Efficient Client-to-Server Assignments for Distributed Virtual Environments

Duong Nguyen Binh Ta[1], Suiping Zhou[1]

[1]Nanyang Technological University
School of Computer Engineering
Singapore 639798
{pa0236892b, asspzhou}@ntu.edu.sg

Abstract

Distributed Virtual Environments (DVEs) are distributed systems that allow multiple geographically distributed clients (users) to interact simultaneously in a computer-generated, shared virtual world. Applications of DVEs can be seen in many areas nowadays, such as online games, military simulations, collaborative designs, etc. To support large-scale DVEs with real-time interactions among thousands or more distributed clients, a geographically distributed server architecture (GDSA) is generally needed, and the virtual world can be partitioned into many distinct zones to distribute the load among the servers. Due to the geographic distributions of clients and servers in such architectures, it is essential to efficiently assign the participating clients to servers to enhance users' experience in interacting within the DVE. This problem is termed the client assignment problem. In this paper, we propose a two-phase approach, consisting of an initial assignment phase and a refined assignment phase to address this problem. Both phases are shown to be NP-hard, and several heuristic assignment algorithms are then devised based on this two-phase approach. Via extensive simulation studies with realistic settings, we evaluate these algorithms in terms of their performances in enhancing interactivity of the DVE.

1. Introduction

In recent years, advances in high-speed networking technologies, computer graphics and CPU processing power have enabled the rapid development of Distributed Virtual Environments (DVEs). DVEs are distributed systems that allow multiple geographically distributed clients (users) to explore and interact with each other in real-time within a shared, computer-generated 3D virtual world [23], where each client is represented by an *avatar*. A client controls the behavior of his/her avatar by various *inputs*, and the *updates* of an avatar's state need to be sent to other clients in the same *zone* of the virtual world to support the interactions among clients. DVEs have been applied in many areas, such as collaborative design, military simulations, e-learning and multi-player games [23].

Developing DVEs faces many challenges. In particular, various resources are needed, e.g., network bandwidth, CPU cycles, etc. The resource requirements of DVEs may increase quickly as the number of simultaneous clients increases. In addition, the network latency may damage the interactivity and consistency of DVEs [26]. Moreover, since DVEs are human-in-the-loop applications, cheating is also an important problem that need to be considered. Thus, usually a server-based communication architecture is employed for DVE applications. For example, popular Massively Multiplayer Online Games (MMOGs) such as Everquest [3] and Ultima Online [6] are operating on large clusters of servers. However, putting all servers at a central geographic location may result in high communication delays for clients which are far from the servers. Therefore, a *geographically distributed server architecture* (GDSA) is desirable [18, 8]. With this architecture, multiple geographically distributed servers are connected to each other, usually via well-provisioned connections. Each client is connected to one of these servers, and clients interact with each other through these servers.

In order to deal with large-scale DVEs with hundreds, or even thousands of clients interacting simultaneously, usually the virtual world is spatially parti-

1-4244-0054-6/06/$20.00

tioned into several distinct *zones*, with each zone managed by only one server, as in [3]. A client only interacts with other clients in the same zone,[1] and may move to other zones. As a server only needs to handle one or more zones instead of the entire virtual world, the system is more scalable. In this paper, we refer to such a partitioning approach as the *zone-based approach*.

Due to the fact that clients in DVEs are geographically distributed and the heterogeneous nature of the Internet, a client in a zone may have large network delays to the server hosting that zone, thus the interactivity of the DVE for that client may be greatly damaged. Hence, there is a strong need for mechanisms to assign the participating clients to servers in an efficient way to enhance the interactivity of the DVE. This is referred to as the *client assignment* problem (CAP). In this paper, we propose a two-phase approach to the CAP. In the *initial assignment* phase, the zones of the virtual world are assigned to servers. Then, in the *refined assignment* phase, a client is assigned to an appropriate (usually nearby) server to communicate with the server that is hosting the client's zone. Based on this two-phase approach, several assignment algorithms are devised and evaluated. We show that algorithms that take into account network delays in the initial assignment significantly outperform those without doing so.

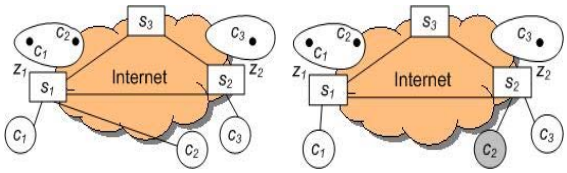
The rest of the paper is organized as follows. Section 2 describes the geographically distributed server architecture, the client assignment problem and some related work. The proposed algorithms are presented in Section 3. Simulation study is described in Section 4, and Section 5 concludes the paper.

2. Formulation of the client assignment problem

2.1. Architecture and definitions

In this paper, we focus on DVEs that adopt the geographically distributed server architecture (GDSA) [18, 8] and the zone-based partitioning approach. The GDSA consists of multiple servers geographically distributed over the Internet, as shown in Fig. 1. All geographically distributed servers are fully meshed with well-provisioned network connections. We note that many existing Content Distribution Networks (CDNs) like Akamai [1], Exodus [4], etc. have adopted such architecture to support the huge access demands to popular Web sites. DVE developers may deploy the server architecture on their own, or they may rent servers and network resources from third-party server-network providers to avoid excessive deployment and management costs [9].

[1]For simplicity, we say that a client is in a zone if its avatar is currently residing in that zone.



(a) (b)

Figure 1. Geographically distributed server architecture

Before formulating the client assignment problem, we introduce the following notations and concepts:

- c_i - A client in the DVE.
- $C = \{c_1, ..., c_k\}$ - The set that consists of all clients in the DVE.
- z_i - A zone in the DVE.
- $Z = \{z_1, ..., z_n\}$ - The set that consists of all zones in the DVE.
- s_i - A server in the DVE.
- $S = \{s_1, ..., s_m\}$ - The set that consists of all servers in the DVE.
- *Contact server* - The contact server of a client is the server that this client directly connects to. Clients only send inputs to their contact servers. The contact server may execute the inputs and send updates to the client if it is hosting the client's zone , or it may forward the client's inputs to another server which is hosting the client's zone. For example, in Fig. 1(a), s_1 is the contact server of c_1 and c_2.
- *Target server*: A target server of a client is the server that is hosting the client's zone. Inputs from a client will be forwarded to its target server. The target server may send updates to the client directly if it is also the contact server of the client, or it may send indirectly via the client's contact server. All clients in a zone have the same target server (therefore, we may say "target server of a zone"), while they may have different contact servers.

For example, in Fig. 1(a), s_1 is both the contact and target server of c_1 and c_2. In Fig. 1(b), we switch c_2 to server s_2 (but the avatar of c_2 is still in zone z_1), the target server of c_1 and c_2 is still s_1, the contact server of c_1 is s_1, while the contact server of c_2 is now s_2. Inputs from c_2 are forwarded to s_1 by s_2. Since the network connection between s_1 and s_2 is well-provisioned with little congestion, the communication between c_2 and s_1

may now have shorter delay. However, this incurs extra resource utilization for inter-server communication between s_1 and s_2.

- R_{s_i} - The resource consumption on a server s_i. This can be measured by CPU usage, network bandwidth usage, etc. Since the network bandwidth often represents the major operating cost in current server-based MMOGs [15], in this paper, we assume that the server CPU is not a bottleneck, and measure the resource consumption by the network bandwidth usage.
- $R^T_{c_i}$ - The amount of server resource, i.e., network bandwidth, utilized by a client c_i on its target server. Note that $R^T_{c_i} > 0, \forall c_i$.
- $R^C_{c_i}$ - The amount of server resource utilized by a client c_i on its contact server. Note that if the contact server and target server of c_i are the same then $R^C_{c_i} = 0$, otherwise $R^C_{c_i} = 2R^T_{c_i}$, since all communications between c_i and its target server are forwarded by its contact server (assuming the resource utilization is measured by bandwidth requirement).
- R_{z_i} - The amount of server resource utilized by a zone z_i on its target server. We have $R_{z_i} = \sum_{c_j \in z_i} R^T_{c_j}$.
- C_{s_i} - The resource capacity of a server s_i.
- $d_{c_i s_j}$ - The round-trip network delay between a client c_i and a server s_j.
- D - The *delay bound* of a DVE. The delay bound indicates the required upper bound of the round-trip communication delay between a client and its target server to guarantee the interactivity of the DVE. For different types of DVEs, there are different delay bound requirements. For example, Multi-player Real-Time Strategy (RTS) games typically require a delay bound of $500ms$ [24], while First-Person Shooter (FPS) games require a delay bound of $250ms$ [14]. It should be noted that the communication delay between a client and its target server is different from the network delay between the client and its target server. The communication delay is the sum of the network delay and the processing delay at the target server. However, in this paper we assume that the server CPU is not a bottleneck, thus the client-server communication delay is determined by the client-server network delay. In the following, the term "network delay" and "communication delay" are used interchangeably.

For interactive applications like DVEs, communication delay is the most important *Quality of Service* (QoS) parameter that the system provides to clients [14]. In this paper, we say that a client is *with QoS* or *without QoS* if the communication delay between the client and its target server is smaller or larger than the delay bound, respectively.

2.2. Client assignment problem

With the geographically distributed server architecture and the zone-based approach, the client assignment problem (CAP) concerns how to assign the participating clients in a DVE to servers so that the total number of clients with QoS is maximized. We formulate the client assignment problem as follows.

Definition 2.1. ***Client assignment problem (CAP)***

For each client c_j in the DVE, find the target server s_k and contact server s_l for c_j to maximize

$$|\{c_j \in z_i : d_{c_j s_k} = (d_{c_j s_l} + d_{s_l s_k}) \leq D, \forall z_i \in Z\}| \quad (1)$$

subject to

$$R_{s_i} \leq C_{s_i}, \forall s_i \in S \quad (2)$$

where $|\cdot|$ denotes the cardinality of a set.

Note that in the Definition 2.1, if s_l and s_k refer to the same server, then $d_{s_l s_k} = 0$.

2.3. A two-phase approach for the CAP

To address the client assignment problem, in this paper we propose a two-phase approach as follows. First, in the *initial assignment* phase, we need to assign the zones to servers, i.e., find a target server for each client. Then, in the *refined assignment* phase, each client is assigned to an appropriate server to communicate with its target server, i.e., find a contact server for each client.

In fact, both the initial assignment and refined assignment are themselves complex optimization problems. In order to obtain good solutions to the CAP, we must optimize some "assignment costs" in both phases. In the initial assignment, we propose the following metric to measure the cost of assigning a zone z_j to a server s_i

$$C^I_{ij} = |\{c_k \in z_j : d_{c_k s_i} > D\}| \quad (3)$$

C^I_{ij} measures the number of clients in a zone z_j that *do not* satisfy the delay bound D, i.e., without QoS. Therefore, by minimizing the total cost when all zones are assigned, the total number of clients with QoS in the DVE would be maximized. The initial assignment problem is formulated as follows.

Definition 2.2. ***Initial assignment problem (IAP)***

Let $I = \{1, ..., m\}$ and $J = \{1, ..., n\}$ be the set of indexes of servers and zones in the system, respectively. For each $i \in I$ and $j \in J$, given the cost C^I_{ij} of assigning zone z_j to server s_i as defined in (3), find an assignment matrix $x = (x_{ij})$, with $x_{ij} = 1$ if zone z_j is assigned to server s_i or $x_{ij} = 0$ otherwise, which minimizes the total cost

$$C^I(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} C^I_{ij} x_{ij} \tag{4}$$

subject to

$$\sum_{j=1}^{n} R_{z_j} x_{ij} \leq C_{s_i}, \forall i \in I, \tag{5}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j \in J, \tag{6}$$

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J. \tag{7}$$

Theorem 2.1. *The IAP described above is NP-hard.*[2]

After the initial assignment phase, some clients may still be without QoS. The refined assignment phase will attempt to further increase the number of clients with QoS in the DVE. One possible approach is to exploit the well-provisioned inter-server connections. We propose the following metric for the refined assignment to measure the cost of selecting server s_k as the contact server for a client c_j, where s_i is c_j's target server

$$C^R_{ij} = \begin{cases} (d_{c_j s_k} + d_{s_k s_i}) - D, & (d_{c_j s_k} + d_{s_k s_i}) > D \\ 0, & (d_{c_j s_k} + d_{s_k s_i}) \leq D \end{cases} \tag{8}$$

C^R_{ij} measures the "distance" from the delay bound D of the communication delay of a client c_j if c_j is assigned to server s_k as its contact server. Therefore, by minimizing the total cost when all clients are assigned, the total number of clients with QoS in the DVE would be maximized. The refined assignment problem is then formulated as follows.

Definition 2.3. ***Refined assignment problem (RAP)***

Let $I = \{1, ..., m\}$ and $J = \{1, ..., n\}$ be the set of indexes of servers and clients in the system, respectively. For each $i \in I$ and $j \in J$, given the cost C^R_{ij} of selecting server s_i as the contact server of client c_j, find an assignment matrix $x = (x_{ij})$, with $x_{ij} = 1$ if client c_j takes server s_i as its contact server and 0 otherwise, which minimizes the total cost

$$C^R(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} C^R_{ij} x_{ij} \tag{9}$$

subject to

$$\sum_{j=1}^{n} R^C_{c_j} x_{ij} \leq (C_{s_i} - R_{s_i}), \forall i \in I, \tag{10}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j \in J, \tag{11}$$

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J. \tag{12}$$

Theorem 2.2. *The RAP described above is NP-hard.*

Proof. The proof for RAP is similar to that for the IAP. □

2.4. Related work

To the best of our knowledge, there is no existing work that directly addresses the client assignment problem described in Section 2. Research on how to assign clients to servers in DVEs is usually formulated as a load balancing problem in a locally distributed server architecture, i.e., all the servers are placed in the same machine room [25, 17]. Such approaches may damage the interactivity of the DVE, since clients may be far away (in terms of network delays) from the servers.

In a more recent work [16], the authors proposed a distributed algorithm for clients to selects the best server in a mirrored architecture for online games, taking into account the network delay between clients and servers. The mirrored architecture replicates the DVE zones at multiple servers. This approach shares some similarities with the web server replica placement problem in CDNs [21]. However, unlike the replication of web documents, DVE replication faces serious consistency issues [26] which may damage the users' experience in interacting with the virtual world. In our approach, only one server has the control over the state of a zone, thus consistency can be guaranteed.

3. Assignment algorithms

To address the client assignment problem (CAP), we propose some algorithms for the initial assignment (IAP) and then for the refined assignment (RAP). Since both IAP and RAP are NP-hard, we seek for heuristic solutions instead of optimal ones. We start with algorithms for the IAP, with which we want to assign zones to servers, i.e., determine target servers

[2]Due to space limitation, we do not include the proof here.

for clients. We propose two algorithms for the IAP: the first one randomly assigns the zones, while the second one is a greedy heuristics to minimize the number of clients without QoS in the system. After the target servers for clients are determined, in the refined assignment phase, we find an appropriate contact server for each client. We again propose two algorithms. The first algorithm assigns clients to contact servers according to the virtual location of clients, and the second algorithm is a greedy heuristics similar to the one for solving the IAP. Finally, by combining the algorithms for the IAP and the RAP, we have the algorithms for the CAP. Thus we have in total four two-phase algorithms for the CAP.

3.1. Algorithms for the IAP

Random assignment of zones (RanZ). In RanZ, zones are assigned to randomly selected servers with the only concern of not overloading the servers. In this algorithm, the following procedure is repeated until all zones have been assigned: first the zone z_j with the largest number of clients is selected, and then a random server s_i with sufficient capacity is selected to take z_j, i.e., the target server of all clients in z_j is now s_i.

```
begin
    foreach z_j ∈ Z do
        find μ_ij = −C^I_ij, ∀s_i ∈ S;
        sort the list L^M_j of values μ_ij descendingly;
        find ρ_j = max_{s≠i_j} μ_sj − μ_{i_j j}, where
        i_j = arg max_i μ_ij;
    end
    sort the list L^R of values ρ_j in descending
    order;
    while Z ≠ ∅ do
        pick zone z_j with highest ρ_j;
        while L^M_j ≠ ∅ do
            pick the highest desirability μ_ij ∈ L^M_j;
            pick server s_i;
            if R_{s_i} + R_{z_j} ≤ C_{s_i} then
                assign z_j to s_i;
                R_{s_i} = R_{s_i} + R_{z_j};
                remove z_j from Z;
                break;
            end
            remove μ_ij from L^M_j;
        end
    end
end
```

Figure 2: IAP - Greedy assignment of zones

Greedy assignment of zones (GreZ). Since RanZ is oblivious to client-server network delays when assigning zones to servers, the obtained performance in terms of number of clients with QoS may not be good, i.e., the cost of the assignment as defined in Equation (4) may be high. Hence, in the GreZ algorithm, we use a greedy heuristics to minimize the total number of clients *without* QoS in the system.

The pseudo-code is shown in Fig. 2. Let $\mu_{ij} = -C^I_{ij}$ (recall that C^I_{ij} measures the number of clients without QoS in zone z_j if z_j is assigned to s_i) be a heuristic measure of the *desirability* of assigning zone z_j to server s_i. The smaller the cost C_{ij} is, the higher the desirability μ_{ij} is. The algorithm iteratively considers all the unassigned zones and pick a zone z_j with the maximum difference ρ_j between the largest desirability $\mu_{i_j j}$ and the second largest desirability μ_{sj}. Then, z_j is assigned to a server s_i with the highest value of μ_{ij} and with sufficient resource capacity. This procedure is similar to the approach used to solve the well-known Generalized Assignment Problem [22].

3.2. Algorithms for the RAP

Virtual Location based assignment of clients (VirC). The VirC algorithm adopts the most "natural" way to assign clients to servers in DVEs. It only considers the virtual location of each client c_j when determining a contact server for c_j, thus c_j will connect to the same server that is also hosting c_j's zone, i.e., the contact server of c_j is the same as its target server. This approach will not incur any inter-server communication cost. However, since network delay from each client to its target server is not considered, the number of clients with QoS may not be improved compared to the initial assignment.

Greedy assignment of clients (GreC). The GreC algorithm is a greedy heuristics which takes into account network delay from each client to its target server when selecting contact server for each client. The pseudo-code is shown in Fig. 3. This algorithm is similar to the GreZ algorithm for the IAP. The major difference between the two algorithms lies in the cost metric used. GreC uses the cost metric mentioned in Equation (8).

GreC starts by considering the round-trip delay to target server of each client in the system. If the network delay from a client c_j to its target server s_i is less than the delay bound, the algorithm selects the same server s_i as the contact server of c_j. Otherwise, c_j is added to a list L^E, which contains only clients having network delay larger than the delay bound. The next part of

```
begin
    foreach c_j ∈ C do
        get target server s_i of c_j;
        if d_{c_j s_i} ≤ D then
            select s_i as the contact server of c_j;
        else
            add c_j to the list L^E;
        end
    end
    foreach c_j ∈ L^E do
        find μ_ij = −C^R_ij, ∀s_i ∈ S;
        sort the list L^M_j of values μ_ij descendingly;
        find ρ_j = max_{s≠i_j} μ_sj − μ_{i_j j}, where
        i_j = arg max_i μ_ij;
    end
    sort the list L^R of values ρ_j in descending
    order;
    while L^E ≠ ∅ do
        pick client c_j with highest ρ_j;
        while L^M_j ≠ ∅ do
            pick the highest desirability μ_ij ∈ L^M_j;
            pick server s_i;
            if R_{s_i} + R^C_{c_j} ≤ C_{s_i} then
                select s_i as the contact server of c_j;
                R_{s_i} = R_{s_i} + R^C_{c_j};
                remove c_j from L^E;
                break;
            end
            remove μ_ij from L^M_j;
        end
    end
end
```

Figure 3: RAP - Greedy assignment of clients

the algorithm attempts to assign each client in L^E to a contact server to increase the number of clients with QoS in the system. This part is similar to the GreZ algorithm, except for the cost function used.

3.3. Two-phase algorithms for the CAP

A two-phase algorithm for the CAP is obtained by combining the algorithms for the IAP and the RAP. Thus, in total we have four different two-phase algorithms, namely RanZ-VirC, RanZ-GreC, GreZ-VirC and GreZ-GreC.

For comparison purposes, based on the Integer Programming formulation of IAP and RAP, we use the so-called "branch-and-bound" algorithm implemented in the Mixed Integer Linear Programming (MILP) solver lp_solve [5] to obtain the optimal solutions for both IAP and RAP. Note that this approach is only applicable when the system size is small, otherwise the running time of lp_solve will become very long (on the order of several hours), which is clearly impractical for highly interactive applications like DVEs.

3.4. Implementation considerations

In this section we address some practical considerations when one tries to implement the proposed assignment algorithms. The first consideration is the dynamic property of DVEs. During the course of interactions in the virtual world, clients may move from one zone to another, new clients may join, existing clients may also leave the virtual world. An obtained client assignment may not be good after some time. Thus, the proposed two-phase algorithm needs to be executed again to ensure good client assignments.

Another issue is how to obtain input data for the proposed assignment algorithms. The input data includes the client-server and inter-server round-trip network delays, and the server resource requirement of each client. The network delays can be obtained using scalable network measurement tools such as King [12] or IDMaps [11]. King uses existing recursive DNS queries to accurately estimate round-trip network delays between arbitrary Internet end hosts, while IDMaps relies on end hosts called *tracers* deployed at strategic locations in the Internet. Both approaches are scalable and incur little estimation overhead.

In this paper, the server resource requirement of each client is measured as the bandwidth requirement. It is well-known that the bandwidth requirement in client-server architectures increases quadratically with the total number of clients that are interacting with each other [20]. Thus, we can estimate in advance the bandwidth requirement of each client in a zone based on the number of clients in that zone, as in [20].

4. Simulation study

4.1. Simulation models and parameters

In our simulations, we use both synthetic topologies generated by the popular topology generator BRITE [2] and real topologies (e.g., the US AT&T continental IP backbone [13]), and obtain similar results. Due to space limitations, we only present here the simulation results with an Internet-like hierarchical topology generated by BRITE. The topology consists of 500 nodes with 20 AS (Autonomous System) domains using Barabasi-Albert model and 25 nodes using Waxman model for each AS. The clients' and servers' locations

Table 1. $pQoS(R)$ with different configurations

DVE conf.	RanZ-VirC	RanZ-GreC	GreZ-VirC	GreZ-GreC	lp_solve
5s-15z-200c-100cp	0.57 (0.6)	0.66 (0.77)	0.79 (0.6)	0.82 (0.66)	0.83 (0.73)
10s-30z-400c-200cp	0.57 (0.61)	0.69 (0.84)	0.83 (0.61)	0.88 (0.69)	0.89 (0.69)
20s-80z-1000c-500cp	0.61 (0.58)	0.75 (0.88)	0.89 (0.58)	0.94 (0.66)	-
30s-160z-2000c-1000cp	0.58 (0.58)	0.76 (0.93)	0.91 (0.58)	0.96 (0.65)	-

are randomly selected among these 500 nodes. The maximum round-trip delay between any two nodes is set to $500ms$. To simulate the well-provisioned inter-server connections which have lower delays than client-server connections, we set the network latency between any two geographically distributed servers to 50% of the actual latency values obtained from the topology generator, as in [16].

Based on existing studies of real online game systems, e.g., [10], we simulate various client distributions (clustered, uniform, etc.) both in the physical and the virtual world. In addition, it is noted that clients gathering in the same zone of a DVE may not necessarily close to each other in terms of their physical locations. On the other hand, studies have shown that clients that are close to each other in their physical locations (e.g., from the same country or the same geographic region) tend to gather in a specific zone of the virtual world due to their common cultural preferences. These phenomena may have great impact on the performance of the proposed algorithms. To model the correlation between clients' locations in the physical world and those in the virtual world, we use a correlation parameter δ, where $0 \leq \delta \leq 1$ [19]. The higher the value of δ is, the stronger the tendency for clients from the close geographic locations to gather in specific zones of the virtual world.

Unless otherwise stated, the following assumptions and default values are used in the simulations. The clients are uniformly distributed in the physical world as well as in the virtual world. The number of servers, number of zones, number of clients, and the value of correlation parameter are 20, 80, 1000 and 0.5, respectively. The minimum bandwidth capacity of server is 10 Mbps, and the total capacity of the system is 500Mbps. For estimating bandwidth requirement as in [20], the input sending frequency of each client (frame rate) is set to 25 messages per second, and the size of each input or update is 100 bytes, which are close to some real settings [7]. The interactive requirement, i.e., the DVE delay bound D is set to $250ms$. Two main performance measures, namely the percentage of clients with QoS in the system (measuring the interactivity of the system), denoted as $pQoS$, and the server resource utilization (measuring the cost of the algorithms), denoted as R, are of interest in the analysis. Results presented here are obtained by averaging the results of 50 simulation runs.

4.2. Results and analysis

The impact of different DVE configurations. In this set of experiments, we evaluate the performance of our algorithms with different DVE configurations. A specific DVE configuration is determined by the number of servers, the number of zones, the number of clients and the total resource capacity of the system. We use the notation *number of servers-number of zones- number of clients-capacity* to denote a DVE configuration. For example, the notation 20s-80z-1000c-500cp means that the DVE has 20 servers, 80 zones, 1000 clients and 500Mpbs server bandwidth in total. The values of $pQoS$ and R for different DVE configurations are shown in Table 1.

From Table 1, it is obvious that among the four proposed two-phase algorithms, GreZ-GreC has the best performance in terms of $pQoS$. The $pQoS$ values of GreZ-GreC are close to the optimal results given by the branch-and-bound algorithm implemented in lp_solve software. Note that lp_solve can only be applied to small size DVEs, i.e., the first two configurations of Table 1. The average execution time of lp_solve for these two DVE configurations are 0.2 and 41.5 seconds, respectively. For larger DVEs (the last two configurations), the results by lp_solve are not shown since the execution time was too long (not finished after more than 10 hours), which is clearly impractical for DVEs which need timely assignment decisions. In all configurations in Table 1, all of our proposed algorithms took less than 1 second of execution time.

In addition, it can be seen that the two algorithms that take into account network delays in the initial assignment phase, i.e., GreZ-VirC and GreZ-GreC, offer better interactivity than the other two which do not. This clearly shows the effectiveness of the greedy zone assignment GreZ.

Fig. 4 shows the cumulative distribution function (CDF) of delays from all clients in the configuration

of 30s-160z-2000c-1000cp to their target server for all algorithms. From this figure, it is clear that GreZ-GreC not only has a high ratio of clients with QoS, but also provides the best interactivity for clients that are without QoS.

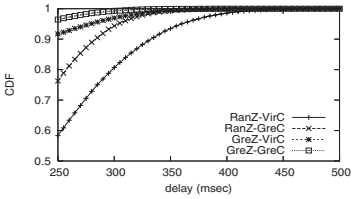
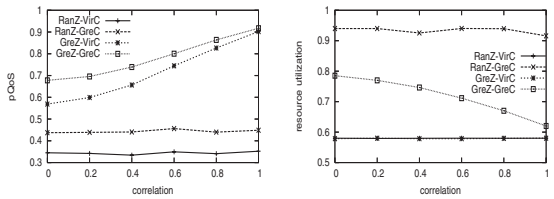


Figure 4. Cumulative distribution of delays

Table 1 also shows the resource utilization in terms of network bandwidth of all algorithms (the values in the brackets). RanZ-VirC and GreZ-VirC have the lowest resource consumption, since in these algorithms, each client c_i has the same server as both target and contact server, thus no extra bandwidth requirement $R^C_{c_i}$ is incurred. GreZ-GreC has a slightly higher bandwidth requirement than RanZ-VirC and GreZ-VirC, but its bandwidth requirement is lower than that of of RanZ-GreC. Given that the more critical concern for DVEs is the interactivity rather than the bandwidth, GreZ-GreC algorithm is the best choice over all algorithms, although GreZ-VirC is also a good alternative if the bandwidth requirement becomes more important.



(a) $pQoS$ (b) R

Figure 5. Impacts of correlations

The impact of correlation parameter. Fig. 5 shows the performance of our algorithms as the physical world-virtual world correlation value changes, and $D = 200ms$. It is observed that the $pQoS$ values of GreZ-VirC and GreZ-GreC which take into account network delays in the initial assignment phase increase significantly with the correlation value, while the results of RanZ-VirC and RanZ-GreC do not change much. This demonstrates the effectiveness of the greedy initial assignment (GreZ) when the clients in a zone are close to each other in the physical world. On the other hand, it seems that the greedy refined assignment (GreC) does not benefit from that phenomenon. In all cases, GreZ-GreC is still the best algorithm in terms of $pQoS$. In addition, the resource utilization of GreZ-GreC is reduced considerably when the correlation value increases, which further demonstrates the effectiveness of GreZ-GreC. It is also noted that GreZ-VirC is also an attractive alternative when the correlation value is very high.

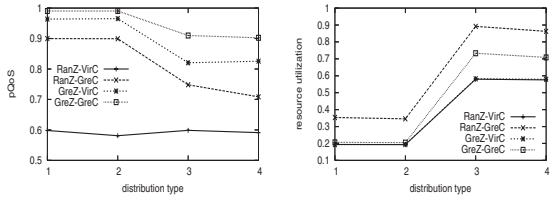
The impact of virtual world and physical world distribution. In this experiment, we evaluate the impact of different client distributions in both the virtual world and the physical world on the performance of the proposed algorithms. The number of clients may be higher in some specific zones of the virtual world than others, due to the clustering of clients in some "hot" zones. For example, in online games, clients may be clustered in the zones with high amounts of game resources such as energy, gold, etc. In the physical world, due to the differences in time zones of geographically distributed clients, at a specific time, the number of online clients in the DVE may be quite different for different geographic regions [10].

We simulate the clustering behavior of clients in the virtual world by randomly selecting some zones to have more clients than other zones. To simulate the clustering of clients in the physical world, some nodes in the network topology are randomly selected to have a large number of clients than the rest nodes. We have simulated a large number of different scenarios by changing the number of clusters and number of clients in each clusters, and obtained similar results. In this paper, we present the simulation results for the 20s-80z-1000c-500cp configuration. For the clustered distribution in the virtual world, the number of clients in a clustered zone is 10 times larger than that in a non-clustered zone. The clustered distribution for the physical world is generated in a similar manner. Table 2 shows the combination of different virtual world (VW) and physical world (PW) distributions.

Table 2. Distribution types

Type	0	1	2	3
Clusters in PW	No	Yes	No	Yes
Cluters in VW	No	No	Yes	Yes

The simulation results for all algorithms with different client distributions are shown in Fig. 6. Although GreZ-GreC's $pQoS$ decreases slightly when clients are clustered in the virtual world (distribution type 3 and



(a) $pQoS$ (b) R

Figure 6. Impacts of client distributions

4), it's still the best algorithm in terms of interactivity. Besides, it is noted that the resource utilization for all algorithms in the distribution type 3 and 4 are much larger than those in distribution type 1 and 2. This shows that the clustered distribution of clients in the virtual world has a great impact on the bandwidth requirement of the system. Meanwhile, it seems that the clustered distribution of clients in the physical world do not have significant impacts on both performance measures of all algorithms.

Table 3. $pQoS$ with DVE dynamics

Time	Before	After	Executed
RanZ-VirC	0.59	0.59	0.59
RanZ-GreC	0.73	0.68	0.71
GreZ-VirC	0.83	0.79	0.82
GreZ-GreC	0.9	0.83	0.9

The impact of DVE dynamics. In this experiment, we evaluate all algorithms in dynamic settings. First, we obtain a client assignment with the configuration 20s-80z-1000c-500cp, and correlation $\delta = 0$. Then, to simulate the dynamic properties of DVEs, we let 200 new clients randomly join, 200 existing clients randomly leave the virtual world and 200 clients randomly move to another zone. The simulation results are shown in Table 3. The column "Before" shows the $pQoS$ of all algorithms before we let the clients join, leave and move. The column "After" shows the $pQoS$ after we let the clients join, leave and move. The data in this column clearly indicates that right after the join-leave-move happened, the interactivity of all algorithms except RanZ-VirC[3] are decreased. Hence, there's a strong need to re-execute these algorithms[4] to achieve better performance. The column "Executed" shows the performance of all algorithms after they are all re-executed. The results show that our algorithms can cope well with the dynamic properties of DVEs.

[3]Since RanZ-VirC makes assignment decision randomly, its performances is not much affected.

[4]$pQoS$ of RanZ-VirC may not be affected, but we may need to re-execute it since the resource constraint may be violated.

The impact of imperfect input data. The simulation results obtained above are based on the assumption that we have perfect information about the network delays between clients and servers. In practice, we usually have rough estimations of network delays rather than perfectly accurate information. To simulate the estimation error, similar to [21], we apply an error factor e to the perfect input data, i.e., assuming the perfect value of delay is d, then the delay value used in the simulation is uniformly distributed in the range $[\frac{d}{e}, de]$. We use two values of e equal to 1.2 and 2, representing the inaccuracies of the popular network delay estimation tools King [12] and IDMaps [11], respectively.

Table 4. Impacts of imperfect input data

e	1.2	2
RanZ-VirC	0.58 (0.58)	0.59 (0.58)
RanZ-GreC	0.7 (0.91)	0.57 (1)
GreZ-VirC	0.86 (0.58)	0.8 (0.58)
GreZ-GreC	0.9 (0.67)	0.78 (0.82)

Table 4 shows the simulation results obtained with the inaccurate estimations of network delays. For $e = 1.2$, GreZ-GreC is still the best algorithm in terms of interactivity, and its $pQoS$ only decreases about 0.04 compared to the case that uses perfect information in Table 1. When the estimation error becomes very large, i.e., $e = 2$, we see that GreZ-VirC is the best: it performs slightly better than GreZ-GreC in terms of interactivity and has the lowest resource utilization (the value in the brackets). This is because GreZ-GreC uses delay information in both the initial assignment and refined assignment, thus, its performance is affected by the large estimation error in both phases, while GreZ-VirC only utilizes network delays in the initial assignment. In overall, despite the imperfect knowledge in network delay, these two algorithms are still much better than the other two algorithms which do not consider network delays in the initial assignment phase. This shows that GreZ-GreC and GreZ-VirC are very useful in enhancing DVE interactivity in Internet environments, where perfectly accurate input data for assignment algorithms is usually impossible or very costly to obtain.

5. Conclusions

Supporting large-scale DVEs with thousands of simultaneous clients interacting in real-time is a challenging task. In this paper, we have described the geographically distributed server architecture for DVEs. In this architecture, multiple geographically distributed servers are connected with each other via well-provisioned networks to provide low-latency inter-server communications, and the large virtual world is partitioned into distinct zones to distribute load among the servers. The client assignment problem arises when assigning participating clients to servers to enhance the interactivity of the DVE. In this paper, a two-phase approach, which consists of an initial assignment and a refined assignment is proposed to simplify the client assignment problem. Based on this approach, several assignment algorithms are devised. Extensive simulation results on realistic settings shows that algorithms that make use of network delay information in the initial assignment phase perform better in terms of interactivity than those which do not, even if the input data to these algorithms is inaccurate. We believe that our two-phase approach with these algorithms (i.e., GreZ-GreC and GreZ-VirC) could be very helpful to the designers and researchers of large-scale DVEs.

Acknowledgement

The authors would like to thank Gunther Raidl for helping with the NP-hardness of the GAP.

References

[1] Akamai. *Available at http://www.akamai.com*.

[2] Brite internet topology generator. *Available at http://www.cs.bu.edu/brite*.

[3] Evequest. *Available at http://www.everquest.com*.

[4] Exodus. *Available at http://www.exodus.com*.

[5] Mixed integer linear programming solver lp_solve. *Available at http://groups.yahoo.com/group/lp_solve*.

[6] Ultima Online. *Available at http://www.uo.com*.

[7] A. Abdelkhalek, A. Bilas, and A. Moshovos. Behavior and Performance of Interactive Multi-player Game Servers. *Special Issue of Cluster Computing: the Journal of Networks, Software Tools and Applications*, 2002.

[8] D. Bauer, S. Rooney, and P. Scotton. Network infrastructure for massively distributed games. In *Proc. of NetGames*, 2002.

[9] Z. Duan, Z. Zhang, and Y. T. Hou. Service overlay networks: Slas, qos and bandwidth provisioning. In *Proc. of the 10th IEEE International Conference on Network Protocols*, 2002.

[10] W. Feng and W. Feng. On the geographic distribution of online game servers and players. In *Proc. of NetGames*, 2003.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y.Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, 9(5), 2001.

[12] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proc. of the ACM SIGCOMM IMW*, 2002.

[13] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz. Generating realistic isp-level network topologies. *IEEE Communication Letters*, 2003.

[14] T. Henderson and S. Bhatti. Networked games: a QoS-sensitive application for QoS-insensitive users? In *Proc. of the ACM SIGCOMM*, 2003.

[15] B. P. J. Mulligan. *Developing Online Games: An Insider's Guide*. New Riders Games, 2003.

[16] K. W. Lee, B. J. Ko, and S. Calo. Adaptive Server Selection for Large Scale Interactive Online Games. *Computer Networks*, 49:84–102, 2005.

[17] J. Lui and M. Chan. An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. *IEEE Transaction on Parallel and Distributed Systems*, 13(3), 2002.

[18] M. Mauve, S. Fischer, and J. Widmer. A generic proxy system for networked computer games. In *Proc. of NetGames*, 2002.

[19] C. D. Nguyen, F. Safaei, and P. Boustead. Comparison of distributed server architectures in providing immersive audio communication to massively multiplayer online games. In *Proc. of ATNAC*, 2004.

[20] J. D. Pellegrino and C. Dovrolis. Bandwidth requirement and state consistency in three multiplayer game architectures. In *Proc. of the ACM NetGames*, 2003.

[21] L. Qiu, V. Padmanabhan, and G. Voelker. On the placement of web server replicas. In *Proc. of IEEE INFOCOM*, 2001.

[22] H. Romeijn and D. R. Morales. A class of greedy algorithms for the generalized assignment problem. *Discreet Applied Mathematics*, 103:209–235, 2000.

[23] S. Singhal and M. Zyda. *Networked virtual environments: design and implementation*. Addison-Wesley, Reading, MA, 1999.

[24] J. Smed, T. Kaukoranta, and H.Hakonen. Aspects of Networking in Multiplayer Computer Games. In *Proc. of the International Conference on Application and Development of Computer Games in the 21st Century*, pages 74–81, 2001.

[25] D. N. B. Ta and S. Zhou. A Dynamic Load Sharing Algorithm for Massively Multi-Player Online Games. In *Proc. of the 11th IEEE International Conference on Networks*, 2003.

[26] S. Zhou, W. Cai, B. S. Lee, and S. J. Turner. Time-space consistency in large-scale distributed virtual environments. *ACM Transactions on Modeling and Computer Simulation*, 14(1):31–47, 2004.