Dynamic Assignment of Zones to Servers for Large-Scale Distributed Virtual Environments

Jian Guo Suiping Zhou
Nanyang Technological University
Singapore 639798
{guojian, asspzhou}@ntu.edu.sg

Abstract

In Distributed Virtual Environments (DVEs), where many human users simultaneously interact with each other in a shared, 3D virtual world, the interactivity of such systems is of crucial importance, since the success of a DVE depends greatly on users' perceptions. In general, enhancing interactivity for large-scale DVEs with thousands of geographically distributed users faces various challenges such as large Internet latencies, high resource demands and high potential of security threats. In this paper, we propose a dynamic zone-server assignment approach to enhance the interactivity of DVEs by dynamically re-arranging zone-server assignment. We give the conditions on when to carry out the re-arrangement and the way how to carry out the re-arrangement. Simulation studies have shown that our approach is very effective in enhancing the interactivity of large-scale DVEs.

1 Introduction

In recent years, advances in networking technologies, computer graphics and CPU power have popularized the deployments of Distributed Virtual Environments (DVEs). Large-scale DVEs refer to a class of applications that enable thousands of geographically distributed users to simultaneously interact with each other in a shared, computer-generated 3D virtual world, where each client is represented by an avatar [8]. A client controls the behavior of his/her avatar by various inputs, and 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. Prominent applications of DVEs include online games, military simulations, collaborative designs, virtual shopping mall, etc. A study by the market research company IDC [6] on the online gaming market of Asia/Pacific (excluding Japan) revealed that the subscription revenue is about US$1.09 billion in 2004, an increase at about 30% compared to 2003. According to this study, this market will be more than doubled in 2009.

Enhancing the interactivity of DVEs is of crucial importance, since the users may not feel that they are interacting with other elements in the virtual world if the responses from the DVE are much slower than what the users experience in real-life. However, in general, seeking a good balance between interactivity and other requirements of DVEs such as consistency is a challenging task. This problem is exacerbated in large-scale DVEs mainly due to the large, non-deterministic message transmission delay over the Internet and the huge resource demands to maintain the virtual world. In this paper, we have proposed a dynamic zone-server assignment approach to enhance the interactivity of large-scale DVEs. Our methodology is different from traditional application-centric approaches, e.g., dead reckoning, which aims to ``hide" the effect of network latencies by predicting the state of remote users at application level. Since the activities of human users in DVEs are highly unpredictable, such approach may result in visual perception errors, or in more serious cases, inconsistent DVE states [11].

Our dynamic zone-server assignment approach is based on a geographically distributed server architecture (GDSA) [1][7], in which multiple geographically distributed servers are connected to each other. Each client is connected to one of these servers. This server-based architecture essentially provides good ways to implement consistency control, resource management and security mechanisms. In addition, in order to deal with large-scale DVEs with hundreds, or even thousands of users interacting simultaneously, usually the virtual world is spatially partitioned into many distinct zones, with each zone managed by only one server, as in [3]. A client only interacts with other clients in the same zone, and may move to other zones. As a server only needs to handle a few 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*.

In a static zone-server assignment, the servers control distinct zones with all clients in a zone connected to the same server that controls the zone. The problem with this static assignment approach is that the client-server communication delay may become large as the avatars move to different zones, which could damage the interactivity of the DVE.

In this paper, we consider an important problem with GDSA and the zone-based partitioning approach: dynamic zone-server assignment problem. Dynamic zone-server assignment problem aims to reduce the client-server communication delay by carrying out re-assignment when necessary. To address this problem, we have developed some dynamic zone-server assignment algorithms. We also give the conditions on when to carry out the re-assignment and the way how to do it. Simulation studies have shown that our approach is very effective in enhancing the interactivity of large-scale DVEs.

The rest of the paper is organized as follows. Section 2 formulates the dynamic zone-server assignment problem and discusses some related work. Section 3 gives our solutions to the problem. Simulation studies are described in Section 4, and Section 5 concludes this paper.

2 Dynamic Zone-server assignment problem

2.1 Problem Description

The general structure of the GDSA is shown in Figure 1. The right hand side of Figure 1 shows the relationship between servers and clients based on round trip time (RTT). For simplicity, we use the same symbol (e.g., $c1$) for a client and its avatar. In this paper we refer to RTT as the communication delay between clients and servers, and the interactivity of DVEs is measured by this communication delay. Here RTT between a server and a client refers to the minimum RTT as a client may have more than one route to connect to a server. In Figure 1, $c1$ is drawn near to $s1$ to illustrate that the RTT between $c1$ and $s1$ is small, i.e., the smaller the RTT between a client and a server is, the closer the client is to a server in the figure.

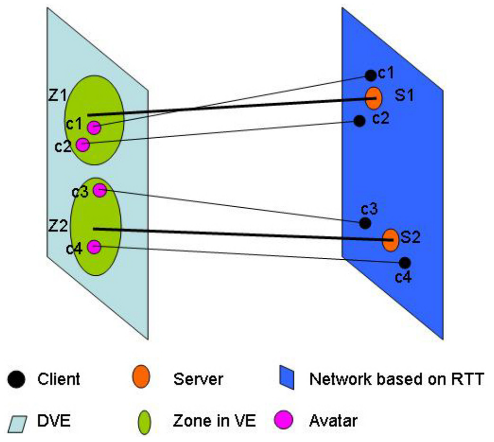

Figure 1: Client-DVE-Server relationship 1

The left hand side of Figure 1 shows the relationship between servers and avatars in the virtual environment. Each client is represented as an avatar in the Virtual Environment (VE). It can interact with other avatars within the same zone but not those in other zones. An avatar can move into a different zone. All avatars in the same zone are under the control of the same server which hosts the zone. As shown in the figure, server $s1$ hosts zone $z1$, and $c1$ and $c2$ are inside $z1$, so $c1$ and $c2$ are under control of $s1$. $c1$ and $c2$ only send "update" to server $s1$, they communicate through $s1$, i.e., there is no direct communication between any clients.

Generally, the RTTs between clients and servers can be considered as fixed since the geographical location of clients and servers can not be easily changed. However, the positions of avatars in the virtual environment may change over time. It may happen that avatars move into a zone hosted by a server which is far away from the clients in terms of RTT. For example in Figure 2, $c1$ and $c2$ move into zone $z2$ which is hosted by server $s2$, $c3$ and $c4$ move into zone $z1$ which is hosted by server $s1$. The RTTs between these two clients ($c1$ and $c2$) and $s2$ become high, and the same situation for $c3$ and $c4$. For such cases, we can re-assign zone $z1$ to server $s2$, $z2$ to $s1$ such that the overall RTT from the clients to their hosting servers is reduced, as illustrated in Figure 3.

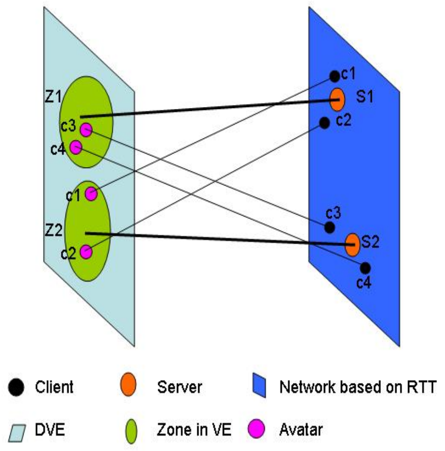

Figure 2: Client-DVE-Server relationship 2

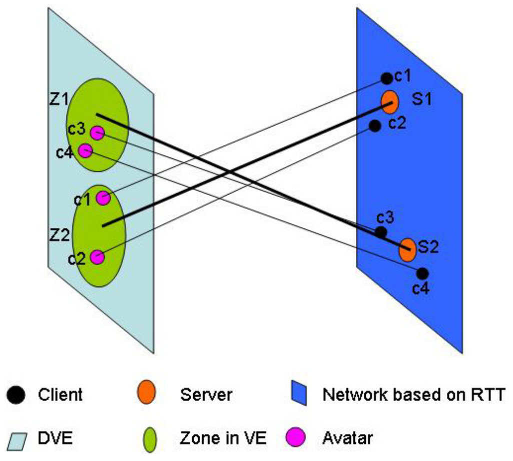

Figure 3: Client-DVE-Server relationship 3

For more general cases, we can carry out such re-assignment whenever the average RTT from the clients to their hosting servers can be reduced.

2.2 Problem Formulation

Before formulating the dynamic zone-server assignment problem, we introduce the following notations and concepts:

- $C = \{C_1, C_2, \cdots, C_{N_C}\}$ - the set of clients, where C_i is a client in the DVE and the total number of clients is N_C.

- $S = \{S_1, S_2, \cdots, S_{N_S}\}$ - the set of servers, where S_i is a server in the DVE and the total number of the servers is N_S.

- $Z = \{Z_1, Z_2 \cdots, Z_{N_Z}\}$ - the set of zones, where Z_i is a zone of the DVE and the total number of the zones is N_Z. In this paper we assume that each server hosts exactly one zone, thus, $N_Z = N_S$.

- $\| Z_i \|$ - the number of avatars (clients) in Z_i.

- $d(C_i, S_j)$ - the RTT between client C_i and server S_j.

- $d(S_i, S_j)$ - the communication delay between server S_i and S_j. We further assume that well provisioned network has been setup between servers, thus, $d(S_i, S_j)$ is small and remains stable.

- $S(Z_i)$ - the server that hosts zone Z_i.

- Permutation - a rearrangement of the elements of an ordered list *L* into a one-to-one correspondence with *L* itself.

- Permutation cycle - a permutation whose elements trade places with one another.

- Transposition - a permutation of two elements, which is also a permutation cycle of two elements.

- $M : Z \to S$ - the assignment of zones to servers. Since $N_S = N_Z$, *M* is a permutation of order N_S.

- $T(Z_i, S_x)$ - the total round trip time between all clients inside zone Z_i and a server S_x, which can be calculated as:

$$T(Z_i, S_x) = \sum_{C_j \in Z_i} d(C_j, S_x) \qquad (1)$$

- $T_A(Z_i)$ - the average communication delay between the clients in a zone Z_i and server $S(Z_i)$, which can be calculated as:

$$T_A(Z_i) = \frac{T(Z_i, S(Z_i))}{\| Z_i \|} \qquad (2)$$

- $swap(S_\alpha, S_\beta)$ - Swap all clients hosted by S_α with all clients hosted by S_β. Suppose that originally server S_α is hosting Z_i, hence all the clients in zone Z_i are under control of S_α; and server S_β is currently hosting zone Z_j, hence all the clients in the zone Z_j are under control of S_β. After the $swap(S_\alpha, S_\beta)$ operation, server S_α will host zone Z_j and all clients inside zone Z_j will be under control of S_α; and server S_β will host zone Z_i and all clients inside Z_i will under control of S_β. The *swap* operation between two servers is carried out by transferring clients under control to the other server, as illustrated in Figure 2 and Figure 3.

Finally we formulate the dynamic zone-server assignment problem as follows: *find the assignment M that minimizes $T_A(Z)$.*

2.3 Related Work

The major contribution of this paper is that we give the conditions on when to do the re-assignment swapping and the way how to do the re-assignment swapping considering the dynamic nature of avatars in DVEs.

To the best of authors' knowledge, there is not much existing work on zone-server assignment that takes into account of the dynamic nature of avatars in DVEs. There are some static server selection mechanisms for DVEs. In [4], the authors gave a server selection method based on the communication delay and communication frequency among clients. However they did not discuss how to deal with dynamic situations when the server selection must change due to the movement of avatars.

In [9] and [10], the authors proposed some clients' assignment methods to reduced the communication delay. However, their goal is to minimize the upper-bound of communication delay for the DVEs, which may not optimize the overall system.

3 Algorithms and Solutions

3.1 Condition on *swap* Operation

Theoretically, at any point of time, we can find an optimal *M* such that $T_A(Z)$, the overall average delay from clients to their hosting servers at that time, is minimized. However when *M* at t_1 is different from M' at t_2, and we need to carry out some *swap* operations, these *swap* operations will incur some overhead, e.g., transferring client's data among servers. Therefore, we introduce a threshold T_S such that a *swap* operation is conducted only when the resultant reduction in average communication delays exceeds T_S.

To illustrate this idea, let's see the following example. Originally servers S_α and S_β host zone Z_i and Z_j respectively, i.e., $S(Z_i) = S_\alpha$ and $S(Z_j) = S_\beta$. The overall average delay of clients in zone Z_i and clients in zone Z_j before the *swap* operation will be:

$$\begin{aligned} &T_A(Z_i \cup Z_j) \\ &= \frac{\sum_{C_x \in Z_i} d(C_x, S_\alpha) + \sum_{C_y \in Z_j} d(C_y, S_\beta)}{\| Z_i \| + \| Z_j \|} \qquad (3) \\ &= \frac{T(Z_i, S_\alpha) + T(Z_j, S_\beta)}{\| Z_i \| + \| Z_j \|} \end{aligned}$$

After the *swap* operation the total average delay of clients in zone Z_i and zone Z_j will be:

$$\tilde{T}_A(Z_i \cup Z_j) = \frac{\sum_{C_x \in Z_i} d(C_x, S_\beta) + \sum_{C_y \in Z_j} d(C_y, S_\alpha)}{\| Z_i \| + \| Z_j \|} = \frac{T(Z_i, S_\beta) + T(Z_j, S_\alpha)}{\| Z_i \| + \| Z_j \|} \quad (4)$$

So the condition for this swap operation is:

$$T_{reduced}(Z_i, Z_j) = T_A(Z_i \cup Z_j) - \tilde{T}_A(Z_i \cup Z_j) \geq T_S \quad (5)$$

Whenever there exists such a pair (Z_i, Z_j) satisfying the above condition, we will perform the *swap* operation.

3.2 Simple Swap Algorithm (SA)

```
while (TRUE) do
    L ← ∅;
    foreach pair $(Z_i, Z_j)$ do
        if $T_{reduced}(Z_i, Z_j) \geq T_S$ then
            $L \leftarrow L + \{(Z_i, Z_j)\}$;
        end
    end
    if (L is not empty) then
        Find the pair $(Z_1^*, Z_2^*)$ in L which
        maximizes $T_{reduced}$;
        swap$(S(Z_1^*), S(Z_2^*))$;
    else
        break;
    end
end
```

Figure 4: Swap algorithm

Our swap algorithm checks all pair of zones (Z_i, Z_j) satisfying inequality (5), then finds the pair (Z_1^*, Z_2^*) which maximizes $T_{reduced}$ among such pairs and perform the *swap* operation between (Z_1^*, Z_2^*) until no more pair can be found which satisfies inequality (5).

One important consideration of the algorithm is how to determine T_S. If T_S is too big, the algorithm may not be effective. On the other hand, if T_S is too small, the *swap* operations will be carried out too frequently and the overhead increases accordingly. Therefore, the value of T_S should be carefully chosen. We will explain this in the next section.

3.3 Refined Swap Algorithm (RSA)

To reduce the number of swap operations, we have developed a refined *swap* algorithm.

First, let's introduce some concepts and symbols.

Theorem 3.1 *Any permutation can be uniquely represented as a product of disjoint cycles* [5].

Definition 3.1 *Cyclic Migration* (*CM*): *client migration according to a permutation cycle.*

```
Permutation $P \leftarrow I$;
while (TRUE) do
    L ← ∅;
    foreach pair $(Z_i, Z_j)$ do
        if $T_{reduced}(Z_i, Z_j) \geq T_S$ then
            $L \leftarrow L + \{(Z_i, Z_j)\}$;
        end
    end
    if (L is not empty) then
        Find the pair $(Z_1^*, Z_2^*)$ in L which
        maximizes $T_{reduced}$;
        $P \leftarrow P \times (S(Z_1^*), S(Z_2^*))$;
        Mark $S(Z_1^*)$ as the hosting server of $Z_2^*$
        and $S(Z_2^*)$ as the hosting server of $Z_1^*$;
    else
        break;
    end
end
if $(P \neq I)$ then
    Decompose P to the product of disjoint
    cycles, i.e., $P = \prod_i P_i$;
    foreach $P_i$ do
        Perform $CM(P_i)$;
    end
end
```

Figure 5: Refined swap algorithm

For example the permutation cycle *PC*=(1,2,3), cyclic migration *CM*(*PC*) means servers S_1, S_2 and S_3 will host the zones which are currently hosted by servers S_2, S_3, S_1 respectively before the migration operation. To achieve this, migrations of all clients originally hosted by S_1 to S_3, S_2 to S_1 and S_3 to S_2 are carried out concurrently.

To illustrate the advantage of refined swap algorithm, let's take the above permutation cycle *PC* = (1, 2, 3) as an example. To achieve this by *swap* operation, at least two *swap* operations are needed, e.g., *swap*(S_1, S_2) and *swap*(S_2, S_3). Each *swap* operation contains two zone migrations. Thus, to achieve (1, 2, 3), simple *swap* operation costs four zone migrations while cyclic migration costs only three. Generally, the longer the permutation cycle, the more migration operations the refined swap algorithm saves.

4 Performance Evaluation

4.1 Considerations

We evaluate our proposed algorithms via simulations. In the simulations, we use a two-level, Internet-like topology generated by the BRITE Internet Topology Generator [2]. The topology consists of 2000 nodes and 4000 links. The upper bound delay of each link is uniformly distributed in (0, 30ms). We randomly select nodes in the set of 2000 generated nodes to be servers and clients.

To simulate the movement of avatars, we use the value of *moving distance of one step*/*edge length of the zone* to represent the moving speed of an avatar and we set it to 0.02 in the simulations. The direction of each movement is random and the chance for each avatar to make a move in each step is also random.

Different DVE configurations are used for performance evaluation. A specific DVE configuration is determined by the number of servers, the number of clients and the threshold T_S. We use a notation *number of servers-number of clients-T_S* to denote a DVE configuration. For example, the notation 30s-300c-20 means that the DVE has 30 servers, 300 clients, and T_S is set to 20ms. Results presented here are obtained by averaging the results of 50 simulation runs.

4.2 Results and Analysis

As the avatars move in the virtual environment, any static zone-server assignment may ultimately perform as bad as random assignment. Here random assignment refers to the static assignment that randomly assigns zones to servers. In the experiments, random assignment is used to make comparison with our dynamic zone-server assignment algorithm. To show the effect of swap algorithm, the initial assignments of all dynamic zone-server assignments are randomly chosen.

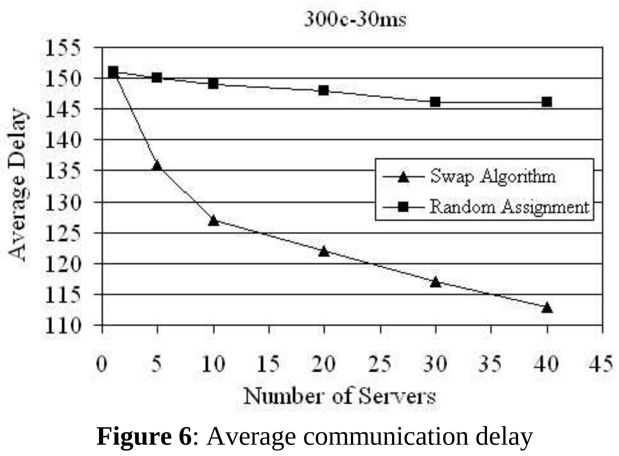

Figure 6: Average communication delay

Figure 6 shows the performance of *swap* algorithm for DVE configuration 300c-30ms with different number of servers. From Figure 6, it is observed that the average communication delay for the overall system is reduced by up to (146-113)/146 = 23.3% comparing to random assignment. The more servers we have, the better result we get. This can be explained by the fact that the chance to find pair (Z_1^*, Z_2^*) is higher when DVE has more servers.

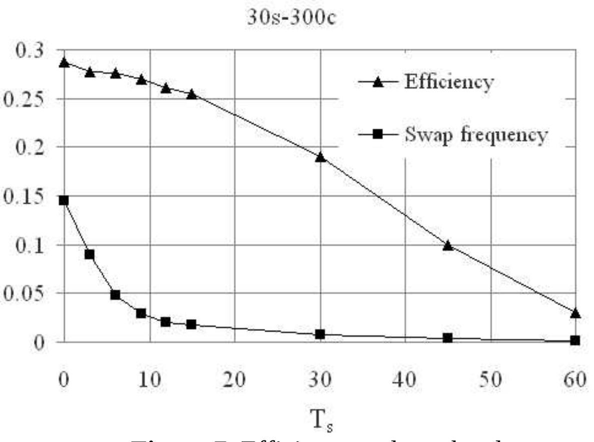

Figure 7: Efficiency and overhead

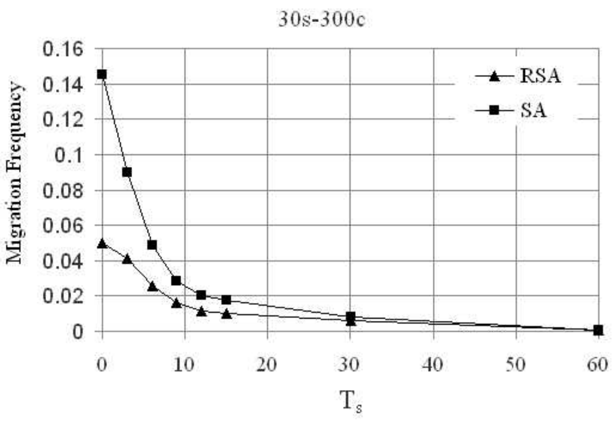

Figure 8: Migration frequency of RSA and SA

Figure 7 shows the effect of threshold T_S with DVE configuration 30s-300c and the efficiency is defined as the percentage of communication delay reduced over random assignment. For example, the average communication delay of all clients is 200ms for random assignment and 120ms for swap algorithm, the efficiency is calculated as (200-120)/200 = 40%. Swap frequency is defined as *number of clients migrated per second*. From Figure 7, it can be seen that both the efficiency and swap frequency reduce as T_S increases. This can be explained by the fact that the chance to find (Z_1^*, Z_2^*) reduces as T_S increases and less swap operation needs be done. For real applications, the overhead brought by swap operation will be proportional to the swap frequency. To seek a balance between efficiency of swap algorithm and the overhead, we can tune T_S to a suitable value. From our experience, normally the value of T_S can be chosen to be around 10% of the average communication delay, where the efficiency is above 25% and the swap frequency is as low as 0.02.

Figure 8 shows the effect of refined swap algorithm with DVE configuration 30s-300c and different T_S. It can be seen that the refined swap algorithm reduces the overhead up to (0.15-0.05)/0.15 = 66.7% comparing to the simple swap algorithm.

5 Conclusions

Due to the huge resource demands and the real-time requirement of large-scale DVEs, a geographically distributed server architecture is usually needed to support such applications. In this architecture, multiple servers are geographically distributed over the network, and the large virtual world is partitioned into distinct zones to distribute load among the servers. In this paper we have proposed a dynamic zone-server assignment approach to enhancing the interactivity of large-scale DVEs by carrying out re-arrangement. We gave the conditions on when to carry out re-arrangement and the way how to do it. Simulation studies with realistic models have shown that our approach is very effective in enhancing the interactivity of large-scale DVEs.

For future work, it would be very interesting to extend the idea to the DVEs in which each server hosts multiple zones.

References

[1] Bauer, D., Rooney, S., Scotton, P., “Network Infrastructure for Massively Distributed Games”, in Proc. of the First Workshop on Network and System Support for Games (NetGames 2002), Braunschweig, Germany, April 16-17, 2002

[2] BRITE Topology Generator, avaiable at http://www.cs.bu.edu/brite.

[3] Everquest, available at: http://www.everquest.com

[4] Fujikawa, K., Hori, M., Shimojo, S., Miyahara, H., “A Server Selection Method Based on Communication Delay and Communication Frequency among Users for Networked Virtual Environments”, Asian Computer Science Conference (ASIAN 2002), LNCS 2550, pp.125-139

[5] Hertein, I., *Abstract Algebra*, Third Edition, John Wiley & Sons, 1999

[6] IDC, available at: http://www.idc.com

[7] Mauve, M., Fischer, S., Widmer, J., “A Generic Proxy System for Networked Computer Games”, in Proc. of the First Workshop on Network and System Support for Games (NetGames 2002), Braunschweig, Germany, April 16-17, 2002

[8] Singhal, S., Zyda, M., *Networked Virtual Environments: Design and Implementation*, Addison-Wesley, Reading, MA, 1999.

[9] Ta, D., Zhou, S., “Efficient Client-to-Server Assignments in Distributed Virtual Environments”, in Proc. of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rhodes Island, Greece, April 25-29, 2006

[10] Ta, D., Zhou, S., Shen H., “Greedy Algorithms for Client Assignment in Large-Scale Distributed Virtual Environments”, in Proc. of the 20th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS), Raffles Hotel, Singapore, May 23-26, 2006

[11] Zhou, S., Cai, W., Lee, B., Turner, S., “Time-space Consistency in Large-scale Distributed Virtual Environments”, ACM Transactions on Modeling and Computer Simulation (TOMACS), vol. 14, no. 1, 2004, pp.31-47.