An experimental evaluation of server performance in Networked Virtual Environments

Juan Luis Font, José Luis Sevillano, Daniel Cascado Department of Computer Technology and Architecture, University of Seville, Spain
{juanlu, sevi, danic}@atc.us.es

***Abstract*—Several works in the literature have recently addressed the study of different Networked Virtual Environments (NVE) due to their increasing popularity and widespread use in fields ranging from entertainment to e-Health. Open Wonderland is one of these NVEs which has been the subject of several studies mainly focused on the client side. This paper aims to cover the server-side performance issues to provide complementary results that can be useful for properly sizing Open Wonderland systems according to the number of expected users. An experimental testbed is used, which provides real data that shows that CPU and outgoing bandwidth are the most critical parameters when the number of clients increase.**

***Index Terms*—Open Wonderland, Networked Virtual Environment, server performance, monitoring, profiling, testbed.**

I. Introduction

Open Wonderland (OWL) falls into the so-called Networked Virtual Environments (NVE), applications that provide virtual world experience based on a distributed simulation. OWL has been used as basis for the "persuasive system" *Virtual Valley* [1], an application aimed to motivate healthy lifestyle habits. Some of the OWL key features supporting this decision were: its GPL v2 license, its status as 100% Java project [2], which ensures a high degree of portability, and a design that allows any organisation or particular to deploy an instance in their own network infrastructure.

The latest version of this client-server architecture is Project Wonderland 0.5, which architecture is shown in Figure 1 [2]. Open Wonderland is subdivided into several independent subprojects comprising its core client and server code, modules providing extra functionality, support for third party applications embedded into OWL scenarios, MTGame Graphic Engine and jVoiceBridge, a pure-java audio platform providing services such as real-time immersive audio or software phone. Open wonderland also relies on other open-source technologies such as Darkstar, a Java platform for scalable communication and persistence games; Glassfish, a scalable, open source pure-Java application server, and jMonkeyEngine, a Java 3D game engine

In the context of NVE, an avatar is a 3D graphic representation of an user within the Open Wonderland virtual world. The movement of the avatars plus the sounds picked up by their microphones are the two of the main sources of traffic generation within the Open Wonderland context. The third main traffic source is due to the embedded third party applications.

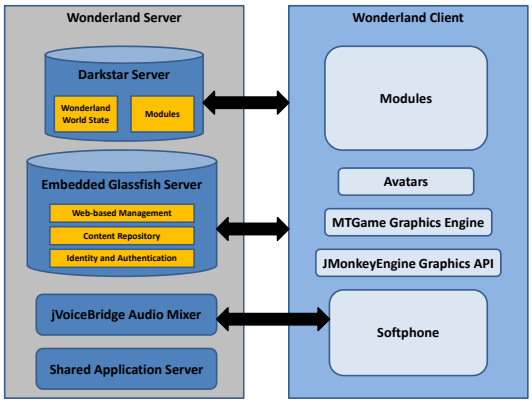

Fig. 1. Open Wonderland Architecture

This paper focuses on defining and configuring a testbed composed by an OWL server and several clients. Several testing sessions were performed over this testbed, increasing the number of concurrent clients in each session. This inducted workload was measured and monitored through several parameters such as CPU and memory usage on the OWL server. Starting from these monitoring results, the potential system bottlenecks were identified, providing an useful information for future OWL systems sizing.

The paper is structured as follows: Section II lists previous work in this area, Section III describes the configuration of the testbed, detailing hardware, software and testing session aspects. Section IV shows the monitoring results obtained for the testing sessions and discusses the evolution of server resource consumption. Finally, Section V summarises the conclusions and proposes related future work.

II. Previous work

Recent years have witnessed the proliferation of literature and work related to performance evaluation of gaming applications based on Virtual World experiences. Widespread and popular gaming genres such as First Person Shooter (FPS) or Real-Time Strategy (RTS) have attracted most academic attention as evidenced by [3], [4], [5], [6], [7], [8] or [9].

In comparison, socially-oriented Virtual World applications have received a less degree of attention, specially in network performance analysis and modelling matters. Recently, several

studies about network performance have been published following this line of work [10], some of them are specifically focused on Open Wonderland, a social-oriented Networked Virtual Environment (NVE). Thus, [11] performs a study of the OWL network traffic and proposes several models for packet inter-arrival time and packet size following a micro-scale description approach. [12] uses these models to implement a simulation tool that generates network traffic following the same generation patterns observed in the previous study.

OWL client resource requirements are easily met by reasonable-cost up-to-date hardware [13], in this case network becomes the potential bottleneck. Network congestion may lead to a drop in the NVE performance and spoil the overall user experience. The study detailed in [11] revealed that traffic originated by avatar movement was not constant over time, however, it was distributed following an exponential distribution. On the other hand, audio traffic proved to follow periodic generation patterns that could be affected by the underlying operating system and applications.

Previous work studied in detail the elements with great impact over performance on the client-side, so the performance on the server-side must to be studied to complete a performance evaluation of an OWL environment. Due to the different nature of OWL clients and server, the performance evaluation of the later will require a different approach from previous related work.

III. Testbed and Methodology details

A testbed was deployed to study the resource consumption and workload evaluation on the OWL server-side in relation to the number of concurrent users.

A. Hardware

The testbed was composed by personal computers with identical hardware resources. Both server and clients machines were equipped with up-to-date hardware according to current standards. Each machine comprised a 4-core Intel i5 750 2.66GHz processor, 4096 MB of physical DDR3 RAM, a dedicated AMD Ati HD 4350 1GB DDR3 RAM GPU and 7200 RPM hard disk.

The client hardware configuration does not respond to any specific requirement beyond the minimum specifications to run the OWL Java client smoothly and displaying the virtual world 3D graphics properly. Clients have the same hardware configuration as the server for convenience, due to the fact that all machines belonged to the same computer room. This hardware configuration was powerful enough to successfully run an OWL server instance according to the OWL Project recommendations [13]. These hardware resources also exceed the recommendations for OWL clients both in computing power and graphic performance.

Gigabit Ethernet technology has been used to interconnect all the machines. They all are connected to a Gigabit Ethernet switch so they share a common collision domain.

B. Software

The OWL server instance ran over GNU/Linux, an Ubuntu 10.04 distribution compiled for 64-bit architecture (amd64). Although the Java platform guarantees OWL to run on any supported architecture, GNU/Linux and other Unix-like operating systems are a common choice for OWL servers due to their maintenance and administration facilities. This operating system family provides mature technologies and tools for remote administration and task automation. The server had the Sun JDK 1.6.0-26 compiled for 64-bit architecture. The Open Wonderland binary deployed on the server was v0.5 nightly-build from 20th February 2012.

On the other hand, Microsoft Windows XP Professional SP3(32-bit) was used in the OWL clients. Windows XP-based clients were extensively studied in previous work [11] so their network behaviour and server interaction are well-known. Moreover, Windows XP still represents a significant percentage of the domestic operating system market so its use on the client-side of the testbed is a realistic assumption. The default Windows Firewall was activated for each client and configured to not block the OWL network traffic. All Windows clients had the Sun JRE 1.6.0-30 for 32-bit systems. AMD Ati GPU drivers were also present to ensure a proper graphic performance.

C. Testing sessions

Several gaming sessions have been performed to determine the evolution of resource consumption on the server-side. The study aims to reproduce gaming sessions that maximise user activity and data exchange between clients and server, generating the worst-case server workload. This scenario is induced by constantly generating avatar movements and audio transmission, avoiding any traffic due to third party embedded applications.

All the sessions took place in the same OWL "world", a predefined scenario shared by all the clients where they could move freely. For simplicity, the chosen scenario was minimalistic to make client movement automation easier. The number of concurrent players in each gaming session ranged from 1 to 15. Although more densely populated sessions such as auditorium-like scenarios with only one speaker and numerous listeners are possible, they do not fit with our description of "highly active" session, where all the users transmit audio and interact constantly. Further users represents a situation that is out of our scope, a regular OWL session can only host a specific number of user actively interacting between them without overwhelming their perception and this kind of session is the situation we want to reproduce.

Each gaming session lasted 10 minutes plus a 1 minute of warm-up to ensure stationary state monitoring only. Due to the simplicity of the scenarios, this stationary state of user activity was quickly reached and it remained constant, so longer sessions are not necessary to get feasible data.

Each testing session consisted in a new OWL session where all the users logged one by one until reaching the desired number of players. Once the initial virtual world instance was

transmitted from the server to each client, the session was ready to start the monitoring and traffic capture.

D. Client activity generation

This study contemplates two main sources of activity to induce workload on the OWL server: avatar movement and audio transmission.

On the one hand, movement of a client's avatar within the virtual world requires to send information about speed and direction to the server. Then, it has to update the avatar position within its instance of the virtual world and propagate the changes to the rest of clients to update and synchronise their respective views of the world. All this process comes at a certain resource cost. On the other hand, the server receives all the audio data sent by clients, mixing, processing and forwarding it to the rest of clients. Considering that audio is a steady flow of data with a significant associated bandwidth, its transmission and processing also requires resources in terms of network, computing and memory.

A number of concurrent OWL clients ranging from 1 to 15 were deployed for the testing sessions, so it was necessary to automate avatar movements to keep them performing at high activity rates without the intervention of a human operator behind each avatar providing navigation keystrokes and speaking to a microphone to generate audio traffic.

Human interaction on the keyboard was replaced by Visual Basic Script (VBScript) piece of code capable of activating the OWL client window and continuously sending navigation keystrokes. VBScript interpreter is shipped with Windows XP by default and allows to create run scripts that use operating system functionalities and API such as the SendKey function [14].

Avoiding human interaction to provide audio feedback required the use of "Stereo Mix", a feature present in the Windows XP audio mixer and related to the system soundcard and audio drivers. This feature allows to redirect the output audio signal to the microphone input channel, so a media player can be used as audio source.

E. Monitoring

Resource monitoring tasks were performed using System Activity Reporter (SAR), a Solaris-derived system monitoring tool ported to GNU/Linux and available in Ubuntu within the sysstat package. SAR allows to monitor several system resource parameters such as overall CPU and memory usage, I/O operations, network traffic, etc; with a minimum impact over the system performance. Monitoring frequency and duration is also configurable as well as the output files to store resulting logs.

The SAR tool was invoked for each testing session with a monitoring frequency of 1s during the whole 11 minute gaming session, fetching data for all the available load parameters for later study. The resulting log for each session was stored in separate files. SAR is characterised by its minimum impact over the monitored system, it generates monitoring data at a rate below 10KB/s,

Network traffic has also been captured during testing sessions using *tshark*, a command line version of the packet analyser *Wireshark*. The impact due to this tool is discussed later. The present paper only deals with the total bandwidth generated and consumed during the testing sessions, the captured traces will be the subject of further study in future papers.

IV. Results

This paper focuses on a subset of all the resource parameters that can be monitored by SAR. Specifically the figures for *Percentage of CPU Usage*, *Percentage of Memory Usage*, *Percentage of CPU time spent in I/O* operations and network bandwidth have been studied. After discarding the warm-up values, a total of 600 measurements have been used per parameter and session.

A. CPU Usage

The OWL server was equipped with a 4-core processor, allowing 4 concurrent threads. The Java Virtual Machine instance of the OWL server automatically balanced the CPU load between the system cores so the *Percentage of CPU Usage* addressed in this section corresponds to the mean load of the four cores. No frequency scaling policies were enabled during the testing sessions, so all the CPU percentages correspond to the processor working at nominal capacity.

Mean % CPU load value for each session ranged from 10% for 1 single client to near 90% for the 15-client case while the standard deviation remains stable. The evolution of the CPU usage is approximately linear over the number of players. The CPU requirements for the 15-client session are near the maximum CPU capacity of the testing OWL server, so further sessions increasing the number of players would have lead to a performance drop due CPU overload in this server. Table I shows the % CPU usage mean (μ_{cpu} column) and standard deviation (σ_{cpu}) figures for the testing sessions.

TABLE I
TOTAL % CPU USAGE STATISTICS OVER SESSIONS

N. Client	μ_{cpu}	σ_{cpu}		N.Client	μ_{cpu}	σ_{cpu}
1	9.17	0.97		9	67.63	3.46
2	18.55	1.33		10	71.93	2.51
3	28.06	2.00		11	74.80	3.58
4	36.26	2.68		12	75.26	3.28
5	43.44	3.27		13	81.90	3.66
6	47.15	2.82		14	85.61	2.88
7	52.31	2.47		15	88.09	3.60
8	59.35	2.84				

Figure 2 shows the evolution of mean *Total % CPU Usage* in relation to the number of clients per session. Should be note that a small part of the overall CPU usage is due to I/O operations, also detailed in Figure 2. The *% CPU excluding I/O* represents the CPU time used by processes not including I/O operations (*%CPU for I/O*). These and other resource parameters will be discussed later. CPU time consumption increased faster than the rest of studied parameters, arising as the limiting factor in the testbed server.

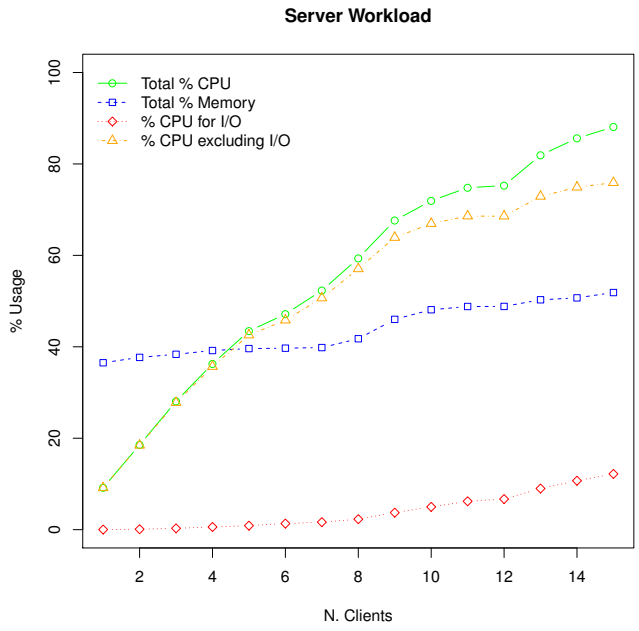

Fig. 2. Server % CPU, Memory and I/O usage per session

B. Memory usage

Total available memory considered by the monitoring tool is not equal to the total physical memory installed on the server, since the memory used by the Linux kernel and some devices is not counted as available. Thus, the total available memory for user space processes on the OWL server was of 3894 MB. This total was used as basis for calculating the *Percentage of Memory Usage*.

Although GNU/Linux systems tends to use all the available memory for performance reasons, only a fraction of the used memory is really allocated as process memory, the rest is used as buffer and cache memory, which are used by the kernel to speed up I/O operations. Thus, the used memory figures do not include the buffer or cache.

The increase of server memory in use per each new OWL client is relatively moderate, the overall server memory usage, also including process not related to OWL, ranges from 36.5% for the 1-client session to 51.8% for the 15-client session, being the mean increase of 1.1% which translates in a memory usage from 1421 MB to 2017 MB for 1 and 15-client sessions respectively with a mean increment of 42 MB per new client. Table II shows the *% Memory Usage* (μ_{mem} column) and standard deviation (σ_{mem}) for each testing session. Regarding these figures and other SAR measurements, server did not use swap memory during any of the testing sessions so it was able to host all the concurrent clients while still having an important percentage of free memory. Figure 2 shows that the memory usage is approximately linear in relation to the number of clients, although with a very low slope, that is, with an increase much less pronounced than the CPU usage.

TABLE II
% MEMORY USAGE STATISTICS OVER SESSIONS

N. Client	μ_{mem}	σ_{mem}		N.Client	μ_{mem}	σ_{mem}
1	36.50	0.10		9	46.00	1.40
2	37.68	0.21		10	48.10	0.32
3	38.36	0.17		11	48.81	0.52
4	39.18	0.16		12	48.83	0.45
5	39.61	0.23		13	50.28	0.34
6	39.70	0.21		14	50.70	0.59
7	39.83	0.29		15	51.86	0.46
8	41.76	0.65				

C. I/O

The *% CPU usage for I/O* indicates the total percentage of CPU time spent on I/O operations. Although mean values are provided, CPU load for I/O has not a stable value over time. Instead, CPU load due to I/O ($\mu_{i/o}$) is periodically concentrated in short intervals of time separated by idle periods, which explains the values for the standard deviation shown in Table III ($\sigma_{i/o}$). Figure 2 shows a plot of these values along the evolution of the overall CPU usage and memory usage. The *Total % CPU* usage breaks down into *% CPU for I/O* and *% CPU excluding I/O*. Virtual world information is loaded in each client at the beginning of the gaming session. Then, the I/O activity in the server is relatively low. The CPU time spent on I/O operations is negligible for a small number or users and its increase is explained by the packet capture process performed in the server during the testing sessions. The greater the number of users, the greater the size of the captured traces, which translates in a higher CPU activity due to I/O operations. The data rate during the traffic captures ranges from 116 KB/s for 1-client session to a maximum of 7.6 MB/s for 15-client session. After ignoring the I/O operations due to trace capturing, it can be stated that OWL itself is not an I/O intensive application once their users have logged.

TABLE III
% CPU USAGE FOR I/O OP. STATISTICS OVER SESSIONS

N. Client	$\mu_{i/o}$	$\sigma_{i/o}$		N.Client	$\mu_{i/o}$	$\sigma_{i/o}$
1	0.02	0.06		9	3.73	12.69
2	0.10	0.37		10	4.98	15.01
3	0.29	1.28		11	6.21	15.76
4	0.58	2.62		12	6.69	17.61
5	0.88	2.93		13	9.00	18.90
6	1.33	5.51		14	10.70	22.13
7	1.64	6.89		15	12.19	21.99
8	2.30	8.51				

D. Network Bandwidth

The main network traffic sources on the OWL testbed were the avatar movement and audio transmission. Clients send their update requests to the server and it processes and forwards these synchronisation packets to the rest of clients to update their virtual world view. The server also receives audio from

clients, mixing and sending the correspondent environmental audio to each client.

Table IV shows bandwidth figures in KB/s for both outgoing and incoming server traffic. Incoming traffic (BW_{IN} column) increases linearly in relation to the number of clients from 84 KB/s to 1395 KB/s, depending on the number of clients. The mean incoming traffic increase per client is 93.8 KB/s with a standard deviation of 27.2. The overall standard deviation (σ_{IN}) is also shown. On the other hand, outgoing server traffic (μ_{OUT} column) experienced a greater increase compared to incoming traffic. The outgoing traffic increased quadratically in relation to the number of clients within the testing sessions.

TABLE IV
BANDWIDTH (KB/S) STATISTICS OVER SESSIONS

N. Client	BW_{IN}	σ_{IN}	BW_{OUT}	σ_{OUT}
1	84.02	1.34	23.00	0.71
2	168.02	1.65	194.36	46.04
3	256.57	1.93	377.32	54.84
4	342.21	2.82	604.45	21.52
5	432.76	2.71	878.65	7.90
6	523.34	2.66	1183.51	11.29
7	614.43	2.79	1535.20	39.84
8	709.40	3.86	1929.83	25.34
9	802.96	3.69	2363.09	83.39
10	835.37	4.27	2852.32	105.50
11	993.65	23.04	3297.85	227.72
12	1070.06	7.45	3675.15	206.11
13	1196.96	9.66	4545.05	294.46
14	1298.53	7.89	5219.82	219.47
15	1395.87	11.03	5739.21	343.51

This asymmetry between incoming and outgoing traffic is explained by the duplication and forwarding of part of the OWL traffic, corresponding to client update request. Each update request received by the OWL server has to be forwarded to the rest of n-1 clients within the gaming session, thereby increasing the outgoing traffic over the incoming one. Figure 3 shows the difference between incoming and outgoing traffic, which is accentuated with the increasing number of clients.

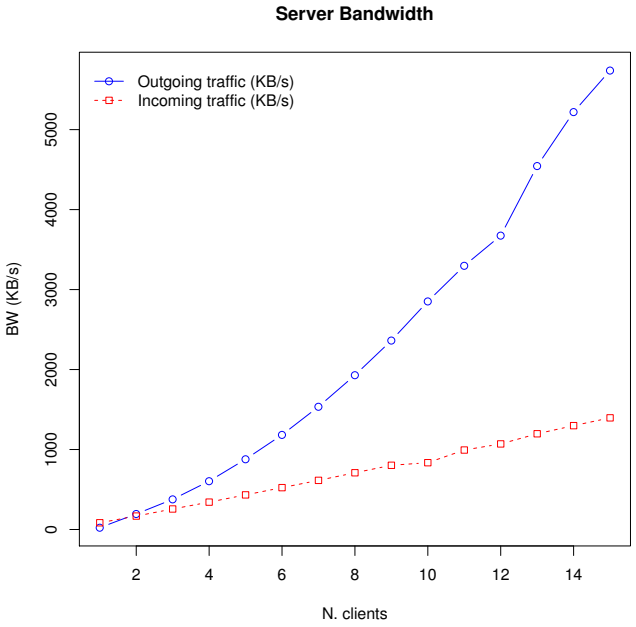

Fig. 3. Incoming and Outgoing Server Traffic over Sessions

V. CONCLUSIONS AND FUTURE WORK

This paper focused on studying the resource consumption on the server-side of a Networked Virtual Environment (NVE) based on Open Wonderland (OWL). For this purpose an OWL testbed was defined and configured,using up-to-date hardware for both clients and server, all of them connected by a Gigabit Local Area Network. Server ran Ubuntu GNU/Linux while all the clients shared the same software configuration based on Windows XP Professional. In order to avoid human interaction on each client during testing sessions, the movement and audio generation were automated.

The study of server monitoring logs generated by SAR identified the CPU usage as the system bottleneck, becoming by far the limiting factor in the proposed testbed. Its increment over the testing sessions was greater than the rest of studied parameters, the server processor working at almost full capacity for the more populated sessions. Adding new clients had not a great impact over memory usage, which increased smoothly between sessions while CPU usage for I/O operations became only noticeable for higher number of concurrent clients. Finally, it was observed an asymmetry between the incoming and outgoing server traffic due to the inner OWL communication mechanisms used to propagate object updates between clients.

Summing up, CPU computing power defines the OWL server capacity in terms of concurrent clients. Outgoing server traffic also deserves special attention since its bandwidth requirements increase quadratically in relation to the number of clients.

Future work includes further studies on the nature of the network traffic as well as determining the resources required to avoid penalisation of the user experience when hosting a higher number of clients.

ACKNOWLEDGEMENT

This work was partially supported by the project PROCUR@ - IPT-2011-1038-900000, funded by the program INNPACTO of the Spanish Ministry of Science and Innovation and FEDER funds; project Vulcano: TEC2009-10639-C04-02; and by the Telefonica Chair “Intelligence in Networks” of the University of Seville, Spain.

Special thanks to the Computing Centre and staff of Escuela Técnica Superior de Ingeniería Informática, University of Seville, for their assistance and collaboration.

REFERENCES

[1] D. Cascado, S. Romero, S. Hors, A. Brasero, L. Fernandez-Luque, and J. Sevillano, “Virtual worlds to enhance ambient-assisted living,” in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 31 2010-sept. 4 2010, pp. 212 –215.

[2] O. W. Project, *Open Wonderland Project*, accessed March 15^{th} 2011. [Online]. Available: http://www.openwonderland.org/

[3] W. Feng, F. Chang, and J. Walpole, "Provisioning on-line games: a traffic analysis of a busy counter-strike server," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 151–156.

[4] G. Armitage, "An experimental estimation of latency sensitivity in multiplayer Quake 3," in *Networks, 2003. ICON2003. The 11th IEEE International Conference on*. IEEE, 2003, pp. 137–141.

[5] M. Claypool, D. LaPoint, and J. Winslow, "Network analysis of counter-strike and starcraft," in *Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International*. IEEE, 2003, pp. 261–268.

[6] J. Färber, "Traffic Modelling for Fast Action Network Games," *Multimedia Tools and Applications*, vol. 23, no. 1, pp. 31–46, May 2004.

[7] P. Branch and G. Armitage, "Towards a general model of first person shooter game traffic," *Swinburne University of Technology, Tech. Rep. 050928A*, pp. 1–11, 2005.

[8] A. Cricenti and P. Branch, "ARMA(1,1) modeling of Quake4 Server to client game traffic," *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games - NetGames '07*, pp. 70–74, 2007.

[9] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The effect of latency on user performance in Warcraft III," in *Proceedings of the 2nd workshop on Network and system support for games*. ACM, 2003, pp. 3–14.

[10] H. Liu and M. Bowman, "Scale virtual worlds through dynamic load balancing," in *2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, Oct. 2010, pp. 43–52.

[11] J. "Font, D. Cascado, J. Sevillano, F. Díaz-del Río, and G. Jiménez, "Network traffic analysis and evaluation of a multi-user virtual environment," Jan. 2012, accepted for publication in SIMPAT.

[12] J. "Font, D. Cascado, and J. Sevillano, "Design, Implementation and Validation of a Simulation Tool for Networked Virtual Environments," Jan. 2012, accepted for publication in CITS 2012, Amman, Jordan.

[13] "FAQ | open wonderland," http://openwonderland.org/about/faq. [Online]. Available: http://openwonderland.org/about/faq

[14] "VBScript - SendKeys method - TechNet articles - home - TechNet wiki," https://social.technet.microsoft.com/wiki/contents/articles/5169.aspx. [Online]. Available: https://social.technet.microsoft.com/wiki/contents/articles/5169.aspx