Workload Characterization in Multiplayer Online Games*

P. Morillo[2], J.M. Orduña[1], and M. Fernández[2]

[1] Universidad de Valencia, Departamento de Informática,
Avda. Vicent Andrés Estellés, s/n
46100 - Burjassot (Valencia), Spain
Juan.Orduna@uv.es
[2] Universidad de Valencia, Instituto de Robótica,
Polígono de la Coma s/n
46980 - Paterna (Valencia), Spain

Abstract. In recent years, distributed virtual environments (DVEs) have become a major trend in distributed applications, mainly due to the enormous popularity of multiplayer online games in the entertainment industry. Although the workload generated by avatars in a DVE system has already been characterized, the special features of multiplayer online games make these applications to require a particular workload characterization.
This paper presents the workload characterization of multiplayer online games. This characterization is based on real traces, and it shows that the movement patterns of avatars used to develop optimization techniques for DVE systems can be extrapolated to First Person Shooting networked games. However, the workload that each avatar adds to the game server is higher than the one proposed in the literature.

1 Introduction

The enormous popularity that multiplayer online games have acquired nowadays has allowed a huge expansion of Distributed Virtual Environments (DVEs). 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 the 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 system through different networks, and even through the Internet. Although DVE systems are currently used in many different applications such as civil and military distributed training [1], collaborative design [2] or e-learning [3], the most extended example of DVE systems are commercial, multiplayer online game (MOG) environments [4–7].

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

Different architectures have been proposed in order to efficiently support multiplayer online games: centralized-server architectures [5, 7], networked server architectures [8, 9] and peer-to-peer architectures [10, 11]. Figure 1 shows an example of each one of these architectures. In this example, the virtual world is two-dimensional and avatars are represented as dots. The area of interest (AOI) of a given avatar is represented as a circumference.

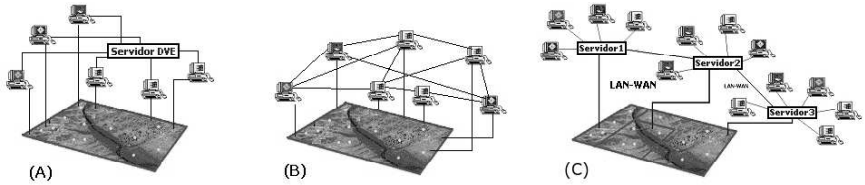

Fig. 1. Different architectures for DVE systems: a) Centralized b) Peer-to-peer c) Networked-server

Regardless of the system architecture, some key issues add complexity to the design of an efficient and scalable DVE system. Some of the following ones have become nowadays an open research topic:

- View Consistency: This problem has been already defined in other computer science fields such as database management [12]. In DVE systems, this problem consists of ensuring that all avatars sharing a virtual space with common objects have the same local vision of them [13–15]. Some proposals have been made to provide consistency [16, 17, 11].
- Message Traffic Reduction: Keeping a low amount of messages allows DVE systems to efficiently scale with the number of avatars in the system. Traditionally, techniques like dead-reckoning described in [8] offered some level of independence to the avatars. With network support, broadcast or multicast solutions [18, 11] decrease the number of messages used to keep a consistent state of the system.
- Partitioning scheme: In networked-server architectures, the problem of efficiently distributing the workload (avatars) among different servers in the system determines the throughput and the quality of service (QoS) provided to users [19, 20].

A common feature in all these research topics is the need of characterizing the user behavior, in order to determine the workload generated to the DVE system (both in terms of computational and communication requirements). This characterization must be performed in two aspects:

- High-level abstraction of the application. This item determines, for example, the user behavior in networked games, in terms of mobility and user actions. In this sense, several models have been proposed [21, 22].

– Low-level measurements of system workload. These measurements determines the workload generated by each user (avatar) to the system, in terms of computational and communication requirements. Typical examples of this kind of measurements are number of messages generated per second, average number of neighbor avatars inside the AOI of a given avatar, average movement rate of avatars, etc. Some characterization of these aspects have been already proposed [23, 24]

.

Nevertheless, multiplayer online games (which are the most popular real-time applications, making up around half the top 25 types of non traditional traffic for some Internet links [25]) have special requirements, due to their real-time characteristics [26]. Despite the video game market is currently divided in more than ten types of sub genres (fighting games, role-playing games, simulation games, etc.) [27], first person shooter games (FPS) have focused the attention of the research community. The reason for such interest is due to the fact that FPS games impose the most restrictive requirements on both network traffic and QoS, because of their fast-paced nature [21]. In FPS games, the game is visualized in a first person perspective by users, who are located in a 3D virtual scene (called map) holding a gun or similar weapon [27].

Therefore, both a high-level and a low-level workload characterization are required to determine the actual workload that users in these system generate. This characterization is essential to develop efficient optimization techniques for these systems and also to design methods for providing Quality of Service. Last years several proposals have been made about characterizing the high-level abstraction of networked games [26, 21, 22]. These studies have shown that multiplayer online games have distinctive features in terms of the effects of latency on users and user behavior. Nevertheless, to our knowledge no proposal has been still made about low-level workload characterization in networked games.

The purpose of this paper is to perform a low-level characterization of networked games, starting from real traffic obtained from real games. The characterization study shows that, unlike the case of high-level workload, the low-level workload generated in networked games do not significantly differ from the workload generated in other DVE systems. Therefore, the workload models used in optimization and QoS techniques can also be applied to multiplayer online games.

The rest of the paper is organized as follows: section 2 describes the characterization setup for obtaining real traces from real networked games. Next, section 3 shows the characterization study performed on the obtained traces. Finally, section 4 shows some conclusions and future work to be done.

2 Characterization Setup

Unreal Tournament (UT) is a popular first-person action game from Epic Games [5]. In recent years, this multiplayer game has become an important testbed for

performance evaluation in fields like DVE systems [28], Human Computer Interaction (HCI) [29] and autonomous agents systems [30]. We have used a special version of this multiplayer game, called UT Game of the Year (GOTY), to obtain real traces of a networked game. The low-level analysis of such traces allows the workload characterization of networked games. The considered version of the multiplayer game not only includes an improved graphical engine and a important set of game modes (DeathMatch, Capture-The-Flag, Domination), but it also allows control avatars by using a high level language called UnrealScript. From the system architecture's point of view [8], Unreal Tournament is based on a client/server model, where a single monolithic server of the game performs constant game updates, denoted as server ticks. In each server tick, server sequentially computes the current state of the simulation and sends game state information to each client. This technique is known as *Frequent State Regeneration (FSR)* [8], and data packets contain information about the position, speed and even the model of each avatar.

Despite Unreal Tournament includes different game modes, we have selected the opposite modes (called "DeathMatch" (DM) and "Capture-The-Flag" (CFT)) in terms of both interactivity and behavior of users. In DM mode, the purpose of the game consists of killing as many opponents as possible whilst avoiding being killed. In this mode, players are independent and spend the simulation time moving and jumping around the virtual scene to kill the rest of players by shooting or strafing them. However, in CFT mode both the goal and the user behavior are completely different. In this case, players are organized in two different teams (read and blue) which are represented by a flag. The flags of the teams are located in two different zones of the scene called bases. The purpose of the game consists of those players belonging to the same team to collaboratively finding the enemy base and carry enemy flag to the team base. Unlike other mode games, DM and CTF mode games are strongly recommended when the number of players in UT is high (UT dedicated servers support games of up to 16 players).

A UT Lan Party was organized at our laboratories in early September 2005. In this experiment, a selected set of students of our university were invited to play in a session of two hours of duration. The 16 players were playing UT-GOTY version 4.3.6. on a dedicated server. The hardware platforms used as computer clients as well as the server were PC's with Pentium IV processor running at 1.7 GHz, 256 Mbytes of RAM and NVidia MX-400 graphic cards. All of them have installed Windows Xp and shared a 10 Mbps Ethernet as the interconnection network. Unlike other older FPS games as Quake [31], it was not necessary to patch or modify the source code of the game server in order to obtain the traces of played games in our experiments. In this case, we used UnrealScript to define custom monitors inside of the game map. Once the monitored maps were loaded in the current game, they forced UT engine to write the position of all the simulation clients in each server tick. In order to define comfortable maps for the UT Lan Party, where 16 players can comfortably enjoy the game, the largest four maps were properly monitored. These maps correspond to maps

called “DM-Deck16” and “DM-Shrapnel” for “DeathMatch” game mode, and “CTF-Darji16” and “CTF-HallOfGiants” for “Capture-The-Flag” game mode. All of them are well-known in the UT player community and can be found in any of the existing commercial version of this popular multiplayer game.

We have obtained five different trace files of UT game played by 16 players. Three of them correspond to three DM mode games, and we have denoted them as DTH1A, DTH1B and DTH2A. DTH1A and DTH1B traces correspond to two traces of the game played with the same map (Deck16). The idea is to check if the results show significant changes in different instances of the same game. The other two trace files correspond to CFT mode-games, and we have denoted them as FLAG1 and FLAG2. The simulation times required for obtaining each trace file were 2,25 minutes for DTH1A, 4 minutes for DTH1B, 5 minutes for DTH2A, 4 minutes for FLAG1 and 16 minutes for FLAG2.

3 Workload Characterization

The first stage in the characterization study is to check if the movement patterns followed by avatars correspond to those used in the literature [19, 24, 32, 33]. We have depicted the successive locations of each of the avatars in the environment at every simulation time, according to the traces. Figure 2 graphically shows the results obtained for two of the traces. Although the other three are not shown here for the sake of shortness, they are very similar. Figures 2 A) and B) respectively show the location of avatars in trail mode (the lines represent the locations visited by each avatar) for one of the DM traces and one of the CTF trace files. Figure 2 C) is a snapshot of the location of avatars in a given instant for the same CTF trace file. Using this tool to visually check the movement patterns, we can conclude that all of the traces show a random movement pattern. In terms of workload, this pattern is equivalent to CCP [24] movement pattern.

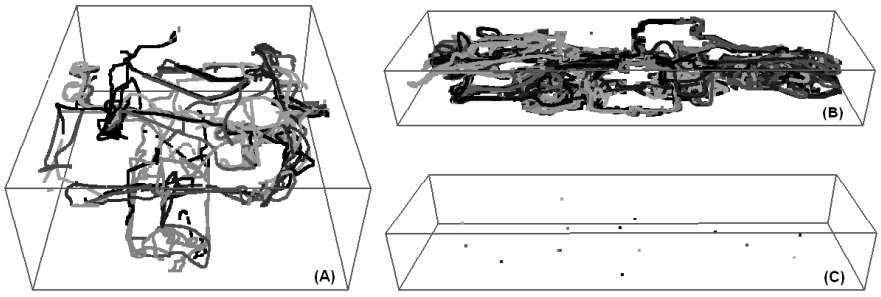

Fig. 2. Locations of avatars in different traces: A) DM mode of DTH1A trace file B) DM mode of FLAG2 trace file C) CFT mode of FLAG2 trace file

The next step in our analysis has been the detailed study of the timestamps and locations of avatars. A previous experiment was performed (before obtaining

the file traces) in order to empirically measuring the vision range (the AOI size) of avatars. In this experiment (performed in a diaphanous map), we progressively moved away two close avatars while measuring the distance to each other, until they could not see each other. Such distance was measured as 100 m. in the virtual world. It must be noticed that although two avatars cannot see each other at a given moment (due to the map geometry, including walls, corridors and/or corners) avatars must exactly know the location of all the neighbor avatars in their AOI, since they can become visible at any moment. Therefore, using 100m. as the AOI, we analyzed the obtained trace files. Although we have analyzed the values in the trace files for each of the avatars, in this paper we will only show the average and the standard deviation values obtained for the sixteen avatars, due to space limitations.

Table 1 shows the analysis of DTH trace files. This table shows in the first row the average values obtained for the sixteen avatars, while it shows the standard deviation in the next row. The average number of avatars in the AOI of each avatars were 15 in all of the DTH traces. That is, the AOI of avatars in UT reaches the whole virtual world for these maps. The most-left column shows the number of events generated in each trace (number of packets sent by each avatars). The the next column shows the number of useful events (notice that avatars send a packet at each server ticks, even though they remain stopped and have no change to report about) to compute movement rates and speeds. The column labeled with "Dist" shows the linear distance (in meters) traversed by the avatars during the game. Next, column labeled with "Speed" shows the average movement rate of avatars (in kilometers/hour) during the game. Finally, the column labeled with "Ev/sec" shows the rate of useful packets per second sent by avatars (that is, the communication workload).

Table 1. Empirical results for DTH traces

	DTH1A				
	Events	Useful Ev.	Dist.	Speed	Ev/sec
Average	4330	1295	382	10	10
Std. Dev.	167	779	181	5	6
	DTH1B				
	Events	Useful Ev.	Dist.	Speed	Ev/sec
Average	10721	2732	749	12	12
Std. Dev.	548	1285	273	4	6
	DTH2A				
	Events	Useful Ev.	Dist.	Speed	Ev/sec
Average	11373	4354	893	12	16
Std. Dev.	2937	2291	336	4	8

The results show in table 1 show that only around one third of the events are useful (that is, they include information about a new location of the avatar).

This result indicates that avatars remains stopped most of the simulation time, as indicated in the DVE literature [34]. Another important result is that the useful event rate generated by avatars is no lower than 10 events per second in any case. These rates are higher than the ones of assumed as typical in previous studies focused on DVEs. This movement rate was assumed as 2 movements/second [35], 5 events/second [18, 36], or 10 events/second in the case of games [37]. If useless events are considered (as FSR scheme [8] does), then the event rate is even higher. This is the reason for games using FSR scheme [8] to limit the number of players to 16.

Table 2 shows the analysis of FLAG trace files. The average AOI of avatars in FLAG1 and FLAG2 trace files are 10 and 11, respectively. That is, these maps are bigger than DTH maps, allowing some avatars to remain most of the time out of the AOI of other avatars. Although in this version of the game the percentage of useful events are similar to the percentages shown in table 1, the rate of useful events per second significantly increases, showing that avatars move more often in this game mode. These results indicate that avatars in FPSs show higher movement rates than in typical DVEs, and therefore they generate a higher workload [23].

Table 2. Empirical results for FLAG traces

	FLAG1				
	Events	Useful Ev.	Dist.	Speed	Ev/sec
Average	11961	4782	710	10	18
Std. Dev.	4291	2559	340	5	10
	FLAG2				
	Events	Useful Ev.	Dist.	Speed	Ev/sec
Average	34575	23968	4028	15	25
Std. Dev.	1820	6527	825	3	7

Additionally, we have measured the distances among the avatars during the simulations, in order to monitor how clustered they are in the virtual world and how these clusters vary during the simulations. Figures 3 and 4 show this measurements in a graphical way. This figures show on the x-axis the normalized simulation time (since each trace files contains different simulation times, this magnitude must be normalized in order to compare the different trace files). On the y-axis, these figures show the average value of the average distances from each avatars to all of its neighbors.

Both figures show that the average inter-avatar distance do not significantly increase as the simulation time grows. That is, avatars do not tend to head for certain points in the virtual world, as HPN [33] or HPA [32] movement patterns assume. Instead, the inter-avatar distance keeps a flat slope during the simulation. Therefore, the number of neighbors inside the AOI of avatars remains practically constant. Since the same behavior can be seen in both figures, we can

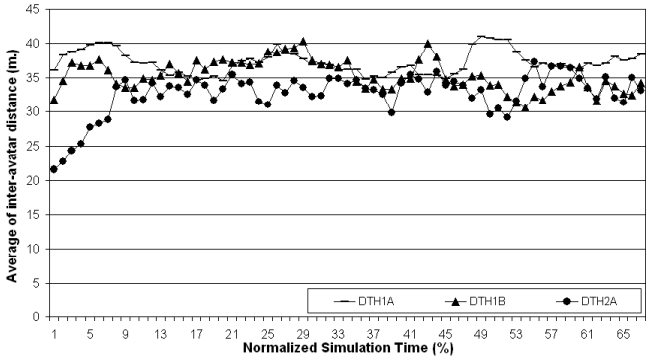

Fig. 3. Average inter-avatar distance for DTH file traces

conclude that the most adequate movement pattern for FPS games is CCP [24], since this pattern generates a constant workload during all the simulation. The existence of some peaks in the plots suggests that although HPN pattern could appear during certain periods, avatars do not follow HPA pattern at all.

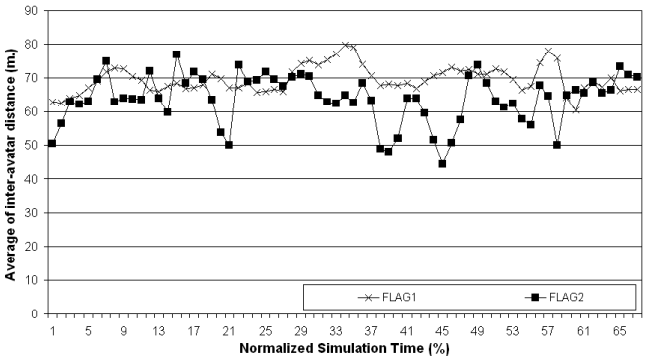

Fig. 4. Average inter-avatar distance for FLAG file traces

When comparing the two figures, we can see that the only difference between figure 3 and figure 4 is the different levels of inter-avatar distances. This difference is due to the nature of the different game modes. However, all of the plots in each figure show a flat slope.

4 Conclusions

In this paper, we have proposed the workload characterization of multiplayer networked games by obtaining real traces and analyzing the low-level measure-

ments of system workload. In order to analyze the obtained traces, we have performed both a visual and an statistical analysis.

The results show that the movement patterns followed by avatars are very similar (in terms of system workload) to some of the movement patterns propose din the literature for DVE systems. However, due to the fast-paced nature of FPS games, the movement rate of avatars are higher than the movement rates proposed for DVE systems. Therefore, we can conclude that the workload generated by multiplayer networked games follow some movement patterns already proposed, but moving faster than other DVE applications.

References

1. Miller, D., Thorpe, J.: Simnet: The advent of simulator networking. IEEE TPDS **13** (2002)
2. Salles, J., Galli, R., et al., A.C.A.: mworld: A multiuser 3d virtual environment. IEEE Computer Graphics **17**(2) (1997)
3. Bouras, C., Fotakis, D., Philopoulos, A.: A distributed virtual learning centre in cyberspace. In: Proc. of Int. Conf. on Virtual Systems and Multimedia (VSMM'98). (1998)
4. (Everquest: http://everquest.station.sony.com/)
5. (Unreal Tournament: http://www.unrealtournament.com/)
6. (Lineage: http://www.lineage.com)
7. (Quake: http://www.idsoftware.com/games/quake/quake/)
8. Singhal, S., Zyda, M.: Networked Virtual Environments. ACM Press (1999)
9. Lui, J.C., Chan, M.: An efficient partitioning algorithm for distributed virtual environment systems. IEEE Trans. Parallel and Distributed Systems **13** (2002)
10. Mooney, S., Games, B.: Battlezone: Official Strategy Guide. BradyGame Publisher (1998)
11. Gautier, L., Diot, C.: Design and evaluation of mimaze, a multi-player game on the internet. In: Proceedings of IEEE Multimedia Systems Conference. (1998)
12. Berstein, P.A., V.Hadzilacos, N.Goodman: Concurrency, Control and Recovery in Database Systems. Addison Wesley-Longman (1997)
13. Zhou, S., Cai, W., Lee, B., Turner, S.J.: Time-space consistency in large-scale distributed virtual environments. ACM Transactions on Modeling and Computer Simulation **14**(1) (2004) 31–47
14. Fujimoto, R.M., Weatherly, R.: Time management in the dod high level architecture. In: Proceedings tenth Workshop on Parallel and Distributed Simulation. (1996) 60–67
15. Roberts, D., Wolff, R.: Controlling consistency within collaborative virtual environments. In: Proceedings of IEEE Symposium on Distributed Simulation and Real-Time Applications (DSRT'04). (2004) 46–52
16. Kawahara, Y., Aoyama, T., Morikawa, H.: A peer-to-peer message exchange scheme for large scale networked virtual environments. Telecommunication Systems **25**(3) (2004) 353–370
17. Hu, S.Y., Liao, G.M.: Scalable peer-to-peer networked virtual environment. In: Proceedings ACM SIGCOMM 2004 workshops on NetGames '04. (2004) 129–133
18. Knutsson, B., Lu, H., Xu, W., Hopkins, B.: Peer-to-peer support for massively multiplayer games. In: Proceedings of IEEE InfoCom'04. (2004)

19. Morillo, P., Orduña, J.M., Fernández, M., Duato, J.: Improving the performance of distributed virtual environment systems. IEEE Transactions on Parallel and Distributed Systems **16**(7) (2005) 637–649
20. Morillo, P., Orduña, J.M., Fernández, M., Duato, J.: A method for providing qos in distributed virtual environments. In: Euromicro Conf. on Parallel, Distributed and Network-based Processing (PDP'05), IEEE Computer Society (2005)
21. Tan, S.A., Lau, W., Loh, A.: Networked game mobility model for first-person-shooter games. In: Proceedings of the 2nd workshop on Network and system support for games (NetGames'05). (2005)
22. Henderson, T., Bhatti, S.: Modelling user behaviour in networked games. In: MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia, New York, NY, USA, ACM Press (2001) 212–220
23. Morillo, P., Orduña, J.M., Fernández, M., Duato, J.: On the characterization of avatars in distributed virtual worlds. In: EUROGRAPHICS' 2003 Workshops, The Eurographics Association (2003) 215–220
24. Beatrice, N., Antonio, S., Rynson, L., Frederick, L.: A multiserver architecture for distributed virtual walkthrough. In: Proceedings of ACM VRST'02. (2002) 163–170
25. McCreary, S., Claffy, K.: Trends in wide area ip traffic patterns - a view from ames internet exchange. In: Proceedings of ITC Specialist Seminar, Cooperative Association for Internet Data Analysis - CAIDA (2000)
26. N. Sheldon, a.E.G., Borg, S., Claypool, M., Agu, E.: The effect of latency on user performance in warcraft iii. In: Proc. of 2nd workshop on Network and system support for games (NetGames'03), ACM Press New York, NY, USA (2003) 3–14
27. Wolf, M., Perron, B.: The Video Game Theory Reader. Routledge Publisher (2003)
28. Wray, R.: Thinking on its feet: Using soar to model combatants in urban environments. In: 22nd North American Soar Workshop Proceedings. (2002)
29. Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J., Agu, E., Claypool, M.: The effects of loss and latency on user performance in unreal tournament. In: Proc. of ACM Network and System Support for Games Workshop (NetGames). (2004)
30. Lozano, M., Cavazza, M., Meada, S., Charles, F.: Search based planning for character animation. In: 2nd International Conference on Application and Development of Computer Games. (2003)
31. Steed, A., Angus, C.: Supporting scalable peer to peer virtual environments using frontier sets. In: Proc. of IEEE Virtual Reality, IEEE Computer Society (2005)
32. Greenhalgh, F.C.: Analysing movement and world transitions in virtual reality teleconferencing. In: Proceedings of 5th European Conference on Computer Supported Cooperative Work (ECSCW'97). (1997)
33. Matijasevic, M., Valavanis, K.P., Gracanin, D., Lovrek, I.: Application of a multi-user distributed virtual environment framework to mobile robot teleoperation over the internet. Machine Intelligence & Robotic Control **1**(1) (1999) 11–26
34. Greenhalgh, C.: Understanding the Network Requirements of Collaborative Virtual Environments. In: Collaborative Virtual Environments. Springer-Verlag (2001)
35. Min, P., Funkhouser, T.: Priority-driven acoustic modeling for virtual environments. Computer Graphics Forum **19**(3) (2000)
36. Anthes, C., Haffegee, A., Heinzlreiter, P., Volkert, J.: A scalable network architecture for closely coupled collaboration. Computing and Informatics **21**(1) (2005) 31–51
37. Dickey, C.G., Lo, V., Zappala, D.: Using n-trees for scalable event ordering in peer-to-peer games. In: Proc. of Intntl. workshop on Network and operating systems support for digital audio and video (NOSSDAV'05), ACM Press (2005) 87–92