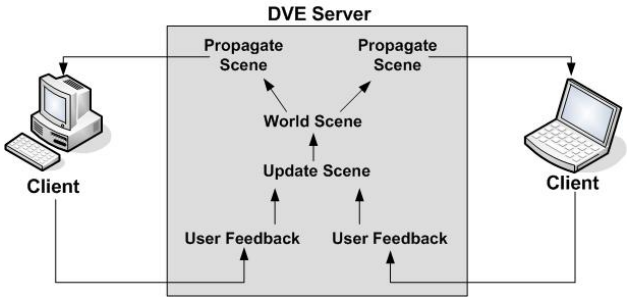

Figure 2: DVE client-server communication

4. DISTRIBUTED FRAMEWORK APPROACH

We consider a DVE system, which is comprised by a fixed number of servers (computers). Each server has a certain amount of resources and can serve requests/messages from the users. We consider as resources the capacity of the Central Processing Unit (CPU). At this point it should be mentioned that in real DVE systems, resources are also related to the network efficiency, in term of bandwidth availability. However, the network is a dynamically changing medium, whose state cannot be controlled by the application designer. Thus, throughout the simulation model, the approach presented encounters the CPU capacity as the main resource of the DVE system.

Each time that a user enters 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. For the monitoring task, the ConMan Server performs network management operations using the SNMP [12] protocol.

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. The processes that take place and their processing, based both on their frequency and resources they require, affect the servers' performance. Therefore, the performance of each server is constantly checked (after a fixed period of time) by the ConMan Server. However, for ensuring higher reliability, each server has a self-monitoring mechanism. In particular, each server runs an SNMP agent, which monitors the CPU usage. When a defined threshold is reached, the SNMP agent sends a "trap" to the ConMan Server, which notifies it that the specific machine reaches the point of saturation. When the trap message is received by the ConMan Server, the "partitioning" of the saturated server is initialized. The workflow process is presented in Figure 3.

In particular, when a "trap" is received the ConMan Server performs all the necessary actions for re-balancing existing workload among the servers of the system. It should be mentioned that the technique for workload re-balance is DVE specific.

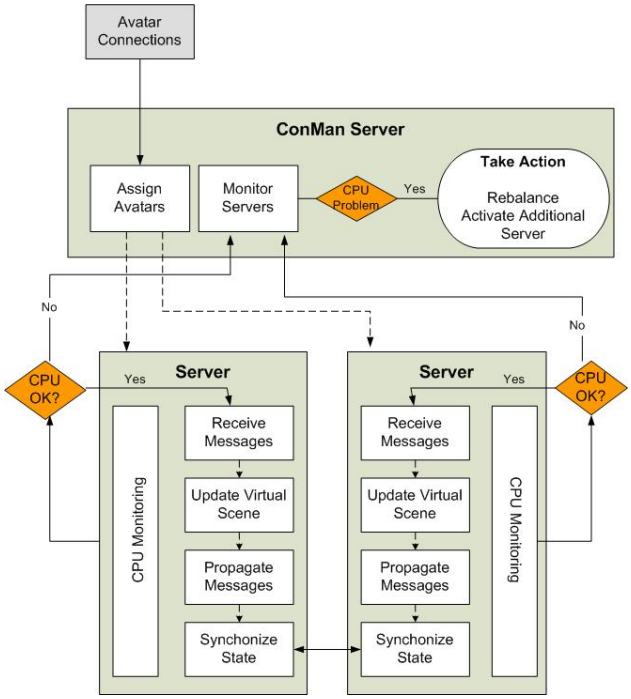

Figure 3: DVE Workflow

Based on the above, for creating a networked servers architectural framework we need to handle the following problem:

"Given a certain number of servers, with defined processing power, we need to find the optimum assignment of resources for serving as many users as possible with guaranteed efficiency, based on each application's special characteristics".

This problem falls into the area of operational management, where the efficiency is defined by the boundaries set for the CPU usage.

5. SIMULATION FRAMEWORK

As mentioned above, the approach presented faces the problem stated as an operational management one. To this direction, for testing the framework's efficiency and for studying the workflow process of the networked servers DVE, a discrete event simulation model is developed using SIMUL8 (version 12).

SIMUL8 [11] is an integrated environment for working with simulation models and its use is mainly focused to operational management problems. It has a powerful language and 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 to provide insights into the workflow process and to estimate the system performance measures. Figure 3 depicts a DVE system workflow, which is made of several interconnected simulation objects (input node, queues, and work centres). These objects as well as the simulation parameters are described in the sub-sections that follow.

5.1 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 Centres: A Work Centre is a place where work takes place on Work Items. Work done at work centres 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 centres 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 Centres. Work Items can be allowed to "expire" while in a storage bin.

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

5.2 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.

Table 1: Mapping of DVE entities to Simul8 objects

DVE Entity	Simul8 Objects
Avatar	Work Item
Virtual World Entry Point	Work Entry Point
Avatar Messages	Work Items
ConMan Server	Component
DVE Servers	Components
Inter-Server Messages	Work Items

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 the messages as well as the inter-arrival times between messages.

Messages: In the DVE simulation model there are three types of messages taken into account and are presented as Work Items. The first type is called "avatar", the second one is called "request" and the third one "synchronization". The avatar processes represent the actual avatars that enter the system, which means that for each avatar there is an avatar message initialized. The request process represents the messages sent by each avatar and are labelled with their parent id (that is the avatar id which sent the message).

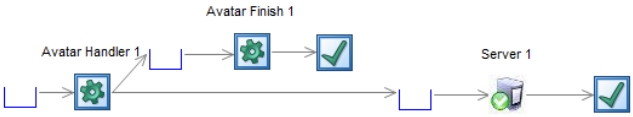

Figure 4: Detailed Server Representation

For simplification purposes, we consider that all messages sent by the users' avatar 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 for tailoring the messages exchanged by using additional labels attached to each message type. Finally, the synchronization messages represent the messages exchanged among the servers of the system for maintaining consistency and awareness. The synchronization messages among the servers take place when neighbouring avatars situated in different servers interact. As neighbouring avatars the simulation model defines the ones that their distance is equal or less than a defined value, which represents the Area of Interest (*AoI*). The *AoI* is an indication for whether two or more users can see each other.

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 and sets the life of each of these messages.

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 Figure 4. Each of these objects serves different tasks within the model. Work done at work centres usually takes up time and sometimes requires the availability of resources. For the DVE simulation, we consider that the time it takes for processing 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. The detailed description of the tasks that the Server Entity performs is presented in the section that follows.

6. SIMULATION MODEL

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

6.1 Simulation Parameters

This sub-section presents the main parameters used for setting up the performance and assessment results of the DVE system.

Number of Servers: this parameter defines the overall number of servers available, which could be used any time needed.

CPU_Usage(t): the actual CPU usage of the server at time *t.* This parameter is used for indicating the state of a server. For creating the dynamic framework we need to define a **maximum** and a **minimum** value for this parameter. The maximum value is used for indicating that a server tends to be overloaded and the SNMP "trap" is sent to the ConMan Server. The minimum CPU usage is a value which indicates that a server could be considered as under-used and its workload and tasks could be assigned to another already working server. At this point it should be mentioned that the proposed approach takes into account this parameter when a server is under this threshold for a specific period of time.

Server_Queue(t): the number of messages in the server's queue, which is an additional parameter for deciding whether the server reaches a saturation point. Like the CPU parameter, for the Server Queue, we need to define a maximum and a minimum value.

Routing technique: this parameter defines the way that workload will be balanced among the servers of the system. This parameter, as mentioned in a previous section, is strongly related to the partitioning and load balancing approach adopted by each type of application and should be carefully implemented for valid results.

System performance Check: this parameter defines the time interval that is used by the ConMan Server for checking the servers' status.

In the simulation model, the logical interactions between the different simulation objects (timing information, routing rules, etc) are handled using Visual Logic, SIMUL8's logic building environment. Custom dialogs are also developed to extend the model's specifications by easily defining the above-mentioned parameters, i.e. changing the number of servers, defining the values of thresholds, the properties of the servers' queues and displaying simulation results.

6.2 DVE Model Overview

The simulation model of the DVE system is presented in Figure 5. In particular, this figure presents the generic form of the DVE system, where both ConMan_Server and the other servers are combined into Component Type entities.

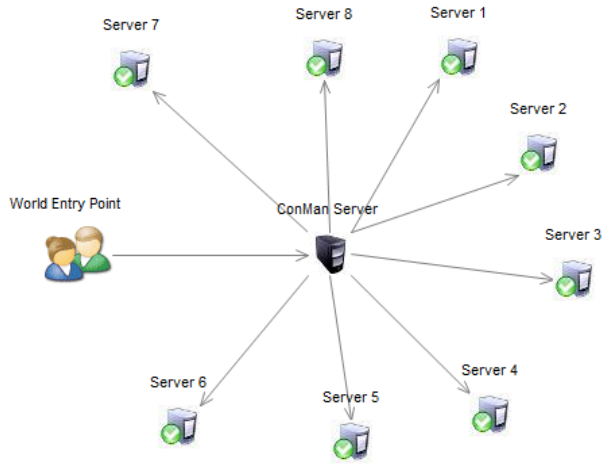

Figure 5: Generic DVE simulation model

The detailed representation of the DVE simulation model is presented in Figure 6. In this figure, the components are left open so that the sequence of events that take place through a DVE session will be clearer to the end user.

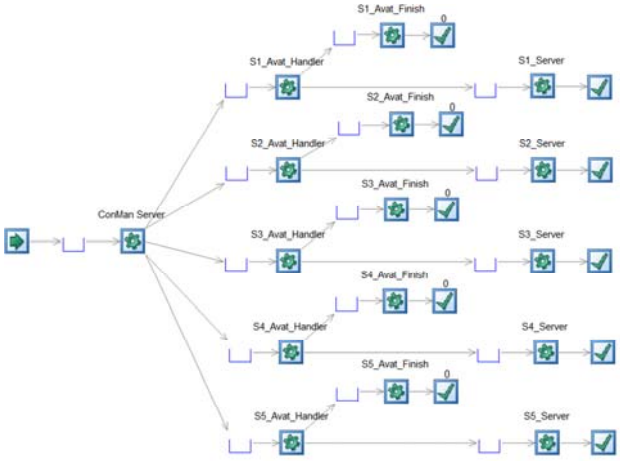

Figure 6: Detailed DVE simulation model

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, which, as mentioned above, labels each of these messages with a unique identifier 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. This attribute could be set according either to existing data for users' stay in the virtual world or approximate values. In the testing simulation performed, the avatar life was based on measures made for the World of Warcraft game [14] where the median user stay was about 50min.

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, which 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. In particular, the selection of the destination could be:

- Circular: avatars are forwarded in a circular way to the available servers of the system
- Equal Probability: avatars are forwarded with an equal probability profile to the servers of the system
- Spatial: in cases where each server manages certain part of the virtual world, avatars, according to their initial position would need to be forwarded to the appropriate server, which handled the corresponding partition.

All, above mentioned, decision techniques could be set to the simulation model according to each system's specifications. At this point it should be mentioned that in its current form, the modelling framework considers average values for most of the parameters taken into account. However, the impact of factors, such as the distance and network workload could be also added in the proposed modelling tool.

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 receive 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 centre 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 code, which, every a fixed time interval checks the number of avatars in the storage bin and generates a request for each of these avatars. For the testing simulations conducted, the time interval was set to 0.2 sec. 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 the testing simulation experiments the serving time of all servers available was set to $5x10^{-3}$ sec.

For monitoring servers' performance a utilization parameter was defined to the simulation model for each of the servers. This utilization parameter represents the CPU usage of each server and is calculated as follows:

$$U_i = \frac{B_i}{T},$$

where B_i is the busy time of server i over a T time interval. The busy time B_i is defined as follows:

$$B_i = req\_served \times s_i\_process\_time,$$

with req_served the number of requests served by server i and $s_i\_process\_time$ the time for processing each request.

The CPU is checked in the simulation model, through the Time Check Logic and Visual Logic, every a fixed time interval (which is set to 1 sec for testing the simulation) and in case that one or more of the servers have exceeded a threshold, the ConMan Server changes the routing scheme for balancing the workload among the participating servers.

7. ACCURACY AND RESULTS

This section presents an indicative simulation conducted for testing the DVE simulation model accuracy and validity. In particular, the scenario simulated was selected in such a way that its results could be easily proven.

7.1 Scenario Setup

For testing the simulation framework, we consider a DVE, which consists of five available servers. All of the servers have the same processing power. Avatars join the virtual environments following an exponential distribution with an average value of 10sec for the inter-arrival times. The average "avatar_life" is set to 50min and we consider that avatar perform actions, thus sending request messages to the servers they are connected, every 0.2 sec. Regarding the routing scheme that ConMan Server adopts for assigning users (avatars) to the servers, we consider a circular one (e.g. like cards be dealt from a pack). The maximum CPU threshold is defined to 80% while the minimum one is set to 10%. Finally, the duration of the experiment is set to 60 min, while the standard check for servers' CPU utilizations takes place every 1 sec. The values of the parameters taken into account are summarized in Table 2.

Table 2: Parameter values for the simulated scenario

Parameter	Value
Number of Servers	5
Servers' Processing Time	0.005 sec
Avatar's Entry Distribution	Exponential (10sec average)
Requests' Inter-arrival Time	0.2 sec

Routing Technique	Circular
CPU Time Check	1sec
Simulation Duration	60 min

7.2 Experimental Results

This section presents the results extracted from the simulation conducted with the DVE model. As described above, the simulation period was set to 60 min and avatars entered the virtual world following an exponential distribution with a 10sec average value. The avatar entry distribution is depicted in Figure 7. Totally, around 330 avatars joined the DVE system, but due to the fact that they had different lifetimes, not all of them needed to be concurrently served by the servers of the system.

One of the most critical aspects for the majority of the DVE systems and an important indicator for the performance is the CPU usage. To this direction the CPU usage percent over time was monitored for testing whether one or more of the servers exceeded the threshold defined. The results for the CPU utilization for all connected servers are presented in Figure 8. As it can be seen, all of the servers remain under the threshold for the simulation period, with a small exception of Server 2.

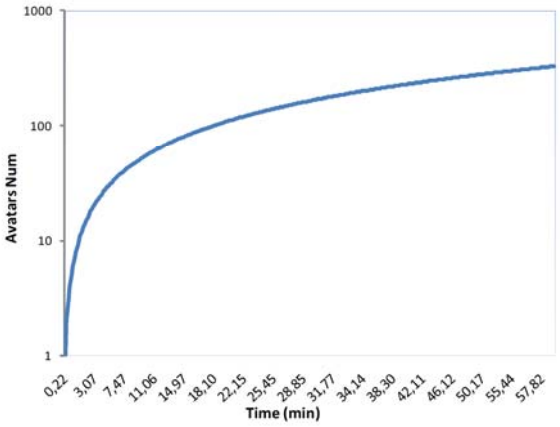

Figure 7: Avatar entry distribution

However, it should be mentioned that even though the threshold was exceeded, the value of 80% can ensure that the server is still away from reaching the saturation point of 100% in CPU utilization, which would affect seriously the overall DVE performance.

Another parameter measured was the number of concurrent users that each server could support, given their behavioural pattern in the system (which is related to the frequency of the requests generated by them).

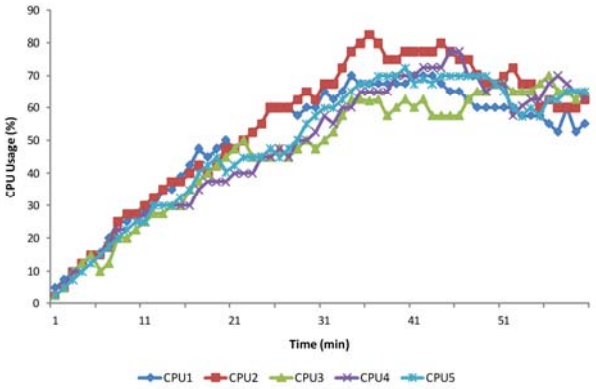

Figure 8: Servers CPU Usage (%)

The results of the experiment conducted for all servers are presented in Figure 9. In particular, this figure presents on the primary axis (left) the CPU usage over time for each of the servers in relation to the number of concurrent avatars in each of them, which is presented in the secondary axis (right). As it can be extracted by Figure 9, all servers behave similarly under certain workload introduced by connected users. Also it can be noticed that the system performs well (which means that the CPU is below the 80% threshold) for about 20-25 concurrent users, while the threshold is exceeded only in the case of more than 30 concurrent users.

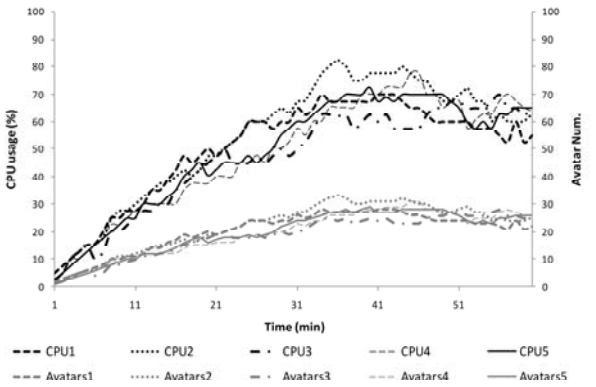

Figure 9: CPU Usage (%) related to the number of avatars over time (for all servers)

Given the similar behavior of all connected servers and the complexity of Figure 9, which might be difficult to be read, the results of the most workloaded server were exracted and are presented in Figure 10.

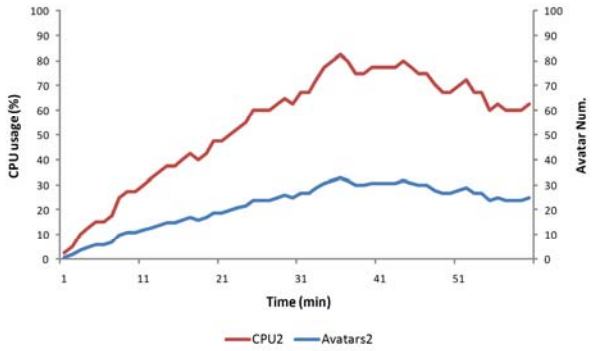

Figure 10: CPU Usage (%) related to the number of avatars over time (a representative example)

7.3 Discussion of the results

The results provided by the operation of the DVE simulation model correspond to what could be expected for the case of a DVE system with the parameters specified in the previous section.

In particular, given the fact that the processing power of the servers is higher from the requests generated by the users, the system is expected to function within the boundaries set. Also, given the capabilities of the servers, the number of concurrent users they can support is more or less expected.

8. CONCLUSION AND FUTURE WORK

This paper presented a framework model for networked servers DVEs, which was simulated using Simu8 simulation tool. The entities, processes and characteristics of generic 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, which is the resource management, could be formed into an operational management problem. To this direction, Simul8 was selected for designing the DVE simulation model, as it one of the most popular and widely used tools, both in the industrial and academic area of operational management.

Furthermore, given the fact that DVEs can widely vary on the scenarios they aim to support, 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 generalized framework, which could be easily tailored for meeting the specific needs of each DVE application. Therefore, the DVE simulation model presented in this paper 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.

To this direction, some of the planned next steps include the simulation and evaluation of various existing techniques, used widely for DVE systems, 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.

9. REFERENCES

[1] Anarchy Online: http://www.anarchy-online.com.

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

[3] C. Bouras, E. Giannaka, T. Tsiatsos, “Partitioning of Distributed Virtual Environments Based on Objects' Attributes”, 11th IEEE International Symposium on Distributed Simulation and Real Time Applications, Chania, Crete, Greece, , 22 - 24 October 2007, pp. 72 – 75

[4] Everquest: http://everquest.station.sony.com/.

[5] Jonh C.S. Lui, M.F. Chan, “An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems”, IEEE Trans. Parallel and Distributed Systems, Vol. 13, March 2002

[6] Joslin, Pandzic & Thalmann, “Trends in networked collaborative virtual environments”, Computer Communications, Volume 26, Number 5, 20 March 2003 , pp. 430-437

[7] Macedonia, Michael R., Zyda, Michael J., Pratt, David R., Brutzman, Donald P., Barham P. T. ”Exploiting Reality with Multicast Groups,” IEEE Computer Graphics & Applications, September 1995, pp.38-45.

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

[9] P. Morillo, S. Rueda, J.M. Orduna and J.Duato, “A Latency-Aware Partitioning Method for Distributed Virtual Environment Systems” , in IEEE Transactions on Parallel and Distributed Systems (TPDS), volume 18, number 9, pp. 1215-1226, September 2007. IEEE Computer Society Press, 2007

[10] P. Morillo, J.M. Orduña, M. Fernández and J. Duato, “Improving the performance of Distributed Virtual Environment Systems”, in IEEE Transactions on Parallel and Distributed Systems (TPDS), volume 16, number 7, pp. 637-649, July 2005. IEEE Computer Society Press, 2005.

[11] Simul8 Simulation Software: http:// www.simul8.com

[12] SNMP v2: http://tools.ietf.org/html/rfc1908

[13] T.A. Funkhouser, “Network Topologies for Scalable Multi-User Virtual Environments”, IEEE VRAIS ‘96, San Jose, CA, April, 1996.

[14] Zhuang, Xinyu, Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. “Player Dynamics in Massively Multiplayer Online Games.” Carnegie Mellon University School of Computer Science (Oct 2007): 1-26. <http://reportsarchive.adm.cs.cmu.edu/anon/2007/CMU-CS-07-158.pdf> (12 Jan. 2008).