A Latency-Aware Partitioning Method for Distributed Virtual Environment Systems

Pedro Morillo, Silvia Rueda, Juan Manuel Orduña, and José Duato, *Member*, *IEEE*

Abstract—Distributed virtual environment (DVE) systems allow multiple users working on different client computers interconnected through different networks to interact in a shared virtual world. In these systems, latency is crucial for providing an acceptable quality of service (QoS), since it determines how fast client computers are reported about changes in the shared virtual scene produced by other client computers. This paper presents in a unified manner a partitioning approach for providing a latency below a threshold to the maximum number of users as possible in DVE systems. This partitioning approach searches the assignment of avatars, which represents the best trade-off among system latency, system throughput, and partitioning efficiency when solving the partitioning problem. Evaluation results show that the proposed approach not only maximizes system throughput, but also allows the system to satisfy, if possible, any specific latency requirement needed for providing QoS. This improvement is achieved without decreasing either image resolution or quality of animation, and it can be used together with other techniques already proposed. Therefore, it can contribute to provide QoS in DVEs.

Index Terms—Distributed applications, distributed/network graphics.

1 Introduction

Distributed virtual environments (DVEs) have become a major trend in distributed applications, mainly due to the enormous popularity of multiplayer online games (MOGs) in the entertainment industry. These highly interactive systems simulate a virtual world where multiple users share the same scenario. The system renders images of the virtual world that each user would see if he was located at that point in the virtual environment. Each user is represented in the shared virtual environment by an entity called *avatar*, whose state is controlled by the user through the client computer. Hundreds and even thousands of client computers can be simultaneously connected to the DVE system through different networks and even through the Internet. DVE systems are currently used in many different applications [1] such as civil and military distributed training [2], collaborative design [3], and e-learning [4]. Nevertheless, the most extended example of DVE systems are commercial MOG environments. These systems use the same simulation techniques that DVE systems do [5], and they are predicted to make up over 25 percent of the local area network (LAN) traffic by the year 2010 [6]

Although centralized server (client-server) architectures or peer-to-peer (P2P) architectures were also proposed for DVE systems [1], architectures based on networked servers are becoming a *de facto* standard for DVE systems [7], [8], [9], [10], [11]. The reason for this trend is that, on one hand, architectures based on a single centralized server are not scalable with the number of connected clients (particularly for the most extended application of DVE systems—MOGs). On the other hand, DVE systems based on P2P architectures seem to be scalable enough, but they must still efficiently solve the *awareness problem*. This problem consists of ensuring that each avatar is aware of all the avatars in its neighborhood [12]. Providing awareness to all the avatars is a necessary condition to provide time-space consistency (as defined in [5], [13], [14], and [15]). However, it is not a sufficient condition. Even when using an awareness method that determines at each moment which other avatars an avatar must exchange messages with, time-space inconsistencies can arise among different avatars because of clock drifts and/or network delays [13]. Awareness is crucial for MOGs, since, otherwise, abnormal situations could happen. For example, a user provided with a noncoherent view of the virtual world could be shooting something that he can see, although it is not actually there. Also, it could happen that an avatar not provided with a coherent view is killed by another avatar that it cannot see. In networked-server architectures, the awareness problem is easily solved by the existing servers, since they know the location of all avatars all the time. Each avatar reports about its movements (by sending a message) to the server where it is assigned to, and all the servers periodically synchronize their state to maintain a coherent model of the virtual world. Thus, each server can easily decide which avatars should be the destinations of the messages sent by a given avatar by using a criterion of distance. There is no need for clients to determine the neighborhood of avatars, since servers can easily do this task. The efficient solving of the awareness problem is one of the main reasons that explain the small number of servers in networked-server architectures (the higher the number of servers, the higher the complexity of

• *P. Morillo, S. Rueda, and J.M. Orduña are with the Departamento de Informática, Universidad de Valencia, Av. Vicent Andrés Estellés, s/n, 46100 - Burjassot (Valencia), Spain.*
E-mail: {Pedro.Morillo, Silvia.Rueda, Juan.Orduna}@uv.es.
• *J. Duato is with DISCA, Universidad Politécnica de Valencia, Camino de Vera, s/n., 46022 Valencia, Spain. E-mail: jduato@gap.upv.es.*

Manuscript received 1 Mar. 2005; revised 3 May 2006; accepted 31 Oct. 2006; published online 9 Jan. 2007.
Recommended for acceptance by X. Zhang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0194-0305.
Digital Object Identifier no. 10.1109/TPDS.2007.1055.

the synchronization technique). Also, it is one of the reasons for the widespread use of these architectures, particularly in MOGs. This paper focuses on networked-server architectures, since MOGs represent the vast majority of current DVE systems.

In networked-server architectures, servers must contain the current status of different 3D models, perform positional updates of avatars, and transfer control information among different clients. Thus, each new avatar represents an increase in both the computational requirements of the application and the amount of network traffic. When the number of connected clients increases, the number of updating messages must be limited in order to avoid a message outburst. In this sense, different approaches have been proposed in order to limit the number of surrounding avatars that a given avatar must communicate with. At this point, it is worth mentioning that avatars are only the representation of users in the virtual world, and they are not responsible for sending or receiving messages. However, for the sake of shortness, we will also use the term avatar to denote the client program running on the client computer controlling an avatar. Concepts like areas of influence (AOIs) [1], [16], locales [17], or auras [18] define a neighborhood area for avatars in such a way that a given avatar must notify its movements (by sending a message) only to those avatars located in its neighborhood. Thus, the AOI of a given avatar determines the amount of network traffic generated by that avatar. Other approaches use three-tiered architectures [19], [20], data filtering [21], or distributed cache management [22] in order to minimize the impact of network traffic on the performance of the DVE system.

Users in distributed systems usually perceive system performance either as the system latency or as the system response time [23], [24], [25], [26], since the critical factor for them is that the system works in "real time." Otherwise, user productivity rapidly decreases [26]. Latency can be described as the time interval required by the system in order to process each information unit. For example, latency is defined in interconnection networks as the time required by the network to carry a message from the origin network interface to the destination network interface [24]. If the distributed system is interactive, as DVE systems are, then latency is measured as the response time of the system. The response time can be defined as the time required by the system to provide an answer to each user request. Nevertheless, users in a DVE system perceive system performance not only as system latency, but also as system throughput. System throughput can be defined as the maximum number of users that the system can simultaneously support. For example, users in a multiplayer game environment want not only for the system to quickly respond to their movements, but also for it to allow them to simultaneously play with the greatest number of players (users) as possible. In order to design actually efficient DVE systems, this special feature must be taken into account. In a previous paper, we showed that DVE systems are kept away from saturation if all the servers are kept below 100 percent of CPU utilization. Using this result, we also designed a partitioning method (based on a load-balancing technique) that maximizes throughput of DVE systems [27]. In this paper, we go one step further, focusing on both system throughput and system latency. Latency can be defined in a DVE system as the time interval from the instant when any neighbor avatar of a given avatar i makes a movement until the instant when avatar i (the client computer controlling i) is notified of that movement.

On the other hand, the term quality of service (QoS) can be applied to any system, meaning that the system not only supports a given client but it also fulfills some specific requirements (potentially in different dimensions) of that client, regardless of eventual system bottlenecks. For example, in network environments, the term QoS means the ability of some networks to conform with some specific user requirements of latency, jitter delays, traffic peaks, and so forth regardless of the current network traffic [28]. System bottlenecks arise because the system is designed with a given capacity in its resources, and the workload that the system supports (in computers, the amount of information to be processed in a given period of time) eventually exceeds this capacity. In DVE systems, latency is crucial for providing QoS to users, since it determines how fast users perceive changes in the virtual world, thus limiting the responsiveness and interactivity of the system. Therefore, a system cannot provide an acceptable QoS to a client if a maximum latency threshold is exceeded. That is, providing a latency below this maximum threshold to clients is a necessary condition for providing QoS.

This paper presents, in a unified manner, a partitioning method that not only maximizes system throughput, but also allows the system to satisfy, if possible, any specific latency threshold that can be required to provide QoS [29]. We will show that the problem of providing a latency below a given threshold to the maximum number of avatars as possible can be tackled by means of the partitioning method, regardless of the threshold value. The idea is to use the aggregated CPU bandwidth of the servers first to avoid system saturation. After that, if the workload generated by the clients is below this aggregated bandwidth, then this approach uses the remaining CPU bandwidth to group in the same server that those avatars surrounded with the greatest amount of neighbors, thus decreasing the latency provided to the maximum number of avatars. In order to do so, this approach looks for the assignment of avatars, which represents the best trade-off among system latency, system throughput, and partitioning efficiency when solving the partitioning problem. We have denoted this problem as the QoS problem to denote that solving this problem provides an assignment of avatars that also takes care of QoS. Since the QoS problem shows high complexity, we propose a partitioning method based on a heuristic search. Therefore, we also present a new implementation and comparative study of several different heuristic methods applied to solve the QoS problem [30], [31]. Finally, we present the performance evaluation of the proposed partitioning method. Evaluation results show that the proposed method can maximize system throughput while providing a latency below a threshold to the highest number of avatars as possible, therefore contributing to provide QoS.

The rest of the paper is organized as follows: Section 2 describes the problem of providing QoS to avatars in DVE systems and the existing proposals. This section also defines the QoS problem and presents the proposed method for solving it. Section 3 presents the implementation, tuning, and comparative study of different heuristic search methods described in the literature when applied to the QoS problem in DVE systems. Next, Section 4 presents the performance evaluation of the proposed method. Finally, Section 5 presents some concluding remarks and future work to be done.

2 The QoS Problem in DVE Systems

The problem of providing QoS to clients in a DVE system has been already described, and some strategies have been proposed [32], [33], [34]. One of these approaches [33] uses latency compensating methods in order to repair the effects of high network jitter. A different approach consists of modifying the resolution of the 3D models depending on the connection speed of the client computer [32]. Both strategies try to provide QoS to avatars in exchange for reducing the quality of graphics (either image resolution [32] or quality of animation [33]). However, none of these strategies takes into account the nonlinear behavior of DVE systems with the number of avatars assigned to each server, as described in [27]. Therefore, they do not care about guaranteeing a maximum latency to clients (they cannot guarantee that the system will not become saturated, greatly degrading the latency provided to all clients regardless of the connection speed or the network jitter). Finally, the other approach consists of a scheduling algorithm for DVE systems based on a client-server architecture. This algorithm minimizes the overall client error produced when the server is saturated, and it can only send a subset of the update messages to clients [34]. However, this approach is for DVE systems based on a centralized server, where avatars cannot be migrated (reassigned to a different server). In the case of migration, this scheduling algorithm could lead to starvation, or it would have to be modified. As these approaches show, the problem of providing QoS in DVE systems shows multiple dimensions (network latency, jitter, consistency, graphical resolution, quality of animation, and so forth). These approaches try to provide good performance in a given dimension (sometimes at the cost of decreasing the performance in another dimension) in spite of potential system bottlenecks (connection speed, network jitter, and number of clients connected to the system).

We propose an approach that improves the performance in a given dimension without affecting any other dimension. Additionally, this approach can be used together with any of the approaches described above. Concretely, we propose a latency-aware partitioning method for providing QoS, since a necessary condition for providing QoS in DVE systems is to keep latency below a maximum threshold value. However, latency cannot be properly measured in distributed systems, and the round-trip delay (RTT) for the messages sent by an avatar i is used instead. A recent work shows that, if this RTT is not greater than 250 ms, then users perceive that the system responds quickly [35]. Therefore,

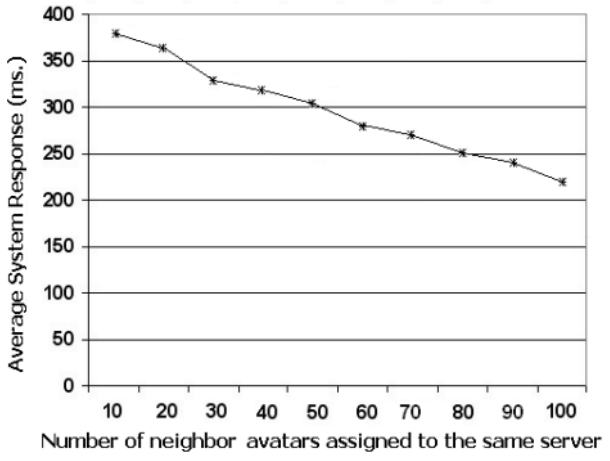

Fig. 1. Average message latencies for an avatar with 100 neighbor avatars in its AOI.

we will consider that a DVE system can offer QoS to a given client only if the average RTT for the messages sent by this client (denoted as ASR, for *Average System Response* [27]) is lower than 250 ms. Nevertheless, this threshold value can be modified as necessary, depending on the application requirements. For the sake of shortness, in the rest of the paper, we will use the expression "provide QoS to an avatar" to denote the fact that the ASR provided to the client computer controlling that avatar is below 250 ms.

The RTT for the messages sent by a given avatar is closely related to the partitioning problem, since the average RTT is an interpolation of the maximum RTT (client-server plus server-server delays) and the minimum RTT (only client-server delays). Thus, if a given avatar i and all its neighbors are assigned to the same server, then the time required for reporting avatar i of the movements of any neighbor avatar will be smaller than if some of these neighbors are assigned to a different server. However, the impact of eliminating the server-to-server delays has not been measured until now. We have evaluated different DVE systems by simulation (as described in Section 4) in order to measure the effect that assigning neighbor avatars to different servers can have on the latency provided to a given avatar. This evaluation has been performed on many different scenarios (different DVE configurations, different initial distributions of avatars in the virtual world, different movement pattern of avatars, different avatar populations, and so forth). A significant example of this evaluation is shown in Fig. 1. This figure shows on the y-axis the average RTTs obtained for messages sent by a given avatar i. The x-axis shows the number of neighbor avatars of i assigned to the same server that i is assigned to. The rest of the neighbor avatars were randomly assigned to other servers. Each point in the plot represents the average value of the average latencies obtained after 30 different simulations, and the standard deviation of the different simulations was not higher than 25 ms in any case.

Fig. 1 shows that the average RTT for messages sent by avatar i linearly decreases as a higher number of neighbor avatars of i are assigned to the same server as i. The reason for that behavior is that, 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. Therefore, the average RTT is lower. Concretely, we can see that the system provides QoS (average RTT under 250 ms) to avatar i if 80 or more neighbor avatars of avatar i (from a total of 100 neighbor avatars) are assigned to the same server as i.

The evaluation results obtained for other avatars (with different AOI sizes) in other scenarios (a different DVE configuration with a different number of servers, different movement patterns of avatars) have all been very similar. These results show that the problem of providing latencies below a given value to avatars in DVE systems can be addressed by means of the partitioning method, since the server-to-server delays represent a significant percentage of the total RTT. The reason for this behavior is that grouping neighbor avatars in the same server eliminates not only interserver network latencies (that could be negligible when compared to some client-server network latencies), but also the aggregated CPU latencies of one or more servers. As shown in the figure above, these aggregated latencies are not a negligible portion of the total latency provided to avatars even when the system works under low load. Therefore, if the system works below the saturation point, then the partitioning method can use the remaining CPU bandwidth of the servers for grouping those avatars whose latency is above the maximum latency threshold.

However, several problems may arise. First, grouping avatars in a given server can unbalance the workload, causing the entire system to enter saturation. The simulations shown in Fig. 1 have been entirely performed while keeping the servers below their saturation point (the best case). Therefore, the behavior shown in these figures is localized, probably becoming nonlinear if the number of assigned avatars had caused any server to become saturated. As shown in [27], saturation greatly decreases system performance, not only making the system unable to limit the latency offered to some avatars, but also greatly degrading the latency offered to all avatars. That is, the behavior shown in these figures can only be guaranteed if none of the servers reaches 100 percent of CPU utilization. Second, the effect on the number of avatars migrated from one to another server in each execution of the partitioning algorithm must be taken into account. In this sense, a recent work shows that an efficient partitioning method must not migrate more than 30 percent of avatars in the DVE system [20]. Therefore, the idea is to design a partitioning method that takes into account all these three aspects. This partitioning method must provide a near optimal assignment of avatars satisfying the following requirements at the same time: ensuring percentages of CPU utilization below 100 percent in all the servers, providing QoS (an average RTT lower than 250 ms) for the highest number of avatars possible, and also migrating the least number of avatars possible. We have denoted this challenge as the *QoS problem*. Since this problem shows high complexity, we have used a heuristic strategy to solve it. The idea is to model the QoS problem as a *quality function* associated to each partition of avatars (assignment of all the existing avatars to the servers in the system). The partitioning

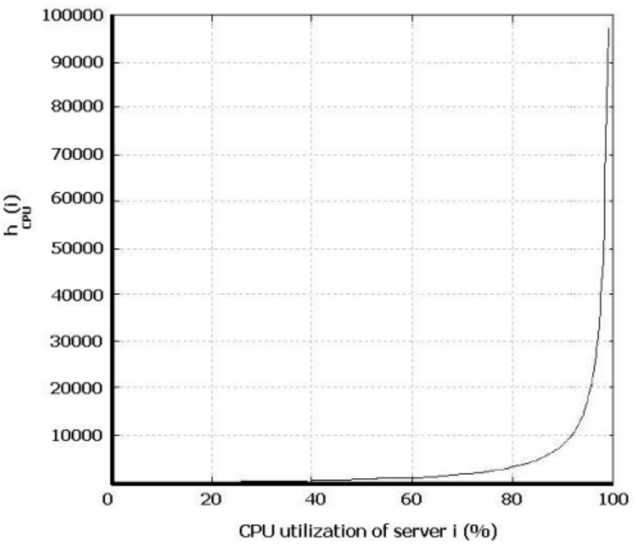

Fig. 2. Evaluation function $h_{cpu}(i)$.

method will consist of a heuristic search in order to find an efficient partition or assignment of avatars that minimizes the quality function.

We have denoted the quality function as f_{QoS}. Let P be a given partition (assignment) of avatars. Since the quality function must take into account the three factors described above, we have defined it as

$$f_{QoS}(P) = \Psi(P) + \Phi(P) + \Gamma(P), \tag{1}$$

where each term measures the quality of the partition according to one of these factors. Function $\Psi(P)$ evaluates the estimated percentage of CPU utilization in all the servers for partition P. The estimation of this term is based on the characterization method proposed in [27]. Function $\Phi(P)$ measures the estimated RTT that the considered partition will provide to the existing avatars. This term is computed taking into account the current RTT for the messages sent by each avatar and the current RTT to the servers where they are assigned. Finally, the term $\Gamma(P)$ measures the number of avatars migrated from one server to another by partition P. In order to compute this term, both the current assignment and the assignment defined by partition P are used. The heuristic search method should look for the partition with the lowest value of f_{QoS}.

Concretely, we have defined the $\Psi(P)$ function as

$$\Psi(P) = \sum_{i=1}^{s} h_{cpu}(i), \tag{2}$$

where S is the number of servers in the DVE system and the function $h_{cpu}(i)$ evaluates the estimated CPU utilization in each server i for partition P. This function has the exponential behavior shown in Fig. 2. If partition P results in an estimated CPU utilization significantly lower than 100 percent in server i, then the value of $h_{cpu}(i)$ will be very low. However, this value will increase if server i reaches a CPU utilization close to 90 percent, and it will shoot up if CPU utilization goes beyond this value for partition P. In this way, we ensure that the heuristic search will reject any

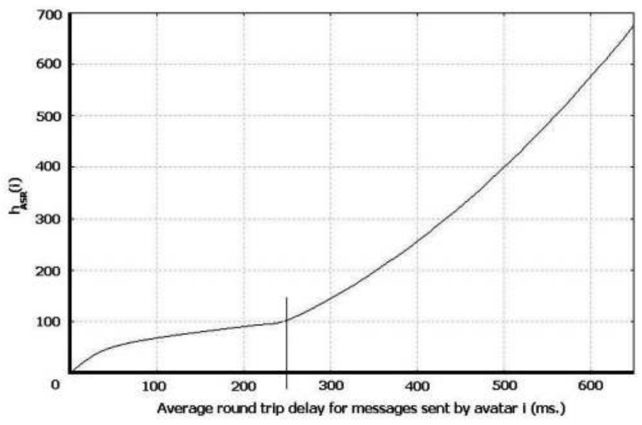

Fig. 3. Evaluation function $h_{asr}(i)$.

partition where any of the servers shows an estimated CPU utilization greater than 90 percent. In other words, this function prevents the DVE from entering saturation.

The second term in f_{QoS} is defined as

$$\Phi(P) = \sum_{i=1}^{n} h_{asr}(i), \tag{3}$$

where n is the number of existing avatars in the DVE system and $h_{asr}(i)$ is a function that evaluates the estimated average RTT that the messages sent by avatar i would have in partition P. Fig. 3 shows the behavior of $h_{asr}(i)$. This figure shows a plot with two differentiated sections. If the estimated RTT for the messages sent by avatar i is below 250 ms, then $h_{asr}(i)$ shows an inverse exponential behavior. If that value is greater than 250 ms, then $h_{asr}(i)$ shows a parabolic behavior. In this way, quality function f_{QoS} will select those partitions that result in a lower average RTT for most of the avatars, particularly if this value is below 250 ms.

Finally, $\Gamma(P)$ is a function that evaluates the partitioning efficiency of partition P, that is, the number of avatars that must be migrated in order to provide partition P. Fig. 4 shows the behavior of this function, which is composed of two sections: a linear section from the zero value to 1/3 of the existing avatars and a parabolic section from that value up. $\Gamma(P)$ helps f_{QoS} to select those partitions migrating the less number of avatars as possible.

Although the three terms in (1) measure different magnitudes, the plain sum of these terms without any weight results in a cost function that can be used to select the desired behavior. It must be noticed that $\Phi(P)$ is defined as the sum of n terms, where n is the number of clients in the system. For the case of DVE systems based on networked server architectures, where the number of servers s is very small if compared to the number of avatars n, function $\Phi(P)$ has a significant effect on $f_{QoS}(P)$ unless any server becomes saturated. Although weighting these terms may provide better performance to the proposed method, we have used the plain sum of these three terms as the quality function to be minimized in order to study the worst case, leaving the weighting of the different terms as a future work.

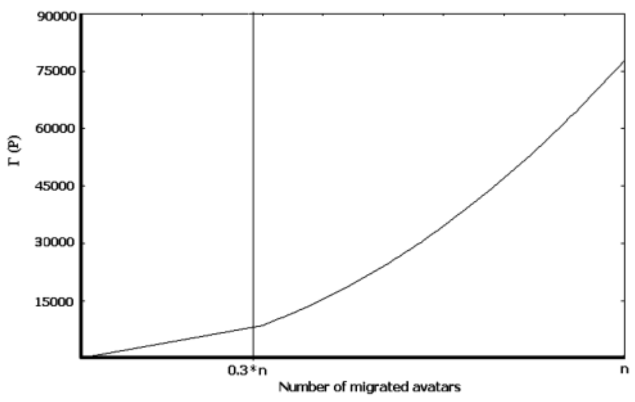

Fig. 4. Evaluation function Γ.

3 Heuristic Methods

Heuristic techniques have been proved as efficient strategies for solving highly complex problems, since they provide near optimal solutions while requiring reasonable execution times [36], [37], [38]. Heuristic search methods can be classified in three general types of techniques: stochastic methods such as Simulated Annealing (SA) [39], evolutive computation methods such as Genetic Algorithms (GA) [40], and constructive methods such as Greedy Randomized Adaptive Search Procedures (GRASP) [36], [41]. In this section, we present a new implementation and comparative study of two different heuristic techniques: GRASP and GA. Unlike the implementations shown in previous work [30], [31] (and as an enhancement of prior work), the implementations shown in this paper use a heap-data structure. The particular ordering property of this data structure [42] allows one to greatly reduce the time required for sorting different partitions. Therefore, it allows a faster execution of each of the heuristic search methods. Reducing the execution time of the partitioning methods is critical, since they must be executed while the DVE system is working and avatars are moving, even joining or leaving the application.

All of the considered methods start from an initial partition. In our implementations, we have imposed to the initial partition the only requirement of not saturating the DVE system. We have used the resulting partition provided by the fine-grain assembly line balance (FGALB) method [43] as the initial partition. This load-balancing method is an enhanced version of the assembly line balance (ALB) method [27] that provides better results and avoids the cascading effect. ALB is a load-balancing method that provides a partition where none of the servers reaches saturation. The basic difference between FGALB and ALB techniques is that, when a given server is saturated, the FGALB method distributes the exceeding workload of this server among the least loaded servers in the system instead of migrating this exceeding workload to a single server. Therefore, using this initial partition, we ensure that the DVE system is not saturated. It must be noticed that the FGALB method only focuses on system throughput, and it does not take into account the latency provided to avatars.

The proposed implementations of the heuristic methods deal with *border avatars*. Two avatars (a_i and a_j) in a DVE system are denoted as border avatars if they are assigned to different servers (s_r and s_x), and they can see each other in the virtual world. The assignment of border avatars is critical, since it allows obtaining partitions with low levels of h_{asr}. Both methods take into account these avatars to provide near-optimal partitions.

3.1 GRASP

GRASP is a constructive technique designed as a multistart heuristic for combinatorial problems [41]. It has been shown to quickly produce good quality solutions for certain problems [36].

The first step in our GRASP implementation consists of sorting the avatars in the initial partition whose messages show an RTT higher than 250 ms (those avatars not provided with QoS) by their presence factor f_p. We define the *presence factor* ($f_p(i)$) of avatar i as the number of avatars in whose AOI avatar i appears. The idea is to provide QoS (latency below a threshold value) to those avatars that require the least system efforts.

The first c elements in the sorted list of avatars (from a population of n avatars) are denoted as *critical avatars*. The GRASP method considers that critical avatars are nonassigned avatars, and they will be assigned by the GRASP method to a server in such a way that QoS is provided to them. The rest of the n avatars (denoted as the e *easy avatars*, where $n = c + e$) will not be reassigned, and they will remain assigned to the same server that they were initially assigned to. The assignment of each of the c critical avatars are obtained in each of the iterations of the GRASP method. The number of iterations (the number of reassigned avatars) is the only parameter to be tuned for the GRASP method. If the c value is set too low, then only a few avatars will be provided with QoS. However, if the c value is set too high (trying to provide too many avatars with QoS), then the GRASP method will not be able to find a partition fulfilling all the requirements for all avatars, and it will take a long time for providing bad partitions.

Each iteration of the GRASP method consists of two steps: construction and local search. The *construction* phase builds a feasible solution, choosing one critical avatar by iteration, and the *local search* derives this temporal solution following a neighborhood criterion. The complete process performed by the GRASP method is illustrated by the following algorithm, written as pseudocode:

```
program GRASP (Int threshold)

CONST
 S          /* Number of servers */
 n          /* Number of avatars (clients) */
 threshold  /* Number of nonassigned avatars */

VAR
 Fqos_GRASP, tmp_cost  : Real
 Ak, Sf,               : Integer
 non_assig             : Integer
 heap                  : Heap

begin
 Initial_Partition (FGALB,threshold)
 ClearHeap(heap)
 non_assig := n - threshold
 for i := 1 to non_assig do
   for j := 1 to n do
     k:= SelectRandomServer(S)
     tmp_cost = TestSolution(j, k)
     AddToHeap(heap, tmp_cost, j, k)
   end_for
   HeapSort (1/4, heap)
   ChooseRandomElement (1/4, heap, Ak, Sf)
   Fqos_GRASP = tmp_cost = INT_MAX
   for j := 1 To AvatarsInAOI (avatar) do
     k := SelectRandomServer (S)
     tmp_cost := TestSolution (j, k)
     if (tmp_cost < Fqos_GRASP)
       Fqos_GRASP : = tmp_cost
         Ak := j
         Sf := k
     end_if
   end_for
   AssingSolutionServer(Ak, Sf)
 end_for
end
```

3.2 A Sexual Elitist Genetic Algorithm (SEGA)

Genetic Algorithms (GA) consist of a search method based on the concept of evolution by natural selection [44]. A GA starts from an initial population, made of P *chromosomes*, that evolves following certain rules until reaching a convergence condition that maximizes a fitness function. Each iteration of the algorithm consists of generating a new population from the existing one by recombining or even mutating chromosomes. A chromosome contains a *genotype* or a string representing an individual (a particular solution of the problem) and also a *phenotype* or features that the genotype represents. We will use this additional information in the phenotype for tuning the behavior of the algorithm.

We have denoted as *SEGA* [31] the particular implementation of GA that we have used to solve the QoS problem in DVE systems. We derived this algorithm from trial and error, starting from that in [40]. In our implementation (SEGA), the genotype consists of an array defining the pair avatar-server. If there are N avatars in the system, this array contains N elements, each one designating the server where that avatar is assigned. The phenotype consists of information about the estimated workload that each avatar adds (in terms of the CPU utilization) to the server where it is assigned. Also, the phenotype indicates if each avatar is a *border avatar* or not. We use f_{QoS} as the fitness function to be (in our case) minimized.

Each iteration consists of generating a *descendant* generation of chromosomes, starting from an *ancestor* generation. The way that the algorithm provides the next generation determines the behavior of SEGA. We have chosen a sexual reproduction technique [40] in such a way that each descendant is generated starting from two ancestors. However, in order to provide high diversity, we have also used nonhomogeneous hybrid derivation (a certain rate of asexual reproduction). Additionally, we use *elitism* in each iteration and, hence, the name of the heuristic method. The term elitism means that some individuals (chromosomes) of a given population are directly passed to the next generation without suffering any variation [44].

Once the descendants are obtained in iteration t, if the finishing condition is not reached, then a recombination process is performed on all of the chromosomes of these descendants. This recombination process consists of randomly selecting two border avatars and exchanging the servers they are assigned to. This process helps to keep diversity while exclusively exploring highly probable solutions. Finally, a mutation can be performed on each resulting descendant. It consists of randomly selecting an avatar in a chromosome and changing its server. The whole process performed in each iteration can be expressed as the following pseudocode statements (where Genotype Gt is the resulting population of the previous iteration t, composed of P chromosomes):

```
Iteration t+1 (Genotype Gt)
CONST
 S          /* Number of servers in the DVE
               system */
 n          /* Number of avatar (clients) */
 Nelitist   /* Num. of elitist chromosomes */
 P          /* Num. of chromosomes in genotype */
 N          /* Num. of avatars in DVE system */
 Sexuality    /* Sexuality rate */

TYPE
 chromosome : int[N]

VAR
 int i
 Anc1, Anc2 : chromosome /* Ancestors */
 Desc : chromosome  /* Descendant */

begin
  Copy_Nelitist_best_of_Gt_to Gi()
  For i:= Nelitist to P+Nelitist do
    Anc1 := Gt[i]
    a := Reproduction_select(Sexuality)
    if a = 0 then /* 0 = Sexual, 1 = Asexual */
      Anc2 := Random_select_from(Gi)
      crossover := Random_select_crossover()
      case(crossover)
        one-point:
          Desc := 1point_cross(Anc1, Anc2)
        multipoint:
          Desc := mpoint_cross(Anc1, Anc2)
        uniform:
          Desc := unif_cross(Anc1, Anc2)
      end_case
    else
      Desc := Anc1
    end_if
    if (NOT converg_condition(Desc)) then
      recombination(Desc)
    end_if
    if (random() < mutation_rate) then
      mutation(Desc)
    end_if
    Gi[i] := Desc;
  end_for
  HeapSort(Gi);
  For i := 0 to P do
    Gt+1[i]} := Gi[i];
  end_for
end
```

3.3 Comparative Study

In this section, we present a comparative study of the heuristic methods described above in order to find which technique is the most suitable one for solving the QoS problem. In order to make a fair comparison, we have defined a certain DVE configuration and, then, we have performed a fine tuning of each method for that configuration. Concretely, we have performed a comparative study for two different configurations, denoted as MEDIUM1 and MEDIUM2. The MEDIUM1 configuration defines a DVE system composed of 250 avatars and three servers, and the MEDIUM2 configuration is composed of 700 avatars and 10 servers. The size of these configurations has been determined by our computing resources, since the final idea is to evaluate the partitions provided by the considered methods by simulation (as shown in Section 4). However, due to space limitations, in this section, we will only show the results for the MEDIUM2 configuration. The results obtained for MEDIUM1 followed the same pattern as the results shown here, except that, in the MEDIUM1 configuration, the workload that the servers support is significantly lower than in the case of the MEDIUM2 configuration.

Each of the considered heuristic techniques described in the previous sections have different parameters that must be properly tuned in order to obtain the best performance for each DVE configuration. For the sake of fairness, the considered methods have been tuned in order to obtain the best performance as possible for each configuration. Nevertheless, we do not present here the plots that illustrate the tuning of each parameter for the sake of shortness. Concretely, the only parameter to be tuned for the GRASP method is the number of initially nonassigned avatars c. We obtained 100 as the c optimum value for MEDIUM1 configuration and 200 avatars as the c optimum value for the MEDIUM2 configuration. The parameters to be tuned in the SEGA method are the following: the number of chromosomes in the population (P), the number of iterations, the mutation rate, number of elitist chromosomes ($Nelitist$), the sexuality rate (the percentage of iterations in which sexual reproduction is used), the minimum standard deviation allowed for the convergence condition, and the maximum number of iterations allowed without improving the fitness function. For both MEDIUM1 and MEDIUM2 configurations, we obtained the same values, except in the number of iterations. Concretely, we obtained as optimum values 100 chromosomes, a mutation rate of 5 percent of iterations, an elitism rate of 50 percent of the population, a sexuality rate of 75 percent of iterations, a minimum

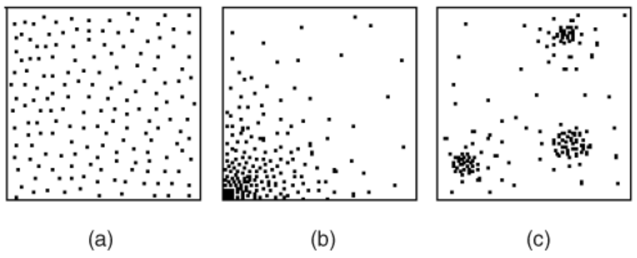

Fig. 5. Distributions of avatars in a 2D virtual world. (a) Uniform. (b) Skewed. (c) Clustered.

standard deviation of 0.005, and 50 iterations as the maximum number of iterations allowed without improving the fitness function f_{QoS}. For the MEDIUM1 configuration, we obtained an optimum value of 200 iterations, and for the MEDIUM2 configuration, we obtained an optimum value of 300 iterations.

In order to completely evaluate the performance of each of the considered heuristic techniques, we have executed each technique assuming that each technique is tested under different situations of the DVE system. We have used the distributions of avatars suggested in the literature: uniform, skewed, and clustered [9]. The reason for using different distributions is that they generate a different workload. Fig. 5 shows an example of these avatar distributions in a 2D virtual world. In this figure, the virtual world is a square, and avatars are represented as black dots.

For each of these distributions, we have computed the FGALB algorithm [43]. This algorithm is targeted to provide a well-balanced partition that keeps the DVE system below its saturation point. Using this partition as the initial partition, each one of the considered heuristic techniques has been then executed, and a final partition has been obtained. Table 1 shows the results for a MEDIUM2 configuration where avatars are located following a uniform distribution (at the instant when the partitioning method is executed, avatars are uniformly distributed in the virtual world). Each column in this table shows the results for each of the considered heuristic techniques (including a column for the partition provided by the FGALB method). The first row (labeled as f_{QoS}) shows the value of the quality function corresponding to the partition provided with each method. The second row (labeled as $\Gamma(P_0)$) shows the number of avatars migrated by the final partition from the initial partition. Finally, the last row (labeled as T_{exe}) shows the execution time required for each heuristic technique. The values in this table are the average values after 10 executions of each technique. Table 1 shows that the GRASP method provides the partition with the lowest value of the quality function while requiring the shortest execution time. Although the differences among the three methods are not significant in terms of the quality function, they are notorious if we focus on the required execution time. The GRASP method requires around 25 percent of the time required by the SEGA method. The reason for this behavior is that the HeapSort procedure is iterative in such a way that the GRASP method can stop HeapSort when it has provided the first quartile of the sorted solutions. That is, the GRASP technique is the method that best takes advantage of the new data structure. The next to last row of this table also shows that the three methods provide partitioning efficiency, migrating less than 30 percent of the existing 700 avatars.

Table 2 shows the results for a MEDIUM2 configuration when avatars are located in the virtual world following a skewed distribution. The results shown in this table are similar to the ones shown in Table 1. However, in this table, the values of the quality function provided by the three methods are higher than the ones in Table 1. The reason is that, in a skewed distribution, there are regions of the virtual world that show a high density of avatars. The avatars located in these crowded regions have an AOI with a lot of avatars in such a way that each movement of these avatars represents a lot of messages. Therefore, the average RTT for the messages sent by these avatars tends to increase. Also, the CPU utilization increases in all servers with respect to uniform distribution of avatars. Again, the GRASP method provides a partition with a slightly better value of the quality function, and it requires the shortest execution time.

Finally, Table 3 shows the results for a clustered distribution of avatars. The first row in this table shows that the partition provided by the GRASP method obtains the best value of the quality function, although the difference with the value provided by the SEGA method is not significant. Although the GRASP method migrates the greatest number of avatars, this number is far from

TABLE 2
Results for a MEDIUM2 DVE System with a Skewed Distribution of Avatars

	FGALB	GRASP	SEGA
f_{QoS}	99803	74319	77642
$\Gamma(P_0)$		167	153
T_{exe} (s.)		4.15	17.69

TABLE 1
Results for a MEDIUM2 DVE System with a Uniform Distribution of Avatars

	FGALB	GRASP	SEGA
f_{QoS}	70415	59182	61415
$\Gamma(P_0)$		133	78
T_{exe} (s.)		1.78	8.17

TABLE 3
Results for a MEDIUM2 DVE System with a Clustered Distribution of Avatars

	FGALB	GRASP	SEGA
f_{QoS}	94867	79864	80002
$\Gamma(P_0)$		134	103
T_{exe} (s.)		14.29	24.08

30 percent of the avatars. Regarding the execution times, it is worth mentioning that the values in this table are still higher than the ones in Table 2. Again, the GRASP method requires the shortest time.

Taking into account the results in these tables, the new implementation of GRASP search seems to provide the best performance for solving the QoS problem. This implementation requires the shortest execution times for all the distributions of avatars, whereas the values of the quality function provided by this method are similar to the values provided by the other methods. The main reason for this behavior is that the new implementation of the SEGA method does not exploit the iterative property of the HeapSort procedure. Also, it is worth mentioning that the f_{QoS} value provided by all the heuristic methods are significantly lower than the one provided by the FGALB method.

4 Performance Evaluation

The results in the previous section suggest that significant improvement can be achieved in the number of avatars provided with QoS (latency below a threshold value) if the proposed partitioning method is used, regardless of the heuristic technique used. However, these results are theoretical (values of the quality function), and practical results are needed to show that the proposed method can actually improve the QoS provided to avatars in a DVE system. In this section, we evaluate different DVE systems in order to measure the actual improvement that the proposed partitioning method can provide.

We propose the evaluation of generic DVE systems by simulation. The evaluation methodology used is the one described in [27]. A simulation consists of each avatar performing 100 movements at a rate of one movement every 2 seconds. An iteration consists of all the avatars performing one movement (that is, one iteration is performed every 2 seconds). Each time an avatar performs a movement, it (the client computer controlling that avatar) sends a message with a time stamp to all its neighbors and the neighbors send back an acknowledgment (ACK), as explained in [27]. The reason for using ACK messages is to avoid clock skewing when computing system latency.

For comparison purposes, we have simulated the partitioning method proposed in the previous section (using two different heuristic techniques, the GRASP and the SEGA methods) and the partitioning method FGALB proposed in [43] (the partitioning method that provides the initial partition). We have chosen GRASP and SEGA because these techniques provide the worst results for the clustered distribution of avatars (the distribution generating the highest workload), and the purpose of this section is to study the performance of the method in the worst case. We have simulated DVE systems with three movement patterns of avatars: Changing Circular Pattern (CCP) [45], Hot-Points-ALL (HPA) [46], and also Hot-Point-Near (HPN) [47]. CCP considers that all avatars in the virtual world move randomly around the virtual scene following circular trajectories. HPA considers that there exists certain "hot points" that all avatars approach sooner or later. This movement pattern is typical of multiuser games, where users must get resources (as weapons, energy, vehicles, bonus points, and so forth) that are located at certain locations in the virtual world. Finally, HPN also considers these hot points, but only avatars located within a given radius of the hot points approach these locations.

TABLE 4
Results for the MEDIUM1 DVE Configuration

	HPA Uniform		
	FGALB	SEGA	GRASP
S0 (max. %)	86	92	86
S1 (max. %)	94	84	93
S2 (max. %)	90	95	91
Av. ASR	221.7	157.3	155.0
QoS	155	238	225
Migrations	17	69	76
	HPN Clustered		
	FGALB	SEGA	GRASP
S0 (max. %)	94	98	93
S1 (max. %)	82	92	86
S2 (max. %)	91	87	92
Av. ASR	253.2	189.4	199.0
QoS	163	239	242
Migrations	22	106	113

All the considered methods (FGALB, as well as SEGA and GRASP) have been executed each time that an iteration finishes and the CPU utilization in any server reaches 95 percent, and the number of avatars provided with QoS in that iteration is saved. The GRASP and SEGA methods were executed after any subsequent iteration if this parameter decreased below the saved value in order to increase it again.

For the sake of shortness, we will only show here the results for some representative combinations of movement patterns and initial distributions of avatars. The experiments performed using the rest of the nine possible combinations of the movement patterns and initial distributions of avatars provided very similar results to those shown here for each DVE configuration.

Table 4 shows the results obtained for a MEDIUM1 configuration and for two different combinations of initial distributions of avatars and movement patterns. For each combination, the results for the different partitioning methods considered are shown in different columns. The first three rows in this table show the maximum percentage of CPU utilization reached in each server during the whole simulation. The fourth row shows the average value of the ASR provided to each of the avatars [27]. This measure provides the average RTT for all the messages sent during the simulation. The next row shows the number of avatars whose ASR has been lower than 250 ms during the simulation. That is, this value measures the number of avatars provided with QoS during the simulation. Finally, the last row shows the number of avatars migrated by each method during the simulation. The average execution time required by each method is not shown in this table because each partitioning method is executed several times during a single simulation under different circumstances. These

TABLE 5
Results for the MEDIUM2 DVE Configuration

	HPN Uniform		
	FGALB	SEGA	GRASP
S0 (max. %)	96	86	97
S1 (max. %)	92	93	96
S2 (max. %)	98	95	99
S3 (max. %)	86	96	96
S4 (max. %)	94	97	94
S5 (max. %)	96	88	91
S6 (max. %)	99	95	89
S7 (max. %)	92	94	98
S8 (max. %)	98	96	93
S9 (max. %)	97	98	99
Av. ASR	274.2	248.4	240.0
QoS	199	282	284
Migrations	29	113	128
	HPA Skewed		
	FGALB	SEGA	GRASP
S0 (max. %)	100	100	96
S1 (max. %)	100	100	100
S2 (max. %)	98	100	100
S3 (max. %)	99	100	94
S4 (max. %)	100	98	100
S5 (max. %)	97	96	96
S6 (max. %)	100	100	100
S7 (max. %)	100	99	96
S8 (max. %)	100	100	98
S9 (max. %)	100	100	96
Av. ASR	426.3	259.6	265.5
QoS	92	252	249
Migrations	36	312	301

circumstances may greatly differ not only between iterations of different simulations, but also between different iterations of the same simulation. Therefore, fairness is not guaranteed when showing the average execution times required for the different methods.

The first three rows in Table 4 show that all the methods are able to keep the system away from saturation. Although the SEGA method reaches a maximum CPU utilization in server S0 very close to this threshold (98 percent) for the HPN pattern, it must be taken into account that these are peak values. The fourth row shows that the two proposed methods improve the number of avatars provided with QoS with respect to the FGALB method. Thus, for the case of HPN movement pattern and a clustered initial distribution of avatars, the FGALB method provides QoS to around 65 percent of the existing avatars, whereas the GRASP and SEGA methods provide QoS to around 95 percent of the existing avatars. Regarding the partitioning efficiency, it can be seen that the GRASP and SEGA methods migrate more than 30 percent during the simulation. However, these partitioning methods have been executed at least five times during the simulations, resulting in an average number of migrated avatars that is actually lower than 30 percent. Therefore, all the considered methods (including FGALB) migrate less than 30 percent of the existing avatars in each execution.

Table 5 shows the evaluation results for the MEDIUM2 configuration. In this case, the combination of the HPA movement pattern and the skewed initial distribution of avatars makes the system work in deep saturation. The reason for this behavior is that, again, the avatars located in the crowded regions have an AOI with a lot of avatars in such a way that each movement of these avatars represents a lot of messages. Moreover, since, in this movement pattern, all avatars tend to concentrate in certain locations of the virtual world, the number of neighbors in the AOI of all avatars tends to increase, generating a greater workload in each subsequent iteration. As a result, in this case, none of the considered methods have been capable of performing 100 iterations without saturating the system. The FGALB method has kept the system away from saturation during 29 iterations, whereas the GRASP and the SEGA methods have kept the system away from saturation during 28 iterations. This is the reason for several servers to reach 100 percent CPU utilization in Table 5 for the HPA pattern. Therefore, the measurements in this case are all referred to a simulation of 28 or 29 iterations. It must be noticed that this is an extreme situation (if the existing servers are saturated, the system should be redesigned to host the actual number of clients). However, we have forced the system to reach this situation in order to study the improvement that the proposed technique could provide under extreme conditions.

Regarding the number of avatars provided with QoS, Table 5 shows that, for both HPN and HPA patterns, the proposed methods (both GRASP and SEGA) increase this value with respect to the FGALB method. Concretely, the number of avatars provided with acceptable latencies (labeled as QoS in Table 5) is increased by at least 41.7 percent (from 199 to 282) in the case of the HPN-Uniform pattern and by 170 percent (from 92 to 249) in the case of the HPA-Skewed pattern. Again, the GRASP and the SEGA methods were executed more than five times, at least, during the simulation. The number of avatars migrated in each execution has been lower than 30 percent of the population for all the partitioning methods. It is also worth mentioning that the results obtained by the GRASP and the SEGA methods are very similar, showing that the heuristic technique used to search the best partition does not have an important effect on the performance of the method. These results show that the proposed partitioning method is able to significantly increase the number of avatars provided with acceptable latencies, particularly when the DVE system supports a high workload (under saturation). That is, it efficiently helps to keep the QoS to the maximum number of clients.

5 Conclusions and Future Work

In this paper, we have shown that the problem of reducing the latency provided to avatars below a given value in DVE systems (a necessary condition for providing an acceptable QoS) can be addressed by means of the partitioning method. Also, we have proposed a new partitioning method, based on heuristic search, that looks for the partition representing the best trade-off among system throughput, system latency, and partitioning efficiency. Although we have used in this paper the latency threshold proposed in the literature, this method can be used for providing any other latency threshold, since it is designed to search an acceptable trade-off. Additionally, it can be used together with any other proposal for providing QoS.

The results show that the proposed method can significantly improve the number of avatars provided with an acceptable latency (acceptable QoS), regardless of both the value considered as the threshold latency and the workload level that the system supports (the CPU utilization of the servers in the system). When the workload generated by the application is below the aggregated CPU bandwidth of all the servers, then the proposed technique uses the remaining CPU bandwidth to group in the same server those avatars surrounded with the greatest amount of neighbors, thus decreasing the latency provided to the maximum number of avatars. Even when all the servers are close to saturation (Table 5), the proposed method is able to increase the number of avatars provided with an acceptable latency. Since these results are obtained with a quality function that is the plain sum of three terms, if this quality function is properly tuned, then the results can still be improved. Also, the results show that the particular heuristic method used to search the best partition does not have an important effect on the performance of the partitioning method itself. However, since the GRASP method requires the shortest execution time, this method adds the lowest overhead and, thus, we think that it is the most appropriate one.

As a future work to be done, we plan to improve the proposed method by tuning the relative weight of the three terms in the quality function for different cases. Also, we plan to design a partitioning that allows different latency thresholds for different avatars (different priorities) in the same DVE system.

Acknowledgments

This paper is supported by the Spanish MEC under Grants CONSOLIDER-INGENIO CSD2006-46 and TIN2006-15516-C04-04.

References

[1] S. Singhal and M. Zyda, *Networked Virtual Environments.* ACM Press, 1999.

[2] D. Miller and J. Thorpe, "Simnet: The Advent of Simulator Networking," *Proc. IEEE,* vol. 83, no. 8, pp. 1114-1123, 1995.

[3] J.S. Dias, R. Galli, A.C. Almeida, C.A.C. Belo, and J.M. Rebordao, "Mworld: A Multiuser 3D Virtual Environments," *IEEE Computer Graphics and Applications,* vol. 17, no. 2, pp. 55-65, 1997.

[4] C. Bouras, D. Fotakis, and A. Philopoulos, "A Distributed Virtual Learning Centre in Cyberspace," *Proc. Int'l Conf. Virtual Systems and Multimedia (VSMM '98),* Nov. 1998.

[5] J. Smed, T. Kaukoranta, and H. Hakonen, "A Review on Networking and Multiplayer Computer Games," Technical Report 454, Turku Centre for Computer Science, 2002.

[6] S. McCreary and K. Claffy, "Trends in Wide Area IP Traffic Patterns—A View from Ames Internet Exchange," *Proc. ITC Specialist Seminar,* Cooperative Assoc. for Internet Data Analysis (CAIDA), 2000.

[7] A. Steed and R. Abou-Haidar, "Partitioning Crowded Virtual Environments," *Proc. ACM Symp. Virtual Reality Software and Technology (VRST '03),* ACM Press, pp. 7-14, 2003.

[8] T. Funkhouser, "Network Topologies for Scalable Multi-User Virtual Environments," *Proc. IEEE Virtual Reality Ann. Int'l Symp.,* pp. 222-228, 1996.

[9] J.C. Lui and M. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 3, pp. 193-211, 2002.

[10] M.R. Macedonia, "A Taxonomy for Networked Virtual Environments," *IEEE Multimedia,* vol. 4, no. 1, pp. 48-56, 1997.

[11] C. Greenhalgh, A. Bullock, E. Frecon, D. Llyod, and A. Steed, "Making Networked Virtual Environments Work," *Presence: Teleoperators and Virtual Environments,* vol. 10, no. 2, pp. 142-159, 2001.

[12] R.B. Smith, R. Hixon, and B. Horan, *Collaborative Virtual Environments,* chapter on Supporting Flexible Roles in a Shared Space. Springer, 2001.

[13] S. Zhou, W. Cai, B. Lee, and S.J. Turner, "Time-Space Consistency in Large-Scale Distributed Virtual Environments," *ACM Trans. Modeling and Computer Simulation,* vol. 14, no. 1, pp. 31-47, 2004.

[14] R.M. Fujimoto and R. Weatherly, "Time Management in the DOD High Level Architecture," *Proc. 10th Workshop Parallel and Distributed Simulation,* pp. 60-67, 1996.

[15] D. Roberts and R. Wolff, "Controlling Consistency within Collaborative Virtual Environments," *Proc. IEEE Symp. Distributed Simulation and Real-Time Applications (DSRT '04),* pp. 46-52, 2004.

[16] L. Zou, M. Ammar, and C. Diot, "An Evaluation of Grouping Techniques for State Dissemination in Networked Multi-User Games," *Proc. Ninth Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecomm. Systems (MASCOTS '01),* 2001.

[17] D.B. Anderson, J.W. Barrus, J.H. Howard, C. Rich, C. Shen, and R.C. Waters, "Building Multiuser Interactive Multimedia Environments at MERL," *IEEE Multimedia,* vol. 2, no. 4, pp. 77-82, 1995.

[18] F.C. Greenhlagh, "Awareness-Based Communication Management in Massive Systems," *Distributed Systems Eng.,* vol. 5, no. 3, p. 129, 1998.

[19] H. Abrams, K. Watsen, and M. Zyda, "Three-Tiered Interest Management for Large-Scale Virtual Environments," *Proc. ACM Symp. Virtual Reality Software and Technology (VRST '98),* pp. 125-129, 1998.

[20] K. Lee and D. Lee, "A Scalable Dynamic Load Distribution Scheme for Multi-Server Distributed Virtual Environment Systems with Highly-Skewed User Distribution," *Proc. 10th ACM Symp. Virtual Reality Software and Technology (VRST '03),* pp. 160-168, 2003.

[21] H. Trefftz, I. Marsic, and M. Zyda, "Handling Heterogeneity in Networked Virtual Environments," *Presence: Teleoperators and Virtual Environments,* vol. 12, no. 1, pp. 37-51, 2003.

[22] M.V. Capps, "The Quick Framework for Task-Specific Asset Prioritization in Distributed Virtual Environments," *Proc. IEEE Virtual Reality Conf. (VR '00),* p. 143, 2000.

[23] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley & Sons, 1991.

[24] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach.* IEEE CS Press, 1997.

[25] D.L. Spohn, *Data Network Design.* McGraw-Hill, 1993.

[26] J. Blommers, *Practical Planning for Network Growth.* Prentice Hall, 1996.

[27] P. Morillo, J.M. Orduña, M. Fernández, and J. Duato, "Improving the Performance of Distributed Virtual Environment Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 7, pp. 637-649, July 2005.

[28] F.J. Alfaro, J.L. Sánchez, L. Orozco, and J. Duato, "Providing QoS in Infiniband for Regular and Irregular Topologies," *Proc. Canadian Conf. Electrical and Computer Eng. (CCECE '03),* 2003.

[29] P. Morillo, J.M. Orduña, M. Fernández, and J. Duato, "A Method for Providing QoS in Distributed Virtual Environments," *Proc. 13th Euromicro Conf. Parallel, Distributed and Network-Based Processing,* 2005.

[30] P. Morillo, J.M. Orduña, M. Fernández, and J. Duato, "A Comparison Study of Metaheuristic Techniques for Providing QoS to Avatars in DVE Systems," *Proc. Int'l Conf. Computational Science and Its Applications (ICCSA '04),* pp. 661-670, 2004.

[31] S. Rueda, P. Morillo, J.M. Orduña, and J. Duato, "A Sexual Elitist Genetic Algorithm for Providing QoS in Distributed Virtual Environment Systems," *Proc. Int'l Parallel and Distributed Processing Symp. Workshops (IPDPS '05),* 2005.

[32] Z. Choukair, D. Retailleau, and M. Hellstrom, "Environment for Performing Collaborative Distributed Virtual Environments with QoS," *Proc. Int'l Conf. Parallel and Distributed Systems (ICPADS '00)*, pp. 111-118, 2000.

[33] Y.W. Bernier, "Latency Compensating Methods in Client/Server In-Game Protocol Design and Optimization," *Proc. 15th Games Developers Conf.*, 2001.

[34] C. Faisstnauer, D. Schmalstieg, and W. Purgathofer, "Priority Scheduling for Networked Virtual Environments," *IEEE Computer Graphics and Applications*, vol. 20, no. 6, pp. 66-75, 2000.

[35] T. Henderson and S. Bhatti, "Networked Games: A QoS-Sensitive Application for QoS-Insensitive Users," *Proc. ACM Int'l Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '03)*, pp. 141-147, 2003.

[36] P. Festa and M. Resende, "Grasp: An Annotated Bibliography," *Essays and Surveys on Metaheuristics*, P. Hansen and C. Ribeiro, eds., pp. 325-367, Kluwer Academic, 2002.

[37] X. Yuan, "Heuristic Algorithms for Multi-Constrained Quality of Service Routing," *IEEE Trans. Networking*, vol. 10, no. 2, pp. 244-256, Feb. 2002.

[38] M. Randall and A. Lewis, "A Parallel Implementation of Ant Colony Optimization," *J. Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1421-1432, 2002.

[39] P.J. Laarhoven and E. Aarts, *Simulated Annealing: Theory and Applications (Mathematics and Its Applications)*. Springer, 1987.

[40] R.L. Haupt and S.E. Haupt, *Practical Genetic Algorithms*. Wiley, 1997.

[41] F.A. Feo and M.G. Resende, "Greedy Randomized Adaptive Search Procedures," *J. Global Optimization*, vol. 6, pp. 109-133, 1995.

[42] M.A. Weiss, *Data Structures and Algorithm Analysis in C++*. Addison-Wesley, 1999.

[43] P. Morillo, J.M. Orduña, M. Fernández, and J. Duato, "A Fine-Grain Method for Solving the Partitioning Problem in Distributed Virtual Environment Systems," *Proc. IASTED Int'l Conf. Parallel and Distributed Computing and Systems (PDCS '04)*, pp. 292-297, 2004.

[44] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1994.

[45] N. Beatrice, S. Antonio, L. Rynson, and L. Frederick, "A Multi-server Architecture for Distributed Virtual Walkthrough," *Proc. ACM Symp. Virtual Reality Software Technology (VRST '02)*, pp. 163-170, 2002.

[46] C. Greenhalgh, "Analysing Movement and World Transitions in Virtual Reality Tele-Conferencing," *Proc. European Conf. Computer Supported Cooperative Work (ECSCW '97)*, p. 313, 1997.

[47] M. Matijasevic, K.P. Valavanis, D. Gracanin, and I. Lovrek, "Application of a Multi-User Distributed Virtual Environment Framework to Mobile Robot Teleoperation over the Internet," *Machine Intelligence and Robotic Control*, vol. 1, no. 1, pp. 11-26, 1999.

Pedro Morillo received the MS and PhD degrees in computer engineering from the University of Valencia, Spain, with a dissertation on "Improving the Performance in Distributed Virtual Environments." Currently, he is an associate professor in the Department of Informatics, University of Valencia. In this department, he belongs to the Networking and Virtual Environments (GREV) group, where he focuses on the design and development of network architectures for distributed virtual environments. Furthermore, his research interests include distributed virtual environment systems, load balancing, metaheuristics, and cluster computing. He also served as a visiting scientist at Iowa State University, Ames, Iowa, in 2004 and the University of Louisiana, Lafayette, in 2006. For more details on his work, go to http://informatica.uv.es/~pmorillo.

Silvia Rueda is an assistant professor in the Department of Informatics, University of Valencia, Spain. She belongs to the Networking and Virtual Environments (GREV) group of this department. She is preparing her PhD dissertation, which she will be defending within a few months. Her research interests include distributed virtual environments systems, virtual reality, and nature-inspired algorithms.

Juan Manuel Orduña received the MS degree in computer engineering from the Technical University of Valencia, Spain, in 1990 and the PhD degree in computer engineering from the University of Valencia in 1998. He worked at Telefónica de España, Manpel Electrónica, and at the Technical University of Valencia as a computer engineer. Currently, he is a lecturer professor in the Department of Informatics, University of Valencia, where he leads the Networking and Virtual Environments (GREV) research group. His research is currently supported by the Spanish MEC, and it addresses interconnection networks, as well as distributed virtual environments. His research has been developed inside the ACCA team.

José Duato received the MS and PhD degrees in electrical engineering from the Technical University of Valencia, Spain, in 1981 and 1985, respectively. He is currently a professor in the Department of Computer Engineering (DISCA) of the same university. He was also an adjunct professor in the Department of Computer and Information Science, Ohio State University. His current research interests include interconnection networks, multiprocessor architectures, networks of workstations, and switch fabrics for IP routers. He has published more than 280 refereed papers. He proposed a powerful theory of deadlock-free adaptive routing for wormhole networks. He served as a member of the editorial boards of *IEEE Transactions on Parallel and Distributed Systems* and *IEEE Transactions on Computers*. He has served as cochair or member of the program committee for more than 40 conferences, including the most prestigious conferences in his area (HPCA, ISCA, IPPS/SPDP, ICPP, ICDCS, Europar, and HiPC). He is a member of the IEEE Computer Society.