A Scalable Synchronization Technique for Distributed Virtual Environments Based on Networked-Server Architectures*

P. Morillo, J.M. Orduña
Departamento de Informática
Universidad de Valencia
SPAIN
Email: juan.orduna@uv.es

J. Duato
DISCA
Universidad Politécnica de Valencia
SPAIN
Email: jduato@gap.upv.es

Abstract

In recent years, large scale distributed virtual environments have become a major trend in distributed applications, mainly due to the enormous popularity of multi-player online games in the entertainment industry. Thus, scalability has become an essential issue for these highly interactive systems. In this paper, we propose a new synchronization technique for those distributed virtual environments that are based on networked-server architectures. Unlike other methods described in the literature, the proposed technique takes into account the updating messages exchanged by avatars, thus releasing the servers from updating the location of such avatars when synchronizing the state of the system. As a result, the communications required for synchronization are greatly reduced, and this method results more scalable. Also, these communications are distributed along the whole synchronization period, in order to reduce workload peaks. Performance evaluation results show that the proposed approach significantly reduces the percentage of CPU utilization in the servers when compared with other existing methods, therefore supporting a higher number of avatars. Additionally, the system response time is reduced accordingly.

1. Introduction

In recent years, large scale distributed virtual environments (DVEs) have become a major trend in distributed applications, mainly due to the enormous popularity of multi-player online games in the entertainment industry. These highly interactive systems simulate a 3-D virtual world where multiple users share the same scenario. Each user is represented in the shared virtual environment by an entity called *avatar*, whose state is controlled by the user through a client computer. The system renders the images of the virtual world that each user would see if he was located at that point in the virtual environment. Hundreds and even thousands of client computers can be simultaneously connected to the DVE through different networks, and even through Internet.

Architectures based on networked servers have been during last years the major standard for DVE systems [23, 13, 28, 14, 8]. In these architectures, the control of the simulation relies on several interconnected servers. Client computers are assigned to one of the servers in the system. In these architectures, when a client computer modifies the state (usually the position) of an avatar, it also sends an updating message to its server, which in turn must propagate this message to other servers and clients. However, servers in the DVE system must schedule the sending of information related to the geometry of the scene, perform positional updates of avatars, and transfer control information among different clients. Therefore, each new avatar represents an increase not only in the computational requirements of the application but also in the amount of network traffic. Due to this increase, scalability is an essential issue when designing DVE systems based on networked-server architectures, particularly if they should support multi-player online games (MOGs). For example, when the number of connected clients increases, the number of update messages exchanged by avatars must be limited in order to avoid a message outburst. In this sense, concepts like areas of influence (AOI) [23], locales [1] or auras [9] have been proposed for limiting the number of neighboring avatars that a given avatar must communicate with. All these concepts define a neighborhood area for avatars, in such a way that a given client computer controlling a given avatar i must notify all the movements of i (by sending an updating message) only to the client computers that control the avatars located in the neighborhood of avatar i. The avatars in the AOI of avatar i are

*This paper is supported by the Spanish MEC under Grant TIC2003-08154-C06-04

denoted as *neighbor avatars* of avatar i. Figure 1 shows an example of a DVE system whose architecture is based on networked servers. In this example, the AOI of each avatar is represented as a circumference.

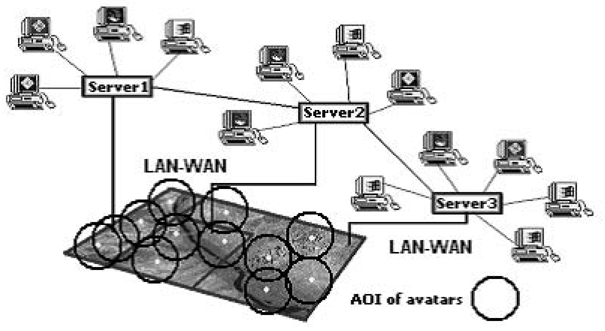

Figure 1. Example of a DVE based on networked servers

In order to provide scalability and allowing large scale DVE systems, alternative schemes like peer-to-peer (P2P) architectures have been proposed. Several online games based on P2P architectures have been designed [17, 16, 6]. Nevertheless, P2P architectures must face the *awareness problem*. This problem consists of ensuring that each avatar is aware of all the avatars in its neighborhood [25]. Solving the awareness problem is a necessary condition to provide a consistent view of the environment to each participant. Effectively, if two neighbor avatars are not aware of such neighborhood, they will not exchange messages about their movements and/or changes, and therefore they will not have the same vision of the shared environment. Thus, providing awareness to all the avatars is a necessary condition to provide consistency (as defined in [30, 4, 22, 24]). However, it is not a sufficient condition. Even when using a awareness method that determines at each moment which other avatars must each avatar exchange messages with, time-space inconsistencies can arise among different avatars because of clock drifts and/or network delays [30]. Awareness is crucial for MOGs, since otherwise abnormal situations could happen. For example, a user provided with a non-coherent 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 DVEs, the awareness problem has been efficiently solved by using a *synchronization technique* [23]. It consists of the existing servers periodically exchanging information among them about the location of all avatars. Each avatar reports about its movements (by sending a message) to the server where it is assigned to, and the server can easily decide which avatars should be the destinations of that message by using a criterion of distance. However, the synchronization technique generates a significant overhead, since all servers must report to the rest of the servers about the location of avatars assigned to them. If the synchronization technique could be implemented in a more scalable way, DVE systems based on networked servers could support a larger number of avatars, improving their scalability.

In this paper, we propose a new, scalable synchronization technique for networked-server DVE systems. Unlike other methods described in the literature, the proposed method takes into account the updating messages exchanged by avatars, thus releasing the servers from updating the location of such avatars at the moment of synchronizing the state of the system. As a result, the communications required for synchronizing the servers are greatly reduced, and this method results more scalable. Performance evaluation results show that the proposed approach significantly reduces the percentage of CPU utilization in the servers when compared with other existing methods, therefore supporting a higher number of avatars. Also, we propose to distribute the avatar updates in different messages along the whole synchronization period, in order to avoid instantaneous CPU saturations of the servers in the system.

The rest of the paper is organized as follows: Section 2 analyzes the existing synchronization techniques for DVE systems based on networked-server architectures. Section 3 describes the proposed technique and how it improves the weaknesses of the existing proposals. Next, Section 4 presents the performance evaluation of the proposed method. Finally, Section 5 presents some concluding remarks.

2. Background

One of the key issues in DVEs has been time-space-consistency [30, 4, 22, 24]. Although providing a total time-space consistency seems impossible due to the inherently distributed nature of DVEs, the *awareness* problem [25] must be fully solved in order to warranty a coherent ordering of events for avatars. The awareness problem can be easily solved in DVE systems based on networked servers, provided that a synchronization technique allows all the servers to exchange the information about the location of those avatars assigned to them. Once each server knows the location of all avatars, it can compute which neighbors must each avatar exchange messages with (that is, it can compute the awareness of each avatar).

Several different synchronization techniques and approaches have been proposed for providing awareness in DVE systems based on networked servers. Some of these proposals are based on *conservative algorithms* [5, 6]. In this kind of techniques, avatars are not allowed to advance their simulation clock until all avatars have finished their computations for the current time period. However, these conservative algorithms perform poorly in fast-paced games, that are latency sensitive applications where frequent updates are important for avatars to achieve their goals. Other synchronization techniques are based on *optimistic algorithms* [26, 27, 2]. In these techniques avatars can optimistically generate new events before they ensure that no earlier events could arrive. However, when such situation occurs then events must be backtracked until inconsistencies are repaired. Last years, different approaches using region-based assignments of avatars to the existing servers have appeared [29, 21]. In these approaches, avatars are assigned to the server managing the region of the virtual world where these avatars are located. Therefore, a given server managing a given region knows the location of all the avatars in this region, and it does not need to know the location of the avatars located in the other regions. Avatars located near the border between two regions must report to the servers controlling both regions (shadow objects [29] or neighborhood regions [21]). In this way, both servers can compute at every moment the awareness of such avatars. Although these approaches do not need a synchronization technique, they lack a proper workload balancing scheme that greatly limits the throughput (the number of avatars that the system can simultaneously support while offering a reasonable response time). Therefore, these systems provide limited scalability, particularly when avatars move following non-uniform movement patterns [20, 18]. Therefore, region-based approaches do not provide the scalability needed for online multiuser games.

3. A New Synchronization Technique

We propose a new synchronization technique for networked-server DVE systems. In order to achieve a scalable technique, both the communication and the computation requirements of such method should be reduced. Therefore, the idea is to take advantage of the messages exchanged by avatars when moving to reduce the amount of information in the messages generated by the synchronization technique. Since the information in these messages must be processed at a given time by a single server, less information means less computation requirements. As next section shows, reducing the amount of this information avoids momentary saturations of the servers, as well as the impact of such saturations on the whole system [19, 20].

The synchronization technique proposed in [23] consists of each server periodically propagating to the rest of the servers the location updates of all the avatars assigned to that server. Let s be the number of existing servers in the system and a the number of existing avatars in the system. Every T milliseconds, this synchronization technique generates N updates, where

$$N = s\,\frac{a}{s}\,(s-1) = a(s-1)$$

Regardless of the format and encapsulation of such updates in messages of different sizes, in order to improve the scalability of the synchronization technique N must be reduced as much as possible.

In our synchronization technique the servers monitor all the messages sent by any avatar and traversing them, and they use these messages for updating the location of the sending avatars between the synchronization intervals. Each server s_i maintains a counter associated to each avatar a in the virtual world not assigned to s_i. This counter is set to 0 at the beginning of the simulation and each time that a message sent by a arrives or traverses s_i. The proposed technique consists of the synchronization technique described in [23], but now adding a process that wakes up every T_c ms. and increments all the counters ($T_c < T$). When the interval T finishes, each server s_i only asks the other servers to update (by sending back a message) the location of those avatars whose associated counter in s_i is greater than a greater value ($MAXTICK$). It is important to notice that T must not be too high when tuning the values for T, T_c and $MAXTICK$, in order to avoid time-space inconsistencies. On other hand, $T_c \times MAXTICK$ must be lower than T, in order to properly detect which are the avatars whose information has not been updated. Additionally, $2 \times T_c \times MAXTICK$ must be greater than T, in order to avoid the sending of duplicated messages to avatars whose information is not updated. Taking into account these conditions, the formula shown above becomes an upper limit in the proposed method. The actual value for N will depend on the number of avatars not assigned to s_i whose messages traverses s_i.

Additionally, the proposed technique avoids the momentary saturation of servers due to the processing of huge messages. Instead of each server sending a huge message to the rest of the servers containing the location updates of all its avatars, in our technique each server iteratively sends several smaller messages with the location updates of different avatars. This uniform distribution of messages over time avoids momentary CPU saturations in any server, keeping the response time to avatars within acceptable limits [20] all the time.

4. Performance Evaluation

In this section, we present the performance evaluation of the synchronization technique described above, evaluating different DVE systems in order to measure the actual improvement that the proposed method can provide.

We propose the evaluation of generic DVE systems by simulation. The evaluation methodology used is based on the main standards for modeling collaborative virtual environments, such as FIPA [3], DIS [10] and HLA [11]. We have developed a simulation tool that models the behavior of a generic DVE system composed of several interconnected servers, and we have performed experimental studies to evaluate the performance of the proposed technique. Following the approach specified in FIPA and HLA standards, one of the servers acts as the main server (called *Agent Name Service* [3] or *Federation Manager* [11]) and manages the whole system. The main server also maintains a partitioning file for assigning a given server to each new avatar. In this way, once we set the network address and the port number where the main server is listening, avatars can join the simulation through this main server, that assigns each new avatar to one of the servers in the system. At this point, the new avatar must connect with the assigned server in order to start the simulation.

In each simulation, all avatars sharing the same AOI must communicate among themselves to communicate both their position in the 3D virtual world and also any change in the state of the elements in that AOI. For evaluation purposes, each client only simulates a given rate of avatar movements through the virtual world, and assumes that no changes are produced in any element of the AOI. This simplifying assumption reduces the system workload, but does not change the behavior of the system. The message structure used for communicating avatar movements is the *Avatar Data Unit (ADU)* specified by DIS [10]. 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 two seconds). Each time an avatar performs a movement, it (the client computer controlling that avatar) communicates that movement to its server by sending a message with a timestamp. That server must then propagate that message to all the avatars (to the corresponding client computers) in the same AOI of the sender avatar. When that notification arrives to these avatars, they return an ACK message to the sending avatar. When all the ACK messages corresponding to a given message m arrives to the sender avatar i of that message, then avatar i can compute the average round-trip delay for message m. In this way, clock skewing is avoided when computing system latency (since an ACK message arrives at the client computer that controls the sending avatar, the same clock is used for both the initial and final timestamp). This is the main reason for measuring round-trip delays instead of latencies in a distributed system. When a simulation finishes, each avatar can compute the average round-trip delay for all its messages. We denote this average value for a given avatar i as asr_i, for the *average system response* for avatar i. Additionally, we have implemented the synchronization technique described in the previous section, in such a way that servers also exchange messages. Since the simulation of a DVE system can be considered as a stochastic procedures, each numerical value in the tables shown in this section has been computed as the average value of 100 different simulations under identical conditions. The standard deviation of the different values has not been higher than 20% of the value shown in the table in any case.

In order to evaluate the performance of a DVE system, usually three different avatar distributions in the virtual world have been suggested in the literature: uniform, skewed and clustered [12]. The reason for using different distributions is that they generate a different workload. Figure 2 shows an example of these avatar distributions in a 2-D virtual world. In this figure, the virtual world is a square and avatars are represented as black dots. For all the distributions, the movement pattern of avatars used in the simulations has been Changing Circular Pattern (CCP) [21]. CCP considers that all avatars in the virtual world move randomly around the virtual scene following circular trajectories. Therefore, when simulation finishes avatars are located in the virtual world following the same distribution they had at the beginning of the simulation.

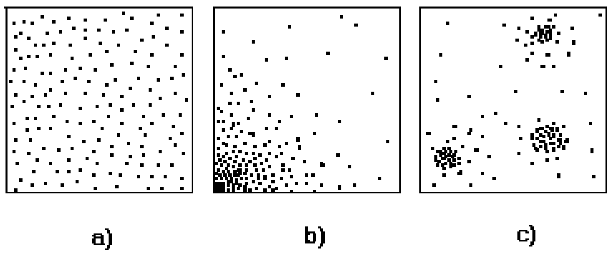

Figure 2. Distributions of avatars in a 2-D virtual world: a) Uniform b) Skewed c) Clustered.

For comparison purposes, we have simulated the synchronization technique proposed in the previous section, and the synchronization technique proposed in [23]. In order to measure the performance of the proposed technique regardless of the message size (each message can contain either the location updates of all the avatars assigned to a server or only a portion of such updates), we have measured the number of updates that servers have performed during a simulation when using both awareness techniques. In all the results shown in this section, we have denoted $T1$ to the synchronization technique proposed in [23], and $T2$ to the synchronization technique proposed in this paper. According to practical observations on the evaluated DVE system, we have set the the $T2$ parameters to the following values: $T_c = 750$ms., $T = 5000$ms. and $MAXTICK = 5$.

Table 1 shows the number of updates performed in each technique for the three considered distributions of avatars in the virtual world, and for different DVE configurations. Each column shows the number of updates for a different distribution of avatars in the virtual world. The table contains four subtables, one for each DVE configuration. In each subtable, the first row shows the number of both avatars and servers composing that configuration. The second row shows the considered distributions of avatars in the virtual world, and the last two rows show the number of updates obtained with each of the considered techniques.

Table 1. Number of updates performed by the awareness methods

	350 Avatars, 5 Servers		
	Uniform	Skewed	Clustered
T1	1400	1400	1400
T2	1144	794	914
	600 Avatars, 5 Servers		
	Uniform	Skewed	Clustered
T1	2400	2400	2400
T2	2042	1282	1428
	600 Avatars, 9 Servers		
	Uniform	Skewed	Clustered
T1	4800	4800	4800
T2	4125	2740	3381
	900 Avatars, 9 Servers		
	Uniform	Skewed	Clustered
T1	7200	7200	7200
T2	6142	4070	4488

Table 1 shows that the synchronization technique proposed in [23] requires the same number of updates for a given DVE configuration, regardless of the distribution of avatars in the virtual world. Effectively, the four rows for the $T1$ results show the same value in all the columns. The reason for this behavior is that, since this method does not use the updating messages exchanged by avatars between two consecutive synchronization periods, it always generates the same number of updates (updates for all the avatars). Thus, the number of updates in this technique exclusively depends on the number of avatars in the DVE configuration. On the contrary, the number of updates generated by $T2$ technique significantly varies with the distribution of avatars in the virtual world. Depending on the grouping of avatars in the virtual world, a different number of updates are exchanged by avatars between synchronization periods (there are more avatars in each AOI as more grouped they are). Therefore, this technique provides different results for different distributions of avatars. Thus, $T2$ technique provides the best results for the skewed distribution of avatars, and the worst ones for the uniform distribution of avatars. Moreover, these results suggest that if non-uniform movement patterns of avatars like Hot-Points-ALL (HPA) [7] or Hot-Point-Near (HPN) [15] had been used, the performance differences between the two considered techniques would have been even greater, since in these patterns avatars tend to head for certain ”hot points” of the virtual world (they tend to group as the simulation proceeds). It can be clearly seen that even for the worst case (uniform distribution of avatars) $T2$ technique generates a significantly lower number of updates.

Although the results shown in Table 1 show a good performance of the proposed technique, it does not prove that it can have an effect on the system response time offered to the client computers. In order to prove that, we have simulated different DVE configurations and we have measured both the percentage of CPU utilization in each server and the average system response time offered to avatars (as defined in [20]). We have simulated the same four DVE configurations under the three distributions of avatars. However, since the purpose of the proposed technique is to add scalability to the DVE system, we only show below the results for the two DVE configurations supporting more avatars per server, for the sake of shortness.

Table 2 shows the results for the DVE configuration of 600 avatars and 5 servers. This table is composed of three subtables, one for each of the considered distributions of avatars. In each subtable, the first four rows show the average percentage CPU utilization of a given server in the DVE system, and also the average system response (ASR) in milliseconds for the avatars assigned to that server when using both awareness techniques $T1$ and $T2$. The last row in each sub-table shows the average values for the whole DVE system.

Table 2 shows that for the uniform distribution of avatars the workload generated in the simulation is relatively small, and therefore the average CPU utilization is around 50%. Even under these low workload conditions, the effects of proposed technique are significative. However, the proposed technique provides the best results for the other two distributions (the ones generating the highest workload). Effectively, for the clustered distribution of avatars the proposed technique is able to reduce the average value of the ASR in a 10% with respect to the value obtained with $T1$ technique. The reason for this improvement is that when the DVE system is close to saturation (an average CPU utilization of 91%) the CPU saving allowed by $T2$ (less updates in the synchronization technique mean a lower CPU workload, and therefore a lower CPU utilization) avoids reaching saturation, reducing the average ASR accordingly [20].

In order to show the improvement that the proposed technique can achieve in a larger DVE configuration, Table 3 shows the results for a configuration of 900 avatars and 9 servers. Again, each row shows the results obtained in each server and the last row shows the average results for all the

Table 2. Results for a medium-sized DVE

Uniform	T1		T2	
	CPU %	ASR	CPU %	ASR
S0	52	275	46	271
S1	48	239	40	261
S2	53	295	49	258
S3	50	261	51	256
S4	51	264	39	231
Av.	51	267	45	255

Skewed	T1		T2	
	CPU %	ASR	CPU %	ASR
S0	80	331	68	306
S1	84	338	72	284
S2	86	320	75	321
S3	79	327	66	345
S4	88	279	71	237
Av.	83	319	70	298

Clustered	T1		T2	
	CPU %	ASR	CPU %	ASR
S0	91	312	71	308
S1	98	406	68	351
S2	88	317	82	290
S3	90	321	70	261
S4	86	341	66	308
Av.	91	339	71	303

Table 3. Results for a large DVE

Uniform	T1		T2	
	CPU %	ASR	CPU %	ASR
S0	48	284	51	245
S1	54	246	49	256
S2	56	261	36	236
S3	53	238	42	210
S4	46	248	38	281
S5	42	260	40	253
S6	50	239	44	237
S7	48	258	33	249
S8	48	298	39	217
Av.	49	259	41	236

Skewed	T1		T2	
	CPU %	ASR	CPU %	ASR
S0	79	331	76	298
S1	81	325	75	305
S2	85	309	59	261
S3	86	387	65	253
S4	77	368	66	259
S5	75	296	70	261
S6	85	274	66	294
S7	80	254	59	284
S8	84	265	72	321
Av.	81	312	68	281

Clustered	T1		T2	
	CPU %	ASR	CPU %	ASR
S0	90	312	82	317
S1	95	321	86	301
S2	88	299	66	274
S3	86	335	63	261
S4	94	421	71	291
S5	92	396	66	269
S6	88	331	69	274
S7	84	299	86	256
S8	86	312	61	306
Av.	89	336	72	283

servers. These results shows that the proposed technique can achieve the best performance improvements for the distributions of avatars generating the highest workload, since in these cases the proposed technique helps to avoid the system saturation. Concretely, for the case of the clustered distribution $T2$ technique manage to reduce the percentage of CPU utilization from 89% to 72%, and it reduces the average ASR from 336 to 283. That is, $T2$ technique reduces the CPU utilization, becoming more scalable than $T1$ technique (supporting more avatars without reaching system saturation) and providing lower latencies than $T1$ technique accordingly.

When comparing Table 2 and Table 3 we can see that the larger performance differences between $T1$ and $T2$ techniques are obtained for the DVE configuration of 900 avatars and 5 servers. Therefore, these results suggest that the larger the DVE system is, the larger performance improvement provides the proposed technique. That is, they suggest that the proposed technique properly scales with the size of the DVE system, validating it as a scalable synchronization technique for DVE systems based on networked server architectures.

Additionally, we have measured the effect of distributing the updates generated by the synchronization technique in a uniform way. Concretely, we have sent a message for each avatar update. However, we have uniformly distributed the sending of these messages along the synchronization period T. In this way, we have eliminated workload peaks due to the synchronization technique. Figures 3 and 4 show the results (in term of instantaneous percentage of CPU utilization) for a given server (S0), although the results for the rest of the servers were very similar. These results correspond to a DVE configuration of 900 avatars and 9 servers when using $T1$ technique. That is, each server must process 800 avatar updates in each synchronization period (5000 ms.). Concretely, Figure 3 shows the CPU utilization along the simulation time when all servers send their updates encapsulated in a single message to the rest of the servers. Figure 4 shows the CPU utilization when servers uniformly distribute their updates in different, short messages along the whole synchronization period.

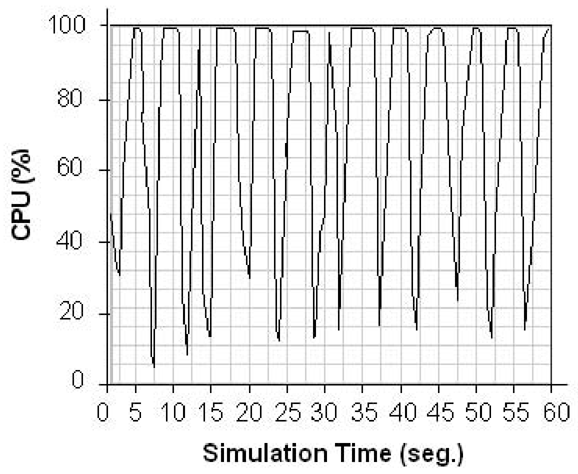

Figure 3. Instantaneous CPU utilization when grouping updates in a single message

Figure 3 shows that the CPU utilization alternates periods of saturation with periods of relatively low load. The saturation periods are due to the simultaneous processing of several, huge messages from the rest of the servers. These messages contain the updates of all the avatars assigned to each server (we are considering $T1$ technique in both figures). Since all the servers send their messages at the beginning of the synchronization period, the receiving of such messages also overlaps in time. This overlapping produces the periodical, instantaneous saturations (utilization peaks) shown in Figure 3.

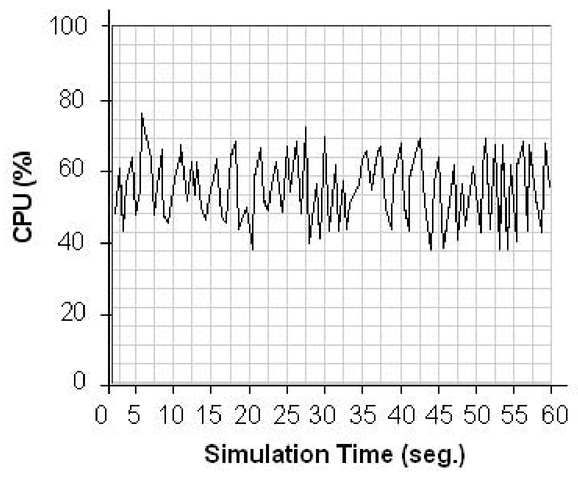

Figure 4. Instantaneous CPU utilization when distributing updates in different messages

Figure 4 also shows a plot with a sawtooth shape. However, the peaks in this plot only reach a percentage of CPU utilization around 75%. That is, the servers do not reach saturation at any time. Therefore, these results show how the same workload supported by servers can be easily made uniform if each avatar update is encapsulated in a single, short message, and these messages are distributed along the synchronization period. Since this temporal distributions of messages can totally avoid the CPU saturation periods, it can provide significant reductions in the response time provided to avatars [20].

It could be thought that the sparse sending of the different updates could have a negative effect on the graphical representation of those avatars whose updates are sent in last places. However, this potential effect can be easily avoided by sending the updates in the same order during subsequent synchronization periods. In this way, the negative effects of this temporal distribution are limited to the first synchronization period, and the same T period is provided for all avatars in subsequent periods.

5. Conclusions

In this paper, we have proposed a scalable synchronization technique for networked-servers DVE systems. Unlike the currently existing techniques, the proposed method takes into account the updating messages exchanged by avatars, thus releasing the servers from updating the location of such avatars when synchronizing the state of the system. As a result, the communications required for synchronization are greatly reduced.

Performance evaluation results show that the proposed technique significantly reduces the number of avatar updates in regard to similar existing techniques. Moreover, the results shows that the proposed technique provides larger reductions when avatars move following non-uniform movement patterns, becoming more efficient as the workload generated by avatars increases.

Results also show that the reduction in the number of updates has a significant effect on the CPU utilization of the servers in the system, in such a way that a DVE system using the proposed technique supports more avatars and provides lower latencies than the same DVE system using the existing techniques. The performance improvement achieved by the proposed technique increases as the greater workload is generated by avatars, since it proportional reduces the number of updates. Therefore, these results validate the proposed technique as a scalable one.

Additionally, we have proposed the encapsulation of each avatar update in single message, and the temporal distribution of such messages along the whole synchronization period. Such distribution makes uniform the workload generated by the synchronization technique, therefore avoiding instantaneous saturations of the server CPU(s) and reducing system latency accordingly.

References

[1] D. Anderson, J. Barrus, and J. Howard. Building multi-user interactive multimedia environments at merl. *IEEE Multimedia*, 2(4), 1995.

[2] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin. An efficient synchronization mechanism for mirrored game architectures. *Kluwer Multimedia Tools and Applications*, 23(1), 2004.

[3] FIPA. Fipa agent management specification, 2000. Available at http://www.fipa.org/specs/fipa00023/.

[4] R. M. Fujimoto and R. Weatherly. Time management in the dod high level architecture. In *Proceedings tenth Workshop on Parallel and Distributed Simulation*, pages 60–67, 1996.

[5] T. Funkhouser. Network topologies for scalable multi-user virtual environments. In *Proc. IEEE Virtual Reality Annual International Symposium*, pages 222–228, 1996.

[6] L. Gautier and C. Diot. Design and evaluation of mimaze, a multi-player game on the internet. In *Proceedings of IEEE Multimedia Systems Conference*, 1998.

[7] C. Greenhalgh. Analysing movement and world transitions in virtual reality tele-conferencing. In *European Conference on Computer Supported Cooperative Work (ECSCW 97)*, page 313, 1997.

[8] C. Greenhalgh, A. Bullock, E. Fr¿on, D. Llyod, and A. Steed. Making networked virtual environments work. *Presence: Teleoperators and Virtual Environments*, 10(2):142–159, 2001.

[9] F. C. Greenhlagh. Awareness-based communication management in massive systems. *Distributed Systems Engineering*, 5(3):129, 1998.

[10] IEEE. *1278.1 IEEE Standard for Distributed Interactive Simulation-Application Protocols (ANSI)*, 1997.

[11] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice-Hall PTR, 1999.

[12] J. C. Lui and M. Chan. An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Trans. Parallel and Distributed Systems*, 13, 2002.

[13] J. C. Lui, M. Chan, and K. Oldfield. Dynamic partitioning for a distributed virtual environment. Technical report, Department of Computer Science. Chinese University of Hong Kong, 1998.

[14] M. R. Macedonia. A taxonomy for networked virtual environments. *IEEE Multimedia*, 4(1):48–56, 1997.

[15] 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 & Robotic Control*, 1(1):11–26, 1999.

[16] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical report, Technical Report HPL-2002-57, HP Laboratories, Palo Alto, 2002.

[17] S. Mooney and B. Games. *Battlezone: Official Strategy Guide*. BradyGame Publisher, 1998.

[18] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. A fine-grain method for solving the partitioning problem in distributed virtual environment systems. In *Proc. of Intl. Conf. on Parallel and Distributed Computing and Systems (PDCS'04)*, pages 292–297. IASTED, ACTA Press, 2004. Best paper award in the area of load balancing.

[19] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. On the characterization of distributed virtual environment systems. In *Euro-Par' 2003 - Lecture Notes in Computer Science 2790*, pages 1190–1198. ACM, Springer-Verlag, 2003.

[20] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. Improving the performance of distributed virtual environment systems. *IEEE Transactions on Parallel and Distributed Systems*, 16(7):637–649, 2005.

[21] B. Ng, A. Si, R. W. Lau, and F. W. Li. A multi-server architecture for distributed virtual walkthrough. In *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 163–170, New York, NY, USA, 2002. ACM Press.

[22] D. Roberts and R. Wolff. Controlling consistency within collaborative virtual environments. In *Proceedings of IEEE Symposium on Distributed Simulation and Real-Time Applications (DSRT'04)*, pages 46–52, 2004.

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

[24] J. Smed, T. Kaukoranta, and H. Hakonen. A review on networking and multiplayer computer games. Technical report, Turku Centre for Computer Science. Tech Report 454., 2002.

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

[26] J. S. Steinman, R. Bagrodia, and D. Jeerson. Breathign time warp. In *Proceedings of Workshop on Parallel and Distributed Simulation*, pages 109–118, 1993.

[27] J. S. Steinman, J. W. Wallace, D. Davani, and D. Elizandro. Scalable distributed military simulation using the speedes object-oriented simulation framework. In *Proceedings of Object-Oriented Simulation Conference (OOS'98)*, pages 3–23, 1998.

[28] P. Tam. Communication cost optimization and analysis in distributed virtual environment. Technical report, Department of Computer Science. Chinese University of Hong Kong, 1998.

[29] J. yao Huang, Y. chang Du, and C.-M. Wang. Design of the server cluster to support avatar migration. In *VR*, pages 7–14, 2003.

[30] S. Zhou, W. Cai, B. 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.