Journal of Network and Computer Applications 34 (2011) 89–101

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

A dynamic management scheme for DVEs

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

[a] Computer Engineering and Informatics Dept., Univ. 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 Paiania, Attika, Greece
[d] Department of Informatics, Aristotle University of Thessaloniki, PO Box 114, GR-54124, Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 9 February 2010
Received in revised form
7 August 2010
Accepted 25 August 2010

Keywords:
Distributed virtual environments
Dynamic resource management
Load balancing
Networked servers' architecture
Communication cost
VR techniques and systems

ABSTRACT

Advances in computer technology and networking infrastructures in combination with advanced applications and services, have expanded the adoption of distributed virtual environments and promoted their use in a wide range of areas, such as learning and training, collaborative work, military applications and multiplayer online games. The characteristics and requirements of such DVEs differ significantly given the diverse objectives, scope and context that each virtual world aims at supporting. However, one common characteristic of DVEs is their dynamic state with users entering and leaving the system randomly, resulting in changes of the requirements for the DVE system. These changes require effective load distribution and management of the communication cost so that consistency is always maintained. This paper presents a dynamic management approach for DVEs driven by the diversity of different applications' characteristics and requirements. This approach exploits the dynamic nature of these systems for selecting and assigning, on an on-demand basis, the resources necessary for the efficient operation of the system. The experiments conducted to validate the behavior of the approach illustrate that it can significantly minimize communication cost among the system servers together with effective workload distribution.

1. Introduction

Distributed virtual environments (DVEs) simulate real or imaginary worlds by incorporating rich media and graphics. DVEs became more popular during the last decade, which is likely attributable to the wide expansion of high-speed internet access providing the basic support medium for these systems, as well as to the significant advances of both hardware and software. A large number of platforms and applications were designed and developed for supporting large-scale DVEs, which were gradually adopted in a wide range of both academic and industrial environments. However, the large number of users these systems aim to support in combination with the need for rich graphics and a high level of realism raise a constant trade-off between system performance and fault tolerance. The decision on the techniques and approaches used to deal with this trade-off is usually related to the objectives, scope and context that each virtual world aims at supporting, along with its special characteristics. In particular, the requirements may vary significantly among virtual worlds with diverse simulated scenarios. For example, in the 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. One common characteristic of DVEs is their dynamically changing state with users entering, navigating, interacting and leaving the system randomly (at will), resulting in continuously changing utilization of resources for the DVE system. These changes, in turn, call for effective load distribution and management of the inter-server communication cost so that consistency is always maintained and extended scalability is supported.

Research has focused on algorithms and techniques for load distribution as well as resource and communication management to improve the performance of these highly demanding systems. Recent research indicates that one of the main issues of networked servers DVEs is scalability. Morillo et al. (2005) presented that DVE systems reach saturation when any of the available servers reach 100% of CPU utilization which dramatically decreases overall system performance, while severely damaging awareness. On this basis, algorithms and techniques for performance optimization and scalability should focus on

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

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.08.013

making the system more resistant to continuous changing states over time. Based on that, it could be stated that for a system with a certain number of servers, the goal is to identify the optimal assignment of resources to serve as many users as possible, while minimizing the communication cost at the same time. This needs to be achieved with guaranteed efficiency based on each application's special requirements. To address this issue, this paper presents an approach for dynamic resource management and load distribution of networked servers DVEs, with the aim to extend their scalability and improve overall performance. More specifically, it proposes a method for optimizing the management of these environments by using the servers of the system on an on-demand basis, to limit the number of reassignments needed and to minimize unnecessary communication cost among the servers in order to reduce the effect of network latency. This dynamic exploitation of system's servers and resources constitutes the main novel contribution of this work. It is therefore advancing the state-of-the-art that is mainly addressing exploitation of all available system servers at any given time, without consideration of the actual and dynamically changing requirements of the virtual world. The behavior of the proposed dynamic management approach is evaluated through a series of experiments for different settings of the virtual world. The results of the experiments clearly show that the major contribution of the dynamic management scheme is the significant reduction of the inter-server communication cost. Given the fact that DVEs depend strongly on the underlying network characteristics, the reduction of the messages exchanged among the system's servers is of increased value for the viability, scalability and performance of the system. Furthermore, the dynamic management approach achieved balanced workload among the system's servers even in highly demanding cases, without reaching the saturation point of 100% of CPU utilization throughout the duration of the experiments.

The rest of the paper is structured as follows: Section 2 outlines some of the related work in the area of algorithms and techniques for load distribution and balancing in DVE systems, while Section 3 presents the dynamic management approach in terms of its main concepts, principles and the parameters measured. Section 4 presents the experiments conducted for evaluating the behavior and efficiency of the approach under different settings of the DVE system. Finally, Section 5 provides conclusions of the paper.

2. Related work

For handling DVEs and facing the scalability issue, existing approaches fall usually into one of the following architectures: (a) networked servers architectures, (b) peer-to-peer architectures and (c) server cluster architectures. Out of the three approaches, the server cluster can provide better latency guarantees, but remains the most expensive and it can also become a single point of failure (Chertov and Fahmy, 2006). Use of peer-to-peer approaches has increased interest in recent years. Even though peer-to-peer architectures seem effective on handling the scalability issue (Morillo et al., 2007), both latency and awareness issues still remain unresolved (Chertov and Fahmy, 2006). Server and network architectures developed to reduce the effect of network latency for DVEs approaches could be classified as (a) latency-driven distribution (LDD) and (b) resource-driven distribution (RDD) (Ta et al., 2006). Specifically, LDD approaches focus on the distribution of a DVE over the networking architecture, while the focus of the RDD approaches is on load distribution. The dynamic management approach presented in this paper refers to distributed networked servers architectures and is similar to the RDD concept. The below paragraphs summarize representative work and approaches relevant in this area.

The load distribution problem, which is related to the effective assignment of world entities, such as clients, cells and regions, to the servers of the DVE system has drawn increased research interest and a number of algorithms and methods have been proposed. In particular, in the LOT technique (Lui and Chan, 2002) authors have showed the key role of finding a good assignment of clients to servers and have proposed a three-step partitioning method for load balancing and communication refinement. Other approaches explored the use of micro-cells (Duong and Zhou, 2003; De Vleeschauwer et al., 2005) for load distribution in large-scale DVEs. Duong and Zhou (2003) used micro-cells to reassign servers in a cluster if the load on the server they are currently residing on becomes too large, while De Vleeschauwer et al. (2005) proposed the dynamic assignment of microcells to a set of servers to redistribute the load. However, the frequent remapping required with these methods could lead to high overhead for servers. Therefore, a locality aware load balancing method was proposed which reduces cross-server communication (Chen et al., 2005). However, even though it considers awareness of spatial locality in a virtual space, the method also leads to frequent region migrations. A new mechanism for sharing roles and separating service regions (SRSS) was proposed (Jang and Yoo, 2008), which reduced unnecessary partitions of short duration. Furthermore, an adaptive strategy that takes into account the non-linear behavior of DVE servers has been presented (Beatrice et al., 2002) with local scope for the load balancing technique. However, this strategy provided good performance only for uniform movement patterns of users. Following this work, Morillo et al. (2003) proposed an adaptive load balancing technique of global scope, which avoided DVE saturation as long as possible, regardless of both the movement patterns of users and the initial distribution of users in the virtual world. Other approaches adopt the use of thresholds for defining the number of clients a server is willing to serve (Chertov and Fahmy, 2006). In particular, each server uses a client threshold value to determine the number of clients it is willing to serve. If the client threshold is exceeded, the server attempts to migrate part of the load to a nearby server. Also, mirrored architectures have been proposed, which replicate DVE zones at multiple servers (Ta et al., 2006).

Regarding commercial DVE systems, the World of Warcraft (WoW) (World of Warcraft Architecture) adopts a distributed architectural model, where the world servers (or realms) constitute complete, self-contained copies of the game world. The realms are distributed among different parts of the world, but a world is a collection of servers, not a single server. When a user logs through his/her account, an authentication server verifies the data and transfers the user to the realm on which he/she last played. As far as it concerns Second Life (Kumar et al., 2008), users run a client program that connects to a central server, which employs four main clusters of machines, namely a central database, a logging database, an inventory database and a search database. The system uses visibility computation to determine the relevant subset of data to send to each client. Servers store all objects and perform the key actions to evolve the world while maintaining reasonable consistency. Second Life divides the world into 256×256 m^2 tiles, each of which is statically bound to a particular server that executes on a single CPU core. To minimize communication, Second Life partitions objects in the virtual world among the servers so that at any given time, only one server maintains an object's state.

Research in the direction of load distribution and resource management for DVEs has been very active with numerous approaches presented and adopted. The majority of these

approaches exploit all available system servers at any given time, despite the actual and dynamically changing requirements and demands of the system, independent of whether they perform load distribution focusing on clients, regions or cells. The work presented in this paper builds on previous research and is based on the dynamic nature of DVE systems and their continuously changing needs. More specifically, it proposes a method for optimizing the management of these environments by using the servers of the system on an on-demand basis for limiting the number of reassignments needed and for minimizing unnecessary communication cost among the servers. In this direction, the approach adopts the concept of performance thresholds to decide both the activation and deactivation of system's servers, based on the current requirements of the DVE. In particular, the work presented describes the way that dynamic management can be achieved in DVE systems for achieving efficient load distribution and minimization of the communication cost, rather than presenting new techniques and algorithms for the actual load balancing and the control of the network latency.

3. Dynamic management of DVE systems

A virtual environment can be considered as a simulation of either an imaginary or real world generated by a computer. In networked servers DVEs, the simulated world does not run on one computer system but on several that are connected over a network. Connected users view the virtual world on their computer (client), thus having their own local copy of the virtual environment. In the majority of existing DVE systems, users have the ability to navigate in the virtual world (i.e., changing their position coordinates), interacting with the objects of the virtual environment (i.e., changing some of their attributes such as location, shape, color, etc.), as well as interacting and communicating with other participating users. Achieving a high-sense of realism and maintaining consistency among all users' views is of critical importance so that all connected users are always aware both of the presence of other entities (either users or virtual objects) as well as of any actions performed.

When a user connects to the DVE system, they are assigned to one of the available servers. This assignment follows the algorithms and techniques that each DVE adopts for handling resource allocation and for achieving load distribution among the servers. Throughout the users' presence in the system, the server that is responsible accepts the messages produced by all users that it handles, processes these messages and updates the state of the virtual world accordingly. Then, it distributes these changes to the users concerned, thus modifying and synchronizing their view of the virtual world. The majority of existing techniques uses all servers available for handling and supporting the DVE system. In this paper we introduce the term *dynamic* for DVEs where the number of servers used changes and adapts to the resources needed over time. This section focuses on the specification of the dynamic management approach, specifically its main concepts, the workflow of the processes and events employed as well as the parameters used.

3.1. Issues in DVEs

As mentioned above, DVEs are characterized by their 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 that inconsistencies and performance degradation are avoided. The partitioning scheme that each DVE system adopts could assist in preventing frequent redistributions. Therefore, partitioning techniques have a dual objective related to: (a) the effective distribution of workload for the optimum resource utilization and (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.

In this direction, a quality function has been introduced by Lui and Chan (2002) for measuring the effectiveness of each partitioning scheme. This approach is focused on achieving equally balanced workload among the servers, which is meaningful only for homogenous systems. In particular, 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. In other words, the workload among the servers might be equally balanced, but the workload may be easy to handle for one server, while it may be an overload for another server. Equally balanced workload is most effective in cases of homogenous systems, in which all machines available are of the same type and with the same capabilities.

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. 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. In cases where all available servers are used, regardless of the techniques adopted, the communication cost could be significantly increased. In cases where multiple servers are available with a small number of connected users, the servers would still need to exchange communication messages for synchronization purposes and for keeping the DVE state updated, thus overloading the total cost of the system.

Given these challenges, load distribution and communication cost comprise two of the most critical factors for the performance of DVE systems that should be accounted for in resource management schemes.

3.2. Proposed dynamic DVE approach

The dynamic management approach presented in this paper utilizes performance thresholds, as proposed by Morillo et al. (2005) that are different for each server based on its processing capabilities and resources in order to address both homogeneous and heterogeneous systems. In particular, the dynamic scheme defines certain boundaries within which an application can operate efficiently. Given that the efficiency of each application depends on the focus of each simulated scenario (a virtual classroom has different awareness requirements from a virtual battlefield), the dynamic management approach considers that

the thresholds defined are scenario-driven and are related to the tolerance of the system in terms of realism and performance.

Furthermore, for addressing the communication cost that can be of great importance due to the native and non-predictable delays of the network, the proposed approach adopts the concept of *necessary* resources for maintaining the consistency of the system. In particular, the resources of the system (which are the servers available) are allocated according to the demands/requests generated by connected users. Thus, in the case of a system with numerous servers and a small number of users, the approach uses only the number of servers necessary to support the connected users and keeps other servers deactivated (idle). In this way, communication messages are only exchanged among the activated servers and the communication overhead is significantly reduced.

We conclude that 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 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.

3.3. Specification of dynamic management scheme

We consider a DVE system comprising a fixed number of servers. Each server has a certain amount of resources and can serve requests/messages from the users. We consider the processing capability of the central processing unit (CPU) as a resource. Each time a user enters the DVE system, a request is sent to a central server, which is denoted as *ConMan* Server (Connection Manager Server). This server performs two main tasks: (a) accepts and authorizes the connection requests from the users and redirects them to the appropriate server and (b) monitors the performance of the working (active) servers and intervenes whenever one or more servers need to be managed. For the monitoring task, the servers of the system perform network management operations using the Simple Network Management Protocol (SNMP) (SNMP v2). The *ConMan* Server communicates with the users only during the connection phase and for the assignment of the user to the appropriate server.

When users enter the virtual world, they start to navigate, interact and perform actions since they are already assigned to a server, thus sending messages/requests to the servers. Each of the DVE system's servers 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 events that take place and their processing, based on their amount and the resources they require, affect the servers' performance. Therefore, each server has a self-monitoring mechanism for the constant check of its state. In particular, each server runs an SNMP agent, which monitors the CPU usage. When the defined threshold/boundary is reached, the SNMP agent sends a "trap" message to the *ConMan* Server, which notifies it that the specific machine reaches the threshold. When the trap message is received by the *ConMan* Server, the unloading process of the saturated server is initialized. More specifically, when a "trap" is received, the *ConMan* Server performs all the necessary actions for redistributing existing workload among the servers of the system. The technique for workload redistribution is DVE specific and some of the most commonly used techniques are described in the section that follows.

Upon the receipt of the trap message, *ConMan* Server first checks among the list of available servers to trace the idle ones, if any. The first idle server traced is activated and workload is balanced among the two servers. If all available servers are active, *ConMan* performs a check for tracing candidate servers for undertaking part of the nearly saturated server. An active server is considered able to handle additional workload when it has enough resources available. For defining this ability of an active server, the approach sets another threshold, denoted as *CPU_capable*. When the utilization of a server (CPU usage) falls below this threshold and stays there for a defined period of time, the server is considered a candidate for undertaking additional workload.

As mentioned above, users enter and leave the environment randomly in a DVE system. When the number of users is decreased, or in cases that the connected users are not very active (low request rate), then the CPU utilization of one or more of the connected servers could significantly decrease. However, despite the fact that the server is underused, its active state leads to unnecessary communication cost for the system, mainly for synchronization purposes. This underused state of a server is triggered by the proposed approach by setting another threshold, denoted as minimum CPU threshold (*CPU_min*). Thus, when the utilization of a server falls below this threshold and keeps this value for a defined time interval, the *ConMan* Server performs a process for deactivating this server. The deactivation of the server can be completed if and only if at least one of the other active servers is able to process the additional workload that will be assigned to it by the underused one, which means that at least one of the active servers' CPU utilization must be below the *CPU_capable* threshold. The dynamic management scheme is presented in pseudo-code in Table 1.

The definition and selection of multiple thresholds is adopted in the dynamic management method in order to assure that all requirements are satisfied as well as for avoiding unnecessary changes in the state of the DVE system. In particular, both activation and deactivation of servers take place if and only if other available servers exist or are able to receive additional workload (through the *CPU_capable* threshold).

In some cases, misleading signals of activation or deactivation of servers can be generated when for example one of the servers reaches the maximum or minimum threshold instantaneously. The proposed approach addresses this issue by defining a certain time interval for which either the overloaded or underused server transmits signals of *CPU_max* and *CPU_min* thresholds before the SNMP trap is sent, as presented in the pseudo-code of Table 2.

3.4. Parameters

This sub-section presents and summarizes the main parameters used for the implementation of the dynamic management scheme on the DVE system. These parameters, some of which have already been mentioned in the previous sub-section, are the following: (a) *CPU_Utilization*, (b) *CPU_max,* (c) *CPU_capable*, (d) *CPU_min*, (e) resource assignment technique and (f) server performance check interval.

Table 1
Dynamic management scheme in pseudo-code.

```
/* The ConMan server "listens" for events from connected servers. These events
are sent when either overloading or under-usage occurs. In such cases rebalancing
is needed.*/
EVENT: "SNMP Trap Message" is received from Server[i]
    IF trap_msg[i] signals CPU_max
   /* The trap messages denotes that a server is overloaded */
        SET Server[i] AS Overloaded
        /*Search among available servers for an idle one */
        FOR all available servers in the system excluding Server[i]
             /*Get each server's state. The state can be either IDLE OR ACTIVE */
            CALL: getServerState RETURNING ServerState
            /*If an idle server is found, it is activated and the rebalancing
            process begins*/
            IF ServerState IS IDLE
                CALL: ACTIVATE(server)
                START: RESOURCE_ASSIGNMENT_TECHNIQUE()
                /*Server is found and rebalancing is called.*/
                BREAK IF Statement
                /*Loop is terminated*/
                EXIT FOR LOOP
                EXIT
            END IF Statement
        END FOR LOOP
        /*If no idle server is found ConMan server checks among the active servers
        for one that can handle the workload. A server can handle additional
        workload if and only if its CPU utilization is below the CPU_capable
        threshold*/
        FOR all available servers excluding Server(i)
            CALL getServerCPU RETURNING ServerCPU
            /*If one of the active servers can handle the load rebalancing is
            called*/
            IF ServerCPU <= CPU_capable
                START: RESOURCE_ASSIGNMENT_TECHNIQUE()
                /*The loop is terminated*/
                BREAK IF Statement
                EXIT FOR Loop
                /*The process is terminated*/
                EXIT
            END IF
        END FOR LOOP
    ELSE IF trap_msg[i] signals CPU_min
   /* The trap messages denotes that a server is under-used */
        SET server[i] AS Underused
        /*Search among available servers for one that can handle the workload.
        A server can handle additional workload if and only if its CPU utilization
        is below the CPU_capable threshold*/
        FOR all available servers in the system excluding Server(i)
            CALL getServerCPU RETURNING ServerCPU
            /*If one of the active servers can handle the load rebalancing is
            called*/
                IF ServerCPU <= CPU_capable
                    START: RESOURCE_ASSIGNMENT_TECHNIQUE
                    CALL: DEACTIVATE(Server[i])
                    BREAK IF Statement
                    EXIT FOR Loop
                    /*The process is terminated*/
                    EXIT
                END IF
        END FOR
    END IF
```

CPU_Utilization(t): the actual CPU usage of the server at time *t*. This parameter is used for indicating the state of a server.

CPU_max: the maximum CPU utilization value used for indicating that a server tends to be overloaded. Whenever this value of utilization is exceeded SNMP "trap" is sent to the *ConMan* Server.

CPU_capable: the CPU utilization value that, when reached, it indicates that an already active server can accept additional workload from another either nearly saturated or underused server.

CPU_min: the CPU utilization value that, when reached, it indicates that a server could be considered as under-used and its workload and tasks could be assigned to another already working/active server.

Resource assignment technique: this parameter defines the way that workload could be re-distributed among the servers of the system. This parameter, as mentioned in the 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. Some of the most common resource assignment techniques implemented in DVEs are the following:

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

Table 2
Self-monitoring and trap mechanism at the server side.

```
/*Server self monitoring and trap mechanism*/
/*Monitor CPU Utilization */
CALL getServerCPU RETURNING ServerCPU
IF ServerCPU >= CPU_max
/*If utilization reaches the maximum threshold start the timer to ensure
that the utilization is increasing*/
    START Time Counter
    /*Monitor CPU usage to check whether it remains above the CPU_max threshold*/
    CALL: getServerCPU RETURNING ServerCPU
    IF Time Counter Has Not Expired
        IF ServerCPU >= CPU_max
            CALL: PrepareTrap()
        ELSE
       /*CPU increasse was instant*/
            RESET Time Counter
        ENDIF Statement
    ELSE
        SEND Trap Message signaling CPU_max
    ENDIF Statement
ELSE IF ServerCPU <= CPU_min
/*If utilization reaches the minimum threshold start the timer to ensure
that the utilization is decreasing*/
    START Time Counter
    CALL getServerCPU RETURNING ServerCPU
    IF Time Counter Has Not Expired
        IF ServerCPU <= CPU_max
            CALL: PrepareTrap()
        ELSE
        /*CPU decrease was instant*/
            RESET Time Counter
        ENDIF Statement
    ELSE
    /*Server is overloaded and a trap message needs to be sent to ConMan Server*/
        SEND Trap Message signaling CPU_min
    ENDIF Statement
ENDIF Statement
```

Server performance check interval: this parameter specifies the time interval used by the servers' self-monitoring mechanism for deciding the transmission of a trap message. This time interval should be carefully selected to ensure that (a) the system monitoring mechanism does not lose critical signals of server's saturation, and (b) the monitoring mechanism will not create an additional load to the system by transmitting misleading traps to the system.

The above mentioned parameters have been used for the performance evaluation that is presented below to demonstrate the behavior of the dynamic management approach for different DVE setups.

4. Performance evaluation

This section describes the experiments conducted for assessing and validating the dynamic management approach for DVEs under different setups of the virtual environment.

The results demonstrate that the dynamic adjustment and allocation of the system's resources to the users' requirements and requests as they are formed over time presented better results compared to the approaches that use all available servers. The main improvement was reduced communication cost in all cases examined. Furthermore, the dynamic management approach achieved balanced workload among the system's servers in terms of CPU usage even in highly resource demanding cases, without ever reaching the prohibitive saturation point of 100% of CPU utilization for any individual server over the duration of the experiments.

The experiments were conducted with the STEADiVE (Simulation Tool for Evaluating and Assessing Distributed Virtual Environments) tool (Bouras et al., 2009), which is a simulation tool implemented with Simul8 (Simul8 Simulation Software). The STEADiVE tool can be used by designers of DVE systems for simulating the performance of their approaches under different scenarios. For the dynamic management approach, two series of experiments were conducted, each aimed at evaluating different aspects of the approach. The first series of experiments deals with the behavior of the dynamic management scheme under different user-driven metrics, while the second series monitors the behavior of the approach under different resource assignment techniques, commonly used for partitioning purposes. The descriptions of the experiments, the scenario setup, and the results for each of the experiments are presented in the subsections that follow.

4.1. Experiment series one-behavior under different user-driven metrics

The first series of experiments were conducted to evaluate the behavior of the dynamic management scheme under different user-driven metrics. These metrics are totally dependent on users' actions, preferences and behavior and cannot therefore be a priori known to the system. The user-driven parameters taken into account are:

- The rate that users join the virtual world that is denoted as inter-arrival time and corresponds to the time in between two consecutive user arrivals. The inter-arrival time is a useful metric used to describe how frequently users enter the virtual worlds, and affects the performance based on the overhead incurred both in terms of workload and communication cost.
- Users' activity profile: by activity profile we refer to the rate of movements and interactions that users perform in a virtual

environment over time. The higher the incoming and activity rate the higher the user requests and the system's resources needed.
- Users' lifetime that is defined as the time that each user spends in the virtual environment, also denoted as session length.

For examining and comparing the behavior of the dynamic management approach under the above mentioned user-driven metrics, the dynamic management scheme is compared to the linear one (Lui and Chan, 2002), which is proven to provide good results for large-scale DVEs. In short, the linear approach follows a three-step process by first distributing the virtual environment into the available servers of the system using a Divide and Conquer technique. In the step that follows for a defined time interval, an algorithm checks the workload on the servers and performs all the necessary re-assignments of entities so that a nearly equally balanced workload is achieved. The third and last step of the approach encounters the exchange of some entities among the servers for the refinement of the communication cost.

To measure the communication cost in the experiments conducted, some of the users that entered the virtual world are labeled as 'border users'. For these users, the number of messages sent, should be communicated to the appropriate servers of the system so that consistency can be maintained. As mentioned earlier, the calculation of the communication cost is defined by the number and size of the messages exchanged among the servers of the system. For simulation purposes, we consider an average message size of x units throughout the experiments conducted. Thus, the communication cost in all experiments and figures presented reflects the number of messages exchanged among the interconnected servers.

Furthermore, another factor to be taken into account in DVEs is the latency of the system to users' requests. As in DVEs latency cannot be measured properly, the round-trip delay (RTT) or ASR (Average Response Time) is used instead (Morillo et al., 2007). For all scenarios examined the results obtained for the average ASR are compared to the 250 ms ASR threshold used in the literature.

4.1.1. Scenarios' setup

The experiments conducted consider a DVE system comprising 8 servers. All of the servers available have the same computing power and are dedicated in serving the DVE system requests (no background applications are running). 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 (Zhuang et al., 2007). In particular, for simulating the inter-arrival times, an exponential distribution has been used with three different cases of mean inter-arrival times.

The relaxation on the exponential distribution stems from the fact that the experiments did not take into account "the time of day" parameter for user's entrance to the virtual world. For the activity profile, we considered three different values for the request rates, while for simulating the user lifetime in the system, users are assigned with a mean value of 40 min lifetime. The parameters and the corresponding values for the two approaches, dynamic and linear one, are presented in Table 3.

The selection of different values for the user-driven metrics formulate different virtual worlds' setups and scenarios tested. Specifically, experiments were conducted for user mean inter-arrival times of 9, 6 and 3 s and with user request rate of 0.5, 0.2 and 0.1 s for each of the inter-arrival times. For each of the scenarios presented, approximately 1000 runs were executed and the results correspond to the average values obtained from these runs.

Table 3
Series one-general parameters.

Dynamic approach parameters	
CPU max threshold	80%
CPU capable threshold	40%
CPU min threshold	10%
CPU check time (SNMP)	1 s
Routing technique	Spatial
Linear approach parameters	
Rebalancing time	5 min
Routing technique	Circular
Common parameters	
Servers number	8
Mean user lifetime	40 min
Experiments' duration	180 min (3 h)
User's incoming distribution	Exponential

Table 4
Servers' CPU usage under different virtual world setups.

Scenarios		S1	S2	S3	S4	S5	S6	S7	S8
Dynamic approach									
9s	**0.5**	40							
	0.2	52	48						
	0.1	55	46	42	59				
6s	**0.5**	60							
	0.2	38	39.5	36	36				
	0.1	41	39	39	35	31.5	30	44	40
3s	**0.5**	62	54.8						
	0.2	42	36.5	35.5	36	30.5	38.5	37.5	35
	0.1	85	74	71	69	73	71	71	71
Linear approach									
9s	**0.5**	5.6	4.4	5.6	5	5.2	4.8	5	5.4
	0.2	14	11	14	12	12.5	12	12.5	13.5
	0.1	28	25	28	26	21	23	27	26
6s	**0.5**	6.8	7.2	8	6.9	7.4	7.4	7.6	9
	0.2	17	18.5	21	18	17.5	19	18.5	21
	0.1	33	36	37	39	39	38	37	43
3s	**0.5**	17	14.2	13.2	15	14	11.6	17	15.2
	0.2	41	36.5	33.5	36	37	29	43	37.5
	0.1	75	71	74	72	77	71	75	72

Even though each scenario tested corresponds to one of the possible combinations of users' inter-arrival times and activity profile, only three representative scenarios are presented based on common characteristics of the results and limited space of this paper. The three scenarios include a best case scenario (optimistic) where both inter-arrival times and activity profile are less frequent (9 s inter-arrival times and 0.5 s for inter-request rate), an average case scenario (6 s inter-arrival times and 0.2 s for inter-request rate) and a worst-case scenario in terms of resources needed (3 s inter-arrival times and 0.1 s for inter-request rate). These different scenarios could also be defined as the requirements of different types of DVE systems, in terms of the actions that users perform within the virtual world. For example, the demanding scenario could simulate the actions and interactions in a battlefield while the optimistic scenario could simulate users' actions in a virtual classroom.

4.1.2. Results

The results of the CPU utilization of all connected servers for all cases examined are presented in Table 4. Columns 1 and 2 (under the heading "Scenarios") of the table present the users' inter-arrival time and requests' rate, respectively, while the rest of the columns present the CPU utilization for each server *Si* ($i=1$–8) of

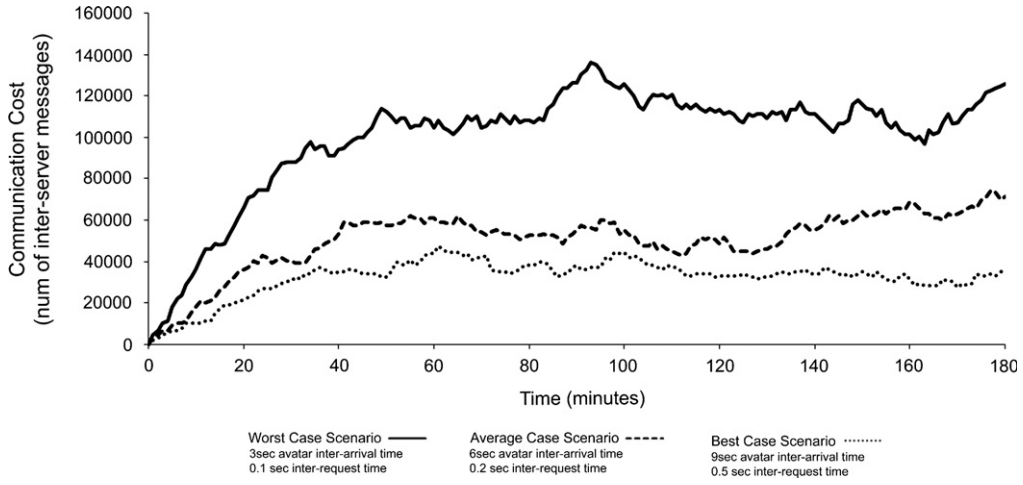

Fig. 1. Communication cost over time for the linear approach.

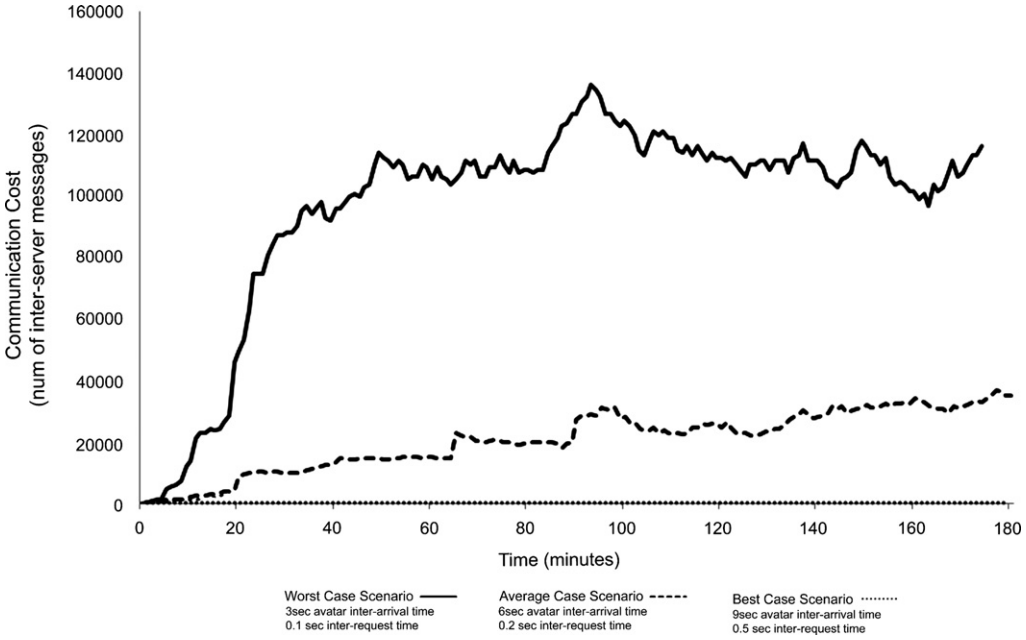

Fig. 2. Communication cost over time for the dynamic approach.

the DVE system. The cells of the table that are left blank indicate that the specific server has not been activated.

For the less demanding scenarios (9 s inter-arrival times) the linear approach, which uses all the servers available, presents very low CPU utilization for all connected servers (Table 4). Furthermore, we observed that the dynamic approach utilizes all of the system's servers only in the cases of the highly demanding scenarios (3 s user inter-arrival times with high request rate 0.1 s). However, even in this case, the dynamic approach achieves a relatively balanced workload among servers. Even though in one of the cases the 80% threshold was exceeded, it did not lead to saturation of the system. On the other hand, the linear approach for the cases of the optimistic and the average scenarios presents low CPU utilization for system servers thus making them under-used.

Figs. 1 and 2 present the communication cost for the linear and the dynamic approach, respectively, for the most demanding (3 s user inter-arrival time and 0.1 s requests inter-arrival time), the average (6 s user inter-arrival time and 0.2 s requests inter-arrival time) and the best case (9 s user inter-arrival time and 0.5 s requests inter-arrival time) scenario.

When comparing Figs. 1 and 2, using the dynamic approach reduced communication cost among participating servers in all experiments compared to the linear approach. This reduction is mainly achieved in the cases of the optimistic (best case) and average scenarios. For the best case scenario, the communication cost is close to zero as one server handles the workload over the experiment's duration.

The reduction of the communication cost can be explained by the fact that in most of the cases for the dynamic management approach the number of servers that need to be updated and synchronized, is smaller, as depicted in Fig. 3.

In addition, for the dynamic approach presented, when a new server needs to be activated, the communication cost that border users introduce concerns only the origin server (that is the server that reaches the *CPU_max* threshold) and the destination server

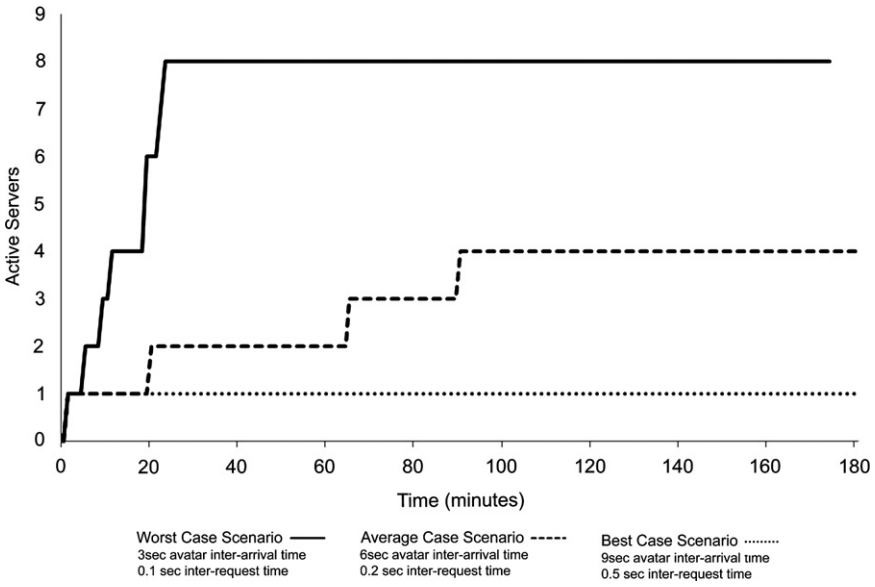

Fig. 3. Number of active servers over time for the dynamic approach.

Table 5
Average ASRs for the linear and the dynamic approach.

Scenarios		Average ASR (in ms)	
		Dynamic approach	**Linear approach**
9s	**0.5**	80	121
	0.2	127	130
	0.1	212	222
6s	**0.5**	90	142
	0.2	214	236
	0.1	243	244
3s	**0.5**	132	198
	0.2	244	245
	0.1	252	258

(that is the server that undertakes workload from the origin server). Thus, even for the most demanding case, which is the one that users enter the virtual environment every 3 s and send requests every 0.1 s, the communication cost of the dynamic approach is lower than the linear approach, even though the number of servers used (active) is the same.

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

Based on the work presented by Henderson and Bhatti (2003), if the ASR is not greater than 250 ms, then users perceive that the system responds quickly. The results obtained from our experiments (Table 5) illustrate that for the scenarios examined both approaches achieve an average ASR below the 250 ms threshold. The only case when both approaches result in the average ASR exceeding this threshold was for the most demanding case (3 s user inter-arrival times with high request rate 0.1 s). However, even in this case the values are not prohibitive. Furthermore, the results show that in most cases, the dynamic approach achieves better results compared to the linear one. It should be noted that the difference in the average ASRs of the two approaches is more obvious in cases where fewer servers are used. This observation 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 neighbor avatars of a given avatar *i* are assigned to the same server as *i*. When the neighbor 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.

These experiments considered a generic DVE, where issues as users' distribution in the virtual space, users' moving pattern and the partitioning method are not specific. As the average ASR can be addressed by means of the partitioning method, the application of specific algorithms in real DVE systems, i.e. GRASP (Morillo et al., 2007), could be applied to achieve better results.

4.1.3. Summarizing the results

The results of the experiments suggest that both approaches behave well, in terms of CPU utilization, in all scenarios examined as none of them reached 100% of CPU usage, which would downgrade the overall system's performance. The major advantage of the dynamic approach over the linear is mainly manifested for cases of low and average incoming rate and activity profile. In such cases, the linear approach presents significantly greater communication cost among the participating servers, when compared to the dynamic one. Furthermore, even for the demanding scenarios, where the two approaches seem to converge in terms of the resources needed, the dynamic approach achieves lower communication cost among the system's servers as well as balanced workload distribution, thus addressing both performance factors. Regarding the system's response time to users' requests, both approaches achieve average ASR times within the performance boundaries set in the literature. However, once again, the dynamic approach seems to provide better ASR values for the cases of low and average incoming rate and activity profile compared to a linear approach. The better ASR is likely due to the use of fewer system servers which results in a higher concentration of neighbor avatars to the same server.

4.2. Experiment series two-behavior under different resource assignment policies

The second set of experiments mainly focuses on evaluating the behavior and performance of the dynamic management approach against different policies for resource assignment. The

policies selected for this series of experiments are described in the following paragraphs.

Linear Classic: This approach is the one presented by Lui and Chan (2002), as described previously. In this approach all servers are assigned with users and load balancing takes place every fixed period of time. The routing technique used for assigning incoming users to the servers of the system adopts the circular approach, while the 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 the dynamic approach for the activation of the servers. In particular, 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.

Equally Balanced Dynamic: In this approach the dynamic scheme is fully adopted for the assignment of workload to the servers of the system. As presented extensively in a previous section, 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

Table 6
Series two-algorithmic approaches.

Algorithmic approach	Resource assignment technique	Rebalancing	Rebalancing triggering
Linear Classic	Circular	All servers	Time
Linear Dynamic	Equal probability	All servers	CPU threshold
Equally Balanced Dynamic	Equal probability	Overloaded server	CPU threshold
Spatial Dynamic	Spatial directed probability	Overloaded server	CPU threshold

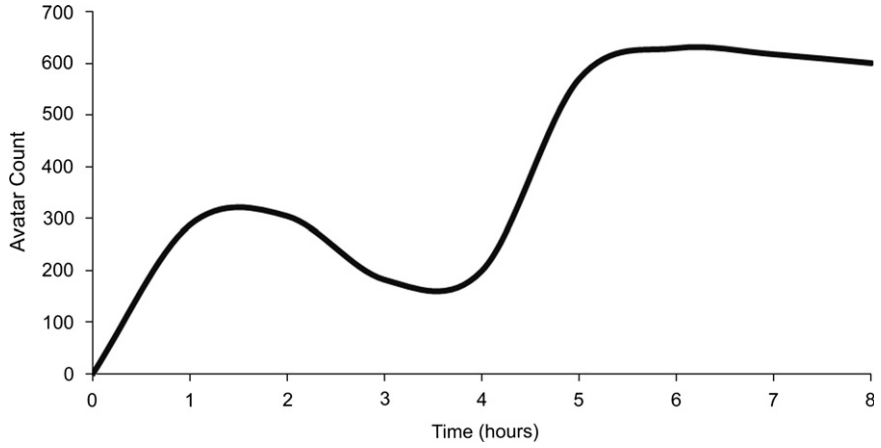

Fig. 4. User presence over time.

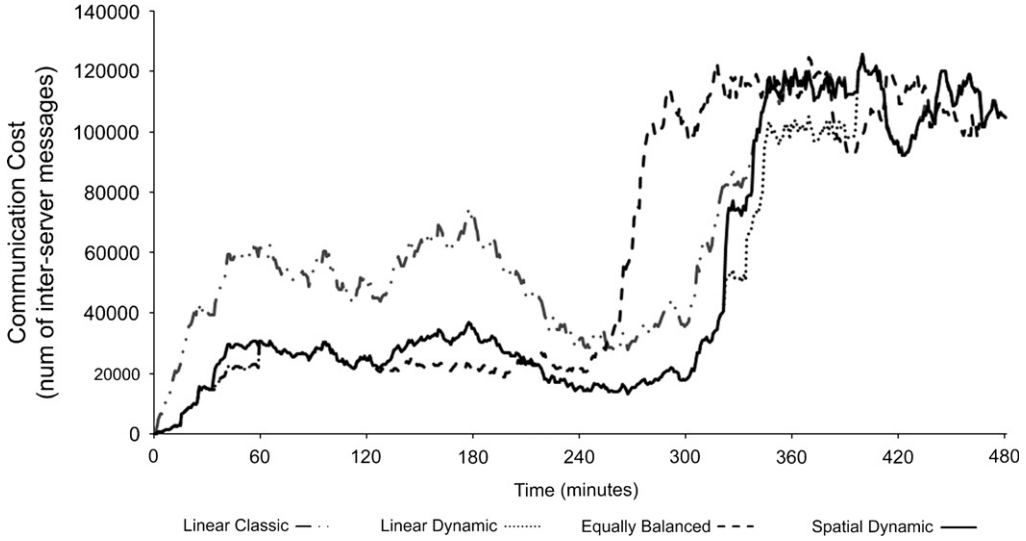

Fig. 5. Communication cost over time for the different approaches.

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. In particular, each connected server is assigned with a certain area of the virtual world 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.

The characteristics of the algorithmic approaches are summarized in Table 6.

4.2.1. Scenarios' setup

The experiments conducted consider a DVE system comprised again by 8 servers. All the available servers have the same computing power and are dedicated to serving the DVE system's requests. The experiments' time was set to 8 h and the users' request rate was set to an average of 0.2 s. Finally, a Poisson distribution was selected for simulating different users' incoming pattern over time, in order to demonstrate "time-of-day" related behavior. Fig. 4 presents the average values for the presence of users over time for the experiments conducted. For each of the scenarios presented, approximately 1000 runs where executed and the results presented below correspond to the average values obtained from these runs.

4.2.2. Results

Fig. 5 presents the communication cost over time for the four approaches examined. We observed that the communication cost was much greater for the Linear Classic Approach where all servers are used compared to the other three.

Likewise, the communication cost is much greater for the first hours, where the presence of users is lower than the one presented over the last three hours. More specifically, for the last hours, where the presence and incoming rate of users is higher, all approaches converge for communication cost. This can be explained by the fact that in the case of multiple running servers, even though there is no actual need, require a greater number of messages to be exchanged among servers for maintaining consistency.

Fig. 6 presents the number of active servers over time. The number of servers required over time converges for the Linear Dynamic, Equal Balanced and Spatial Dynamic approaches. Therefore, we conclude that the dynamic scheme provides similar results, regardless of the resource assignment technique used by each DVE.

4.2.3. Summarizing the results

The results obtained through the second series of experiments with the four approaches mentioned above indicate that the system performs well in terms of CPU usage in all scenarios. None of the servers reached the detrimental threshold of 100%. However, the communication cost was much greater for the Linear Classic approach compared to the other three approaches because all available servers are used for the Linear Classic approach.

4.3. Discussion

As we observed from the two series of experiments conducted, the major contribution of the dynamic management scheme is the significant reduction of the communication cost. Given that DVEs are strongly dependent on the underlying network characteristics (i.e. 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.

Combining and analyzing the results of the two series of experiments, the best results in terms of CPU utilization, load balancing and communication cost were achieved by the Linear Dynamic and Spatial Dynamic approach for all cases examined.

The major difference between these Linear Dynamic and Spatial Dynamic approaches is that when rebalancing is needed for the Linear Dynamic approach, then all active servers are affected, whereas only the overloaded and the newly assigned server are affected in the rebalancing process for the Spatial Dynamic approach (Fig. 7). In real DVE systems, the interventions and updates on the state of the connected servers should be minimized in order to avoid possible impacts on the connected users' experience and smoothness of the overall system's operation.

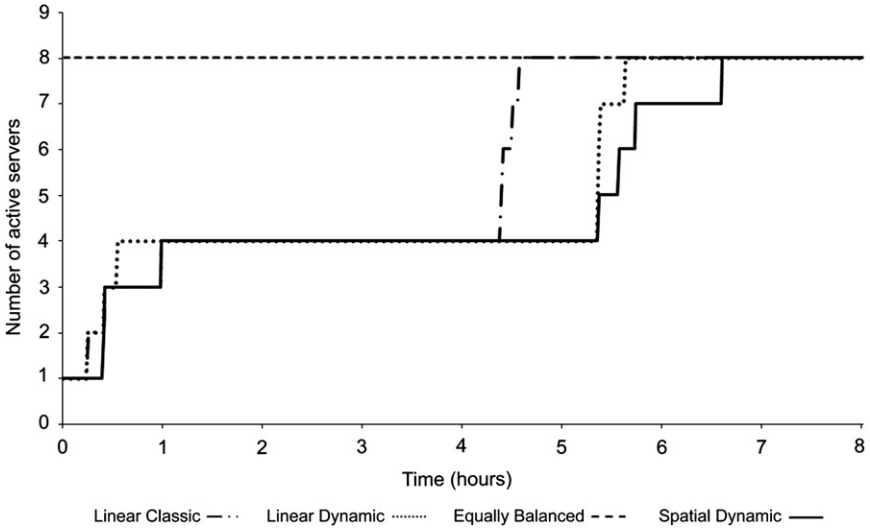

Fig. 6. Number of active servers over time for the dynamic approach.

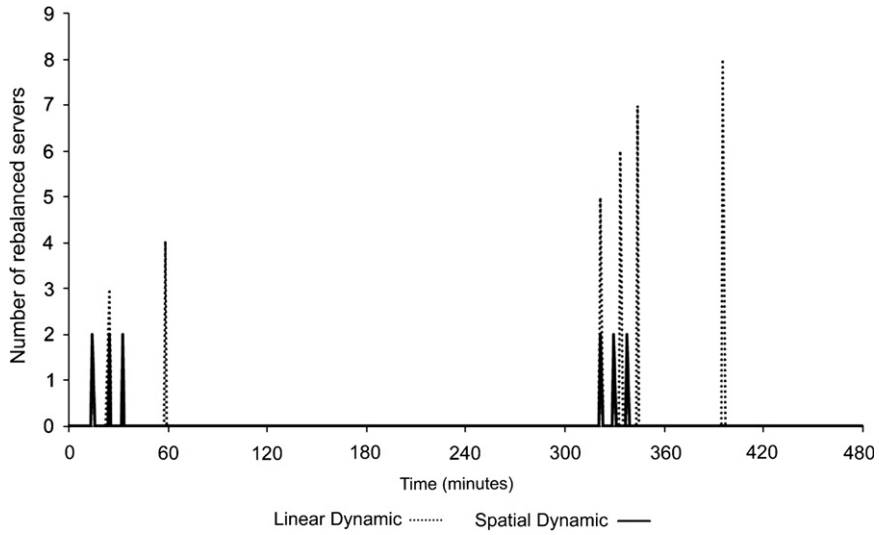

Fig. 7. Number of affected servers during rebalancing.

Table 7
Rebalancing iterations and reassignments over time.

Algorithmic approach	Rebalancing iterations	Average number of reassigned users	Average number of total reassignments over time
Linear Classic	96 times (every 5 min over a period of 8 h)	19	970
Linear Dynamic	7 times	62	430
Equally Balanced Dynamic	7 times	44	310
Spatial Dynamic	7 times	36	260

As a result, another parameter to be taken into account is the overhead of the rebalancing process. As presented in the second series of experiments, the rebalancing takes place every 5 min with the Linear Classic approach, while rebalancing is triggered if and only if one or more of the active servers becomes overloaded or under-used in all other dynamic approaches (Linear Dynamic, Equally Balanced Dynamic and Spatial Dynamic). The Linear Classic approach had significantly lower average number of reassigned users when compared to the dynamic approaches. However, the total number of reassignments for this Linear Classic approach is more than double the number for the dynamic approaches for the 8 h duration of the experiments. The dynamic approaches only perform 7 iterations, one each time that a server is overloaded. From the dynamic approaches examined, the Linear Dynamic approach performs the most reassignments (both per iteration and in total), while the Spatial Dynamic approach performs the least. The high number of reassignments per rebalancing for the Linear Dynamic approach is due to the fact that each rebalancing process affects all activated servers. With the other two dynamic approaches (the Equally Balanced Dynamic and the Spatial Dynamic), rebalancing only affects two servers, the overloaded one and the one that will be activated Table 7.

It should be noted that, throughout the series of experiments conducted with the simulator and presented in this paper, the performance of the overall DVE system for the dynamic approaches monitored did not produce severe delays apart from some instant latency peaks during rebalancing in the Linear Dynamic approach. Also it should be mentioned that the CPU thresholds that is adopted both for triggering and allowing the rebalancing process ensure that there will not be severe rebalancing delays throughout the dynamic management scheme.

Therefore, we conclude that the dynamic management scheme presented in the paper achieves the best results in all criteria tested when it is combined with a spatial resource assignment technique. As far as it concerns the CPU threshold and the way their values are set, it could be mentioned that the CPU threshold is strongly related to the processing capacity of the available servers. In particular, for really powerful machines, which can process simultaneously the workload introduced by a high number of users, the threshold can be set closer to the maximum value of 100%.

5. Conclusions

This paper presented a dynamic management scheme for DVE systems, which exploits the nature of these demanding applications for optimal resource management and extended scalability support. The basic concept of the approach lies in finding an optimal resource assignment, which is driven from the application's requirements as they change and formulate over time.

For validating and illustrating the effectiveness of the proposed dynamic approach, experiments were conducted to compare a number of alternative approaches for load management in DVEs. The experiments were conducted under various settings of the virtual world with different values for users' behavior and trends as well as with different resource assignment approaches. The results showed that the dynamic management scheme, apart from balanced workload distribution, achieves a significant reduction of the communication cost, which is of vital importance for DVE systems, while maintaining average response time to users' requests below the threshold used in the literature. Furthermore, users' behavior, in terms of incoming rate and

activity profile, play a critical role for the workload introduced to the DVE system. Given that this behavior cannot be a priori known, the dynamic adjustment of the system's resources to the users' requirements presents better results, especially for cases of limited user presence and participation. Furthermore, regarding the CPU usage, the dynamic management approach achieved balanced workload, close to the ideal, even in highly demanding cases, without reaching the saturation point of 100% of CPU utilization for any of the system's servers. Finally, through the comparative experiments conducted for different algorithmic approaches, it can be concluded that the dynamic management approach when combined with a spatial distribution approach, provides the optimum results. More specifically, CPU does not reach the detrimental threshold of 100% utilization, the workload is balanced among servers, the communication cost is significantly lower compared to other approaches, average response time to users' requests is below the 250 ms threshold and rebalancing affects only the overloaded and the newly assigned server. These results clearly indicate the benefit of the proposed approach in the DVE management space.

References

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

Bouras C, Giannaka E, Tsiatsos TA. Framework model for DVEs using SIMUL8. In: Proceedings of the second international conference on simulation tools and techniques. Rome, Italy, March 2009.

Chen J, Wu B, Delap M, Knutsson B, Lu H, Amza C. Locality aware dynamic load management for massively multiplayer games. In: Proceedings of the tenth ACM SIGPLAN symposium on principles and practice of parallel programming, Chicago, IL, June 2005.

Chertov R, Fahmy S. Optimistic load balancing in a distributed virtual environment. In: Proceedings of the 2006 international workshop on network and operating systems support for digital audio and video, Newport, Rhode Island, November 2006.

De Vleeschauwer B, Van Den Bossche B, Verdickt T, De Turck F, Dhoedt B, Demeester P. Dynamic microcell assignment for massively multiplayer online gaming. In: Proceedings of fourth ACM SIGCOMM workshop on network and system support for games, Hawthorne, NY, October 2005.

Duong T, Zhou S. A dynamic load sharing algorithm for massively multiplayer online games. In: Proceedings of the 11th IEEE international conference on networks, October 2003.

Henderson T, Bhatti S. Networked games: a QoS-sensitive application for QoS-insensitive users. In: Proceedings of the ACM international conference on applications, technologies, architectures, and protocols for computer communication. (SIGCOMM '03). 2003. pp. 141–147.

Jang SM, Yoo SJ. An efficient load balancing mechanism in distributed virtual environments. ETRI Journal 2008;30(4):618–20.

Kumar S, Chhugani J, Kim C, Kim D, Nguyen A, Dubey P, et al. Second life and the new generation of virtual worlds. Computer 2008;41(9):46–53.

Lui JC, Chan MF. An efficient partitioning algorithm for distributed virtual environment systems. IEEE Transactions on Parallel and Distributed Systems 2002;13(3):193–211.

Morillo P, Orduna M, Fernandez M, Duato J. An adaptive load balancing technique for distributed virtual environment systems. In: Proceedings of the 15th IASTED international, 2003.

Morillo P, Orduna M, Fernandez M. Improving the performance of distributed virtual environment systems. IEEE Transactions on Parallel and Distributed Systems 2005;16:637–49.

Morillo P, Rueda S, Orduna M, Duato JA. Latency-aware partitioning method for distributed virtual environment systems. IEEE Transactions on Parallel and Distributed Systems 2007;18:1215–226.

Ta DN, Zhou S, Shen HS. Greedy algorithms for client assignment in large-scale distributed virtual environments. In: Proceedings of the 20th workshop on principles of advanced and distributed simulation, workshop on parallel and distributed simulation. IEEE Computer Society, Washington, DC, May 2006.

World of Warcraft Architecture: 〈http://electronics.howstuffworks.com/world-of-warcraft5.htm〉. Simul8 Simulation Software: 〈http://www.simul8.com〉, SNMP v2: 〈http://tools.ietf.org/html/rfc1908〉.

Zhuang X, Bharambe A, Pang J, Seshan S. Player dynamics in massively multiplayer online games. Carnegie Mellon University School of Computer Science; 2007 pp. 1–26.