Design, Implementation and Validation of a Simulation Tool for Networked Virtual Environments

Juan Luis Font , Jose Luis Sevillano , Daniel Cascado-Caballero, Gema Lopez-Muñoz , Berhanu Regassa
Department of Computer Technology and Architecture, University of Seville, Spain
{juanlu, sevi, danic, gemalopez, berhanu}@atc.us.es

***Abstract*—The popularisation of Networked Virtual Environment (NVE) applications in several context, including social interaction and *e-Health* fields, makes them an interesting subject of study. The proper estimation of their network requirements are key to ensure a good user experience, an important factor that determines the acceptance and success of the applications. The studies focused on NVE with an important social component are relatively scarce. This paper presents the design, implementation and validation of a simulation tool to assist in the study of the requirements of a NVE application, Open Wonderland. This tool will facilitate the study of several parameters that define the user experience. The study describes the overall structure, development framework and models that define the traffic generation patterns and validates the simulation results by comparing its traces with real traffic from previous studies.**

I. Introduction

Open Wonderland (OWL) is one of the so-called Networked Virtual Environment applications, which are focused on providing a virtual world experience based on a distributed simulation. The success of several of these application in contexts such as gaming or social interaction have contributed to introduce them to larger audiences. This popularisation together with the proliferation of the “persuasive systems”, focused on motivating healthy lifestyle habits [1], have led to our research group to develop a persuasive system called “Virtual Valley”, based on Open Wonderland and relying on the virtual world concept [2], [3].

OWL has several characteristics that make it suitable as basis for a research persuasive system. First, it is licensed under the GPL v2, it is also a 100% Java project [4], which ensures a high degree of portability, and its design allows to any organisation or particular to deploy an instance in their own network infrastructure. The network traffic derived from Wonderland is mainly due to three sources. First, object synchronisation which allows all users to have a coherent view of the virtual world (including moving objects like avatars). Second, messages intended to support communications among users, including voice traffic (the main source of traffic) but also text messages (chat). And, finally, traffic due to the execution of applications shared among different users. The latter is very difficult to model, as it depends on the particular application. Therefore, this study will focus on the first two sources: object synchronisation and voice traffic.

The user experience is a key factor for the success of a NVE such as OWL. Poor graphic performance, network lag or inconsistencies translate into a poor user experience that may lead the user to reject the application. Assuring such a good user experience requires the study of the parameters that have a direct impact on it, identifying the threshold below which the application becomes unusable. In this context, the network traffic performance is of great importance. A virtual world environment requires certain network resources to keep its clients synchronised, sharing a common and coherent view of the same simulated reality. It is important to determine the maximum throughput for OWL clients and servers to properly size the required network resources and be able to predict the throughput evolution depending on the scenario. Network simulators play a useful role to study such requirements.

This paper aims to define, implement and validate a simulation tool to facilitate the study of the network requirements of OWL clients. The models used to implement the traffic generation patterns have been defined in previous work. The paper is structured as follows: Section II lists previous work in this area, Section III contains the network modes proposed for OWL client traffic in previous work. The simulator framework, structure and implementation details are listed in Section IV and the validation and results are discussed in Section V. Finally, the paper closes with the conclusions and future work in Section VI, acknowledgments and bibliography.

II. Previous work

Most studies about Networked Virtual Environments (NVE) focus on multi-player online games, specially on First Person Shooters (FPS) [5]. Real-Time Strategy (RTS) [6] and Massively Multi-player Online Role-Playing Games (MMORPG) [7] are also genres subject of academic research, although in a lesser extent.

The quality of the user experience of each genre depends on different aspects, so various techniques are deployed depending of the requirements. For example, UDP transport protocol is used by those that give preference to time restricted updates over transmission reliability. On the other hand, applications that require a reliable data stream service rely on TCP protocol. OWL follows a hybrid approach using TCP connections for object synchronisation traffic and UDP for audio transmission.

Similar works dealing with requirements for NVEs are scarce: Quality of Service (QoS) is studied in [8] and [9].

Bandwidth requirements for some popular multi-player online games are studied in [10] monitoring network traffic during several sessions.

To the best of our knowledge, there are no published studies about the networking resources needed to support the execution of Virtual Worlds based on Open Wonderland except for the preliminary approach made in [11] and further study performed in [12]. This paper follows a micro scale modelling approach similar to that described in [13] and proposes a set of models to describe the traffic generation patterns of OWL clients.

III. Network traffic models for OWL clients

This section presents a summary of the modelling process results presented in [12], a study of the traffic generation patterns of the OWL clients. These models focused on modelling the Inter-Arrival Time between consecutive packets and packet size for two of the main OWL traffic sources: TCP object synchronisation and UDP audio traffic. The first is used to synchronise virtual world simulation shared by all the clients so its volume of traffic depends on the user activity. The second traffic is responsible of the audio conversations and softphone calls.

On the one hand, the object synchronisation process generates update requests and propagations depending on the user activity. Constant movement translates into a higher rate of object synchronisation packets meeting some restrictions to avoid an updates overflow. Highly active users generate synchronisation packets following a truncated exponential distribution. The rate parameter λ, that determines the exponential behaviour, depends on the number of concurrent users within the session. Experimental results proved that λ decreased with the increase in concurrent users. Table I shows rate λ values calculated by Maximum Likelihood Estimation (MLE) for various experimental sessions. This model is described by a continuous random variable in Equation 1. Inactivity periods have a considerable impact on the packet rate and are of little interest for the current study. There are few possible packet sizes whose probabilities can be described using a discrete random variable as shown in Equation 2.

TABLE I
Rate parameter λ values for object synchronisation IAT

Session	λ
2-client	4.168
3-client	3.687
5-client	3.146

$$F(x;\lambda) = \begin{cases} 1 & , \quad x > 0.5 \\ 1-e^{-\lambda x} & , \quad 0.5 \geq x \geq 0 \\ 0 & , \quad x < 0 \end{cases} \tag{1}$$

$$P(x) = \begin{cases} 0.9704 & , \quad x = 239 \quad bytes \\ 0.0195 & , \quad x = 478 \quad bytes \\ 0.0057 & , \quad x = 536 \quad bytes \\ 0.0044 & , \quad x = 717 \quad bytes \end{cases} \tag{2}$$

On the other hand, audio packets are generated following periodic patterns that can slightly vary depending on the underlying operating system and network stack, although the overall throughput is similar among different configurations. This paper focuses on the Windows XP case, whose Inter-Arrival value behaviour is described in Equation 3. The vast majority of the audio packets have a UDP payload size of 1292 bytes, so this parameter is considered to be constant.

$$P(X=k) = \begin{cases} 0.1426 & , \quad k = 0.0114 \quad s \\ 0.6405 & , \quad k = 0.0221 \quad s \\ 0.0723 & , \quad k = 0.0340 \quad s \\ 0.1446 & , \quad k = 0.0442 \quad s \end{cases} \tag{3}$$

(4)

IV. Implementation of the simulator

The development of the simulation tool has not been undertaken from scratch. Instead, ns-3 discrete-event network simulator has been chosen as starting point. Ns-3 provides a modular and well tested simulation framework, it is an open source community-driven project and provides a well defined building process, comprehensive documentation and the possibility of coding in C++ or Python [14], [15]. Ns-3 is devised to be executed in GNU/Linux and Unix-like environments, being also possible to build and run it in Windows systems using CygWin. The version used in this paper is ns-3.13, last stable release at the time of writing.

Ns-3 code roughly distinguishes internally between models, helpers and simulations. The models are classes that implement the functionality of the modelled real systems, such as network nodes, data packets, protocol stacks, network devices or packet generating applications. On the other hand, helpers are auxiliary classes that make the creation and configuration of large amounts of related entities easier, such as definition of several interconnected nodes, the initialisation of large amounts of network decides with consecutive IP addresses, and so on. Finally, a simulation is a program that defines, configures and launches the simulation experiment based on the models and helpers.

The ns-3 code that implements the OWL clients is written in C++. According to the ns-3 software architecture, the `Node` class acts as a container entity that represents a network node. This entity contains objects that model the network devices, protocol stacks and links to other entities such as physical channels. Several application models can be "installed" on top of a node object. These applications are classes that inherit from the generic `Application` class. They generate and consume network traffic following the logic coded by the programmer. The simulator structure is shown in Figure 1.

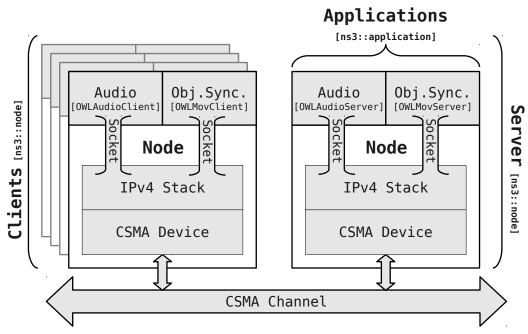

Fig. 1. Simulator architecture

The classes that implement the OWL client-server architecture are listed below:

- `OWLMovClient` generates TCP object synchronisation packets with Inter-Arrival Times defined by the distribution in Equation 1. The TCP payload size is determined by Equation 2. It uses a TCP `Socket` object to pass packets to the IPv4 stack of its containing `Node`. It is set up with the IP address and TCP port of its associated OWL server.
- `OWLMovServer` receives TCP object synchronisation packets from several `OWLMovClient` objects and dumps them to disk into `pcap` files.
- `OWLAudioClient` generates audio packets with Inter-Arrival Times determined by the distribution in Equation 3. The UDP payload size is 1292 bytes. It uses an UDP `Socket` object to pass packets to the IPv4 stack of its containing `Node`. It is set up with the IP address and TCP port of its associated OWL server.
- `OWLAudioServer` receives UDP audio packets from several `OWLAudioClient` objects and dumps them to disk in `pcap` files.

The simulated network infrastructure is defined and configured in the `owl_client_sim.cc` file, located in `$NS_DIR/scratch`. The simulation comprises a Carrier Sense Multiple Access (CSMA) bus topology operating at 100 Mbps and interconnecting an arbitrary number of nodes. This topology is equivalent to the original 100 Mbps Ethernet network used in the testing sessions. All the simulation nodes share the same CSMA channel and are equipped with CSMA network devices. Each node has also a IPv4 stack and associated IP address. All these models are provided by default with the ns-3 simulation and are well tested and documented.

The `Application` child classes are installed in the simulated nodes. In this topology there is only a single server and an arbitrary number of clients. Each client node has a `OWLMovClient` and `OWLAudioClient` object associated while the only server has a `OWLMovServer` and `OWLAudioServer` instance installed.

The ns-3 building process generates an executable file `owl_client_sim`, that lacks Graphical User Interface (GUI) and can be launched from command line using `waf`, the build automation tool used by ns-3. The simulation executable accepts several parameters that are listed below:

- `nClients`: specifies the number of OWL clients participating within the simulation. Default value 2.
- `packetRate`: rate parameter λ, determines the distribution of the IAT values for object synchronisation packets. Clients within scarcely populated sessions have λ values close to 5 while λ tends to 3 for overcrowded sessions. Default value 4.168, two concurrent highly active users.
- `tSimulation`: sets in seconds the duration of the simulated OWL session. Default value 250 s.
- `seed`: specifies the seed to initialise the random number generations.
- `logPath` and `logName`: specify respectively the store directory and the root name for the output trace file. The `logName` string is appended with a time stamp. Default values are "/tmp/" and "owl_trace_".

The simulator generates as output a network trace captured in the OWL server side, including both audio and object synchronisation traffic. This trace is stored as a file in `pcap` format.

V. Simulation results and validation

The packet analyser Wireshark v.1.6.5 was used to study the simulated and experimental traces. Tshark is the terminal-based version of Wireshark, providing the same functionality regardless the GUI and facilitating the scripting and automation tasks. Command line tool *capinfos* was used to extract statistics from the `pcap` files, Bash scripting to articulate the automation tasks and the statistics language R for calculations and plots.

Packets from real traces missed 4 bytes corresponding to the Frame Check Sequence (FCS) for Ethernet. Most operating systems do not provide these bytes to packet analysers. On the contrary, simulated traces include the Ethernet FCS. These 4 extra bytes per packet have been taken into account in the calculation of the throughput for real traffic.

Experimental data from [12] came from three cases, comprising 2, 3 and 5 concurrent OWL clients respectively. The study of such a reduced set of clients is not determined by the server resources, but by the number of users that can simultaneously interact while providing a proper user experience without overwhelming the end user. All the clients within those sessions performed high activity rates. Three equivalent simulation configurations have been defined to be recreated by simulation. The goal is to compare the simulator throughput with previous experimental results. Rate parameter values from Table I have been used to set the object synchronisation packet rate according to the number of concurrent users specified for each simulation run.

Configurations for the 2, 3 and 5-client cases have been run 100 times each one in order to obtain the sample mean ($\bar{X}$) and sample standard deviation ($\sigma_{\bar{X}}$) with a Confidence Interval (CI) of 95%. Each run used a different seed to initialise the random number generators.The duration of each simulated session was of 250 s. Packet sender/receiver classes do not have to establish network connections or load simulation entities which generate heavy data traffic. Instead, they start generating traffic following stationary patterns from real OWL clients. Thus, due to the logic coded in the packet sender/receiver classes, the warm-up period is almost negligible and the clients reach long run throughput values in early simulation stages, so long simulation times are not needed.

Results showed that the simulated object synchronisation throughput decreases as the number of concurrent clients increases. Results for simulator runs using λ parameters from Table I are shown in Table II. 100 simulator runs (samples) have been performed for each configuration (2, 3 and 5-clients), sample means ($\bar{X}$) decrease as the number of clients increases. The sample deviation ($\sigma_{\bar{X}}$) does not experience significant variations. Figures have a 95% Confidence Interval assuming a normal distribution of the sample means values. Experimental results from the previous study are shown in Table III. Although the simulation results show a slight overestimation, real and simulated client throughput follow a similar evolution linked to the number of concurrent users. This correlation can be observed in Figure 2. It should be noted that the experimental results have a relatively high deviation due to the variable nature of the traffic. Despite the slight overestimation, the simulator results do not differ largely if the experimental deviation is taken into account. Simulator means are within or very close to the interval defined by the deviation about the experimental means.

TABLE II
SIMULATED OBJECT SYNCHRONISATION THROUGHPUT

N. Clients	Samples	$\bar{X}(b/s)$	$\sigma_{\bar{X}}$
2	100	1475.36	48.07
3	100	1352.62	49.99
5	100	1231.95	42.76

TABLE III
EXPERIMENTAL OBJECT SYNCHRONISATION THROUGHPUT

N. Clients	$\bar{X}(b/s)$	$\sigma_{\bar{X}}$
2	1407.49	19.27
3	1282.78	141.62
5	1085.91	134.72

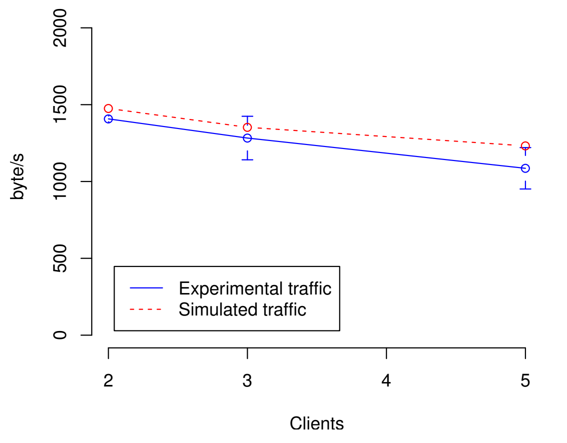

Fig. 2. Simulated Vs experimental object synchronisation throughput

Audio client traffic generation follows a periodic pattern dictated by jVoiceBridge, the software component responsible for sending audio packets following a theoretical period of 20 ms. Experimental results showed a different generation pattern while the overall audio throughput remained constant over time and slightly below the expected maximum theoretical audio throughput [12]. Thus, the audio Inter-Arrival Time model (Equation 3) was proposed to better describe audio traffic and give a more realistic estimation than the theoretical one only based on the used audio coding. A summary of the simulator results for audio traffic are shown in Table IV. All the samples have a similar mean value ($\bar{X}$) independently of the number of concurrent clients participating in the session. The sample variance $\sigma_{\bar{X}}$ and Confidence Interval do not show significant changes between configurations. The simulated audio throughput values overestimate the experimental results shown in Table V. Despite this difference, simulation results still provide a more accurate estimation than theoretical maximum audio throughput, which was estimated in 62.4 KB/s for the audio coding used. Simulated and real throughput also show a considerable degree of correlation that can be observed in Figure 3.

TABLE IV
SIMULATED AUDIO THROUGHPUT

N. Clients	Samples	$\bar{X}$ (b/s)	$\sigma_{\bar{X}}$
2	100	54336.05	310.94
3	100	54363.64	332.80
5	100	54356.50	333.34

TABLE V
EXPERIMENTAL AUDIO THROUGHPUT

N. Clients	$\bar{X}$ (b/s)	$\sigma_{\bar{X}}$
2	48187.14	433.24
3	45678.54	3990.91
5	47816.95	4216.89

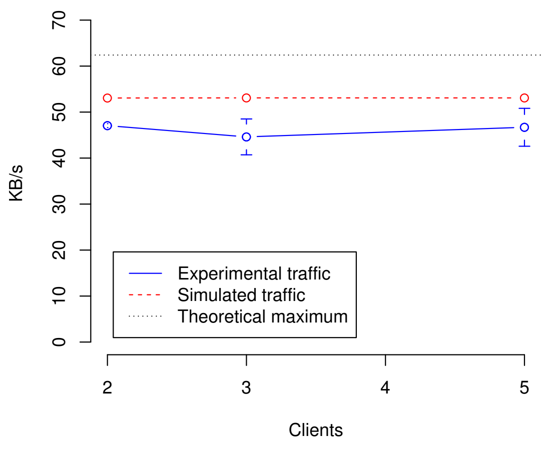

Fig. 3. Simulated Vs experimental audio throughput

VI. Conclusion and future work

This paper describes the design, implementation and validation of a simulation tool for studying Open Wonderland, a Networked Virtual Environment (NVE). The tool focuses on the outgoing traffic of OWL clients. This traffic is divided into two categories: audio and object synchronisation. Their micro scale models were defined in previous work.

Ns-3 project was used as simulation framework to avoid developing the tool from scratch. Some of the ns-3 default models, such as those for Ethernet and IPv4 have been used along with other custom classes that implement the traffic generation behaviour of the OWL clients. The resulting command line tool is configurable through parameters and generate network traces in `pcap` format. This feature allows to study the simulator traces using tools such as Wireshark and Tshark, to automatise the trace generation and the interoperability with other tools.

The validation of the simulation tool was performed defining several simulation scenarios that replicated the experimental sessions from previous work. These configurations described several multi-player sessions whose clients showed high rates of user activity. The simulation input was the rate parameter λ, which determines the packet generation relative to the number of concurrent users. Throughput was chosen as the output simulator parameter to study. Multiple runs of the simulator were performed to determine the mean throughput for both audio and object synchronisation traffic, their sample deviation and a confidence interval of 95%.

The comparison between previous experimental and simulation results showed that they have a considerable degree of correlation. The simulator slightly overestimate the object synchronisation throughput but its response to the increase in the number of users is similar to the experimental results. The simulator also overestimate the audio throughput but it still gives a better estimation than the theoretical maximum audio throughput.

The most immediate future work would be describing more accurately the relationship between the number of concurrent clients within an OWL session and the value of the rate parameter λ that determines the packet generation of those clients. This would simplify the specification of the input parameter of the simulator. Thus, the tool would only need the number of concurrent users to automatically infer the proper λ value.

Further future work would focus on the study of other network parameters that have a direct impact on the user experience, such as packet loss, packet delay or the influence of the underlying network topology and technology (Gigabit Ethernet, Wifi, DSL, etc.). The simulation presented in this paper provides the ground for a modular, expandable and configurable simulator network that would prove of great help for the study of the parameters cited above and could be also exported to other fields that require similar network evaluation approaches.

Acknowledgment

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.

References

[1] B. Fogg, *Persuasive technology: using computers to change what we think and do*, ser. Morgan Kaufmann series in interactive technologies. Morgan Kaufmann Publishers, 2003.

[2] S. Romero, L. Fernandez-Luque, J. Sevillano, and L. Vognild, "Open Source Virtual Worlds and Low Cost Sensors for Physical Rehab of Patients with Chronic Diseases," *Lecture Notes of the Institute for Computer Sciences Social-Informatics and Telecommunications Engineering*, vol. 27, pp. 84–87, 2010.

[3] 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.

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

[5] S. Ratti, B. Hariri, and S. Shirmohammadi, "A Survey of First-Person Shooter Gaming Traffic on the Internet," *IEEE Internet Computing*, vol. 14, no. 5, pp. 60–69, Sep. 2010.

[6] 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.

[7] T. Fritsch and H. Ritter, "The effect of latency and network limitations on MMORPGs: a field study of everquest2," *ACM SIGCOMM workshop on Network*, 2005.

[8] D. Gracanin, Y. Zhou, and L. DaSilva, "Quality of service for networked virtual environments," *Communications Magazine, IEEE*, vol. 42, no. 4, pp. 42–48, 2004.

[9] T. Henderson and S. Bhatti, "Networked games: a QoS-sensitive application for QoS-insensitive users?" in *Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?* ACM, 2003, pp. 141–147.

[10] E. Asensio, "Analyzing the network traffic requirements of multiplayer online games," *Advanced Engineering Computing and Applications in Sciences, 2008. ADVCOMP'08. The Second International Conference on*, pp. 229–234, 2008.

[11] J. Font, D. Cascado, J. Sevillano, G. Lopez, S. Romero, and G. Jimenez, "Network requirements evaluation of a multi-user virtual environment," in *Performance Evaluation of Computer Telecommunication Systems (SPECTS), 2011 International Symposium on*, june 2011, pp. 90 –97.

[12] 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, submitted to SIMPAT.

[13] M. Borella, "Source models of network game traffic," *Computer Communications*, vol. 23, no. 4, pp. 403–410, Feb. 2000.

[14] J. L. Font, P. Iñigo, M. Domínguez, J. L. Sevillano, and C. Amaya, "Architecture, design and source code comparison of ns-2 and ns-3 network simulators," in *Proceedings of the 2010 Spring Simulation Multiconference*, ser. SpringSim '10. New York, NY, USA: ACM, 2010, pp. 109:1–109:8.

[15] J. Font, P. Iñigo, M. Domínguez, and C. Sevillano, J.L.and Amaya, "Analysis of source code metrics from ns-2 and ns-3 network simulators," *Simulation Modelling Practice and Theory*, vol. 19, no. 5, pp. 1330–1346, 2011.