Network traffic analysis and evaluation of a multi-user virtual environment

Juan L. Font, Daniel Cascado, José L. Sevillano *, Fernando Díaz del Río, Gabriel Jiménez

Department of Computer Technology and Architecture, University of Seville, Seville, Spain

A B S T R A C T

Virtual world applications allow users to interact within a simulated world. Network responsiveness and reliability contribute to the user experience, thus being able to model and reproduce certain network scenarios is a key issue to assure proper user experience and for being able to provide an estimation of the required network resources. The present study aims to model the client network traffic for the virtual world application Open Wonderland as the basis to tools for evaluating its network requirements. A micro scale modelling was performed, studying the outgoing network traffic from a black box approach that omits the details of traffic generation of the subcomponents and focuses on their overall combined traffic. The model obtained provides high goodness of fit for audio and object synchronisation traffic, reflected in a Pearson correlation coefficient close to its maximum value and low deviation figures measured by Root Mean Square Deviation.

Keywords:
Virtual world
Open Wonderland
Network traffic
Micro scale model

1. Introduction

Nowadays, the virtual world concept has become familiar to wider audiences thanks to applications such as Second Life or World of Warcraft. Although their focus ranges from gaming to social interaction, they all base the user's experience on a virtual world environment, a distributed simulation shared by several users.

Given the acceptance of virtual world paradigm and following the recent proliferation of the so-called "persuasive systems" focused on motivating healthy lifestyle habits [1], our research group has developed a persuasive system called "Virtual Valley", relying on the virtual world concept [2,3]. Virtual Valley is based on Open Wonderland, a Java open source software for creating collaborative 3D virtual worlds (also known as Collaborative, Networked or Distributed Virtual Environments).

Open Wonderland was originally conceived as a tool for collaborative working by Sun employees [4], and as such it has some characteristics that make it very interesting for our application: focuses on social interaction and communications; open platform that allows new developments; can be installed and used by organizations within their own infrastructure, without the cost of renting a virtual space on a third party server and also allowing control of private medical data; etc. [5].

The latest version of this client–server architecture is Project Wonderland 0.5, shown in Fig. 1 [6]. Open Wonderland is subdivided in several independent subprojects listed below [7]:

- Wonderland: comprises both the core of OWL client and server as well as a set of modules that provide key functionalities such as security, shared applications, avatars and so on. It also contains the web administration server. Specifically, the shared application feature allows sharing applications among different users. Some of these applications are already integrated in Wonderland, like the multi-user PDF Viewer and the SVG White board. But users can also share additional external applications installed in the server (like Firefox or OpenOffice) using the Shared Applications Server-SAS.

* Corresponding author. Tel.: +34 954556142.
E-mail address: sevi@atc.us.es (J.L. Sevillano).

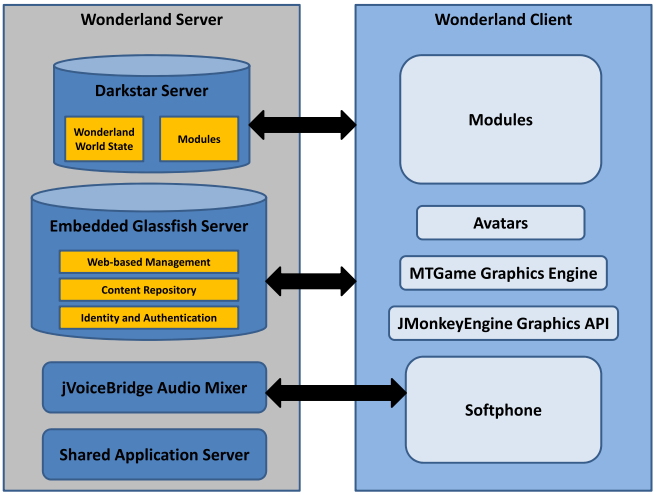

Fig. 1. Wonderland client–server architecture.

- Wonderland Modules: repository for Wonderland extension modules which expand its functionalities. Some of these modules are shipped by default within the OWL binary releases. There are also experimental modules provides by OWL developers and the community.
- MTGame Graphics Engine: high-performance graphics engine that extends jMonkeyEngine. MTGame adds multi-threading capabilities for improved graphics performance.
- jVoiceBridge: pure-java audio mixing platform providing real-time immersive audio (via VoIP) with time distance attenuation and a selectable range of qualities as well as a companion software phone, called *softphone*' that allows phone calls between users within the virtual world. It supports mixing high-fidelity, stereo audio at up to CD quality. Open Wonderland also depends on several several open source projects:
- Darkstar: Java platform started by Sun Microsystems for scalable communications and persistence in games. OWL includes a Darkstar service that manages all client and world state. Nowadays the project development is officially halted but a community fork has been create, called RedDwark Server [8].
- Glassfish: highly scalable, open source pure-Java application server that provides several functionalities for the Java EE platform such as RMI, XML and web server. OWL is based on an embedded instance of the Glassfish server. Wonderland web applications include web-based management of the server and worlds, a content repository for hosting all world data, and an integrated single-sign on system used to maintain identity across Wonderland services.
- jMonkeyEngine: 3D game engine written entirely in Java. JME provides core graphics APIs, including graphics primitive and shader support. The Wonderland graphics system is based on these core APIs, with some extensions from MTGame to support multithreading.

In Wonderland, like in any other virtual world, the user is represented by a 3D object known as an avatar. There can be many other objects in the virtual world, which can be 3D objects like pieces of furniture, buildings, etc.; or 2D objects like screens with applications (web browsers, word processors, and so on). The usual way to model the spatial relationships between objects is using a scene graph. Each object is a node (or *Cell* in Wonderland terminology) in this graph. The *Cells* (representing any volume of space of the virtual world) are organised in a graph with a tree hierarchy [9]. An example of such a tree is shown in Fig. 2.

As it will be discussed in Section 3, 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 aim is twofold: on the one hand, several gaming sessions will be performed, increasing the number of concurrent players between them, and the network traffic will be captured and analysed to obtain a preliminary model of the client network traffic. On the other hand, this model will be the basis for future simulation aimed to test its accuracy and determine some OWL network related parameters such as scalability and needed network resources.

The motivation for a detailed study of the OWL client traffic is to define a micro scale model as a first step to create tools for the evaluation of OWL network requirements. Part of the OWL traffic depends on user activity, a random process that implies variable bandwidth. Simpler network models based on parameters such as long run bandwidth lack the necessary

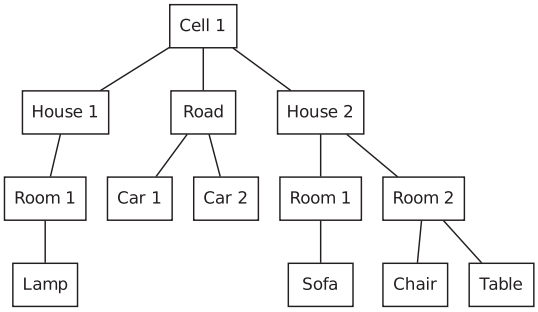

Fig. 2. An example of a tree representing a Wonderland cell.

degree of detail and may be insufficient to make an accurate estimation of requirements. In this context, the micro scale modelling proposed is a valid approach for the traffic description. This paper is the first milestone at creating these network estimation tools, defining the models that will be their basis.

The rest of the paper is organised as follows. The next section reviews the related previous works in the literature. Sections 3 and 4 describe the Open Wonderland v0.5 communications architecture and the methodology and testing environment, respectively. Object synchronisation traffic is discussed in Section 5 and audio traffic in Section 6. Finally, Section 7 presents the conclusions and future works.

2. Previous work

There are not many works in the literature dealing with the network traffic and resources needed to support the execution of Networked Virtual Environments (NVE). A review of architectures and approaches to provide Quality of Service (QoS) for NVEs can be found in [10]. Some QoS experimental results are provided in [11]. In [12] the effect of network latency with a high number of on-line players is studied. In [13] the network bandwidth requirements of some popular multi-player online games are experimentally measured by monitoring the network traffic generated by different game tournaments in a LAN Party. In [14] a micro scale modelling is described.

Most of the studies available are focused on multi-player online games, with special emphasis on First Person Shooters (FPS) [15,16] such as Counter-Strike/Half-Life [17], Quake [18–20] or Halo [21,22]. In a few references there are also studies dealing with network traffic for Real-Time Strategy (RTS) such as Starcraft [23] and Warcraft III [24] and Massively Multiplayer Online Role-Playing Games (MMORPG) [12].

There are several aspects that differentiate OWL from the FPS genre. The trend in this genre is the use of UDP as transport protocol for both object synchronisation and audio transmission, benefiting from its lighter overhead while a certain packet loss can be tolerated without significantly degrading the user experience [15]. On the other hand, the user experience in MMORPG and RTS games can be more sensitive to packet loss than FPS so the use of TCP protocol to assure the transmission of certain events is not uncommon [12]. OWL relies on a hybrid model using TCP connections for object synchronisation traffic and UDP for audio transmission. While FPS are aimed at fast paced action [25], OWL contemplates certain degree of action and multi-user interaction, but their rate and nature are not totally comparable to the FPS ones, so OWL falls into the slow paced game category.

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 [26]. The results obtained in this paper will be useful when designing and implementing these systems, mainly in applications, like e-health systems, with specific characteristics like a minimum level of dependability, timeliness of some critical messages (e.g. alarms in case of falls, altered medical parameters, etc.), limited resources (bandwidth, etc.), and so on.

3. Open Wonderland v0.5 communications architecture

Wonderland is based on a client–server architecture [27]. The Wonderland server must have a fixed, public IP address to which clients can connect. Initially, a Client is disconnected from the server until it calls a *WonderlandSession.login()*. If this succeeds, the session goes into the CONNECTED state. A client may connect multiple sessions to the same server. Once a session is connected to a server, connections may be added to it. Each connection in Wonderland has a unique type for sending different types of data. For example, a client may use one connection for sending cell data, and another one for sending voice communications data. Clients may use as many connections as they need for their interaction with the server. The only limitation is that a client may only have a single connection of a given type connected to a given session. Clients may also use multiple sessions to get multiple copies of a single connection type.

Once in the CONNECTED state, messages can be passed from a client to the server, from the server to one client (not necessarily in response to a client message or request) and also the same message can be sent by the server to a number of

clients. From version 0.5 on, there is no more client–client communication, that is, all messages from clients go directly to the server. The traffic between clients and server/s can be broadly divided into the following categories:

3.1. Object synchronisation

Objects in the virtual world have a set of attributes that can have different values: for instance, coordinates in the virtual world, velocity (for moving objects), etc. The state of each object (or cell) is defined as the values of these attributes at a given time. In Wonderland, the server keeps a copy of all the world data, with data about the cells stored in XML files. When a client connects to Wonderland, it obtains from the server the information about the visible objects (cells). These objects can be classified as static (with fixed attributes that do not vary in time, as it is the case of a mountain or a building) or dynamic (with attributes that can vary in time, like an avatar moving from one region to another) [28]. Every client should have a consistent view of the virtual world, and therefore a mechanism must exist to ensure synchronisation between clients. When an object moves in the virtual world, all clients that want to view the dynamic object must provide the same sequence of state changes. In Wonderland, if anything changes (such as the position of an object), this data is updated on the server and then sent to all the clients, who then update their own local views of the world. For instance, if a client moves his/her avatar, the client notifies the server and sends the new state of the cell to the server. The server then sends the new state of this object to all other clients, which then update their copies of the cell [27].

A typical sequence is as follows: a client sends a MOVE_REQ message to the server when the position of an object changes. Then the server sends a MOVED message to every other client, one per client. An ACK message is then sent back to the server by every client. The server does not send any other MOVED message until the corresponding ACK has been received. This sequence has been confirmed by several experiments.

Although these MOVE_REQ object synchronisation messages are sent every time the object moves, in principle they are limited by the MoveableComponent [29]. According to the Wonderland Project Forum [29], this limit is 5 updates per second, although this statement could not be confirmed in our experiments as we will discuss later on. Object synchronisation messages are sent by default through port TCP 1139, and their length varies between 293–771 bytes, depending on the needed information about the object position, orientation, identification within the virtual world, etc. There are also other protocol messages (ack, presencemanager, audiomanager, etc.) that contribute very little to the TCP traffic. Although all this information can be obtained from the Wonderland documentation, it is not easy to model the actual workload in a real setting as it depends on what kind of movements or how often the avatar moves. There may also be some differences depending on the configuration of the client platform. That is why several experiments (detailed in Section 5.3) have been performed to try to evaluate the actual network workload using different Operating Systems and a different number of clients (see Subsection 4.2).

3.2. Audio

In Wonderland, the standard conversation allows users to speak and hear each other depending on the distance between them. But in addition, users can also initiate a voice chat session with other users. This conversation can be private but can also be made public. The voice functionality can be extended depending on the application. For instance, in [30] a system for virtual meetings is described, where voice communications between users are supported even if a user is not present in the virtual world (using a Virtual Phone for real-time communication). The present study uses the standard Wonderland voice functionality offered by jVoiceBridge, based on the standard SIP (Session Initiation Protocol) and RTP (Real-time Transport Protocol) protocols to transmit voice data from the Wonderland server to the various clients. jVoiceBridge uses a single UDP port for all control data, and an additional two UDP ports per call connected. The UDP control port is used by SIP, and UDP port 5060 by default [31].

In Wonderland, a conference has members who can talk to each other. Calls are the individual parts of a conference, and data from all calls in the conference are added into a "common mix". Data samples from every member are added together to create the output from the server to the clients. Each client's audio must be subtracted out of the common mix when the data is sent to that client, so that the user does not hear him/herself when he/she talks [32]. Therefore, the traffic from the server to the clients has to be sent as different streams. As a result, the voice traffic increases linearly as the number of members in a conference increases.

jVoiceBridge handles audio at three different fidelities:

- 8k ulaw, 8-bit per sample, 8000 samples per second. This means 64 Kbit/s.
- 16k PCM, 16-bit per sample, 16,000 samples per second. This means 256 Kbit/s.
- 44.1k PCM, 16-bit per sample, 44,100 samples per second. This means 705,600 Kbit/s.

The voice system sends packets every 20ms, or 50 times per second. In addition, the client may send/receive a mono or stereo stream to/from the voice bridge. For example: 16k PCM, 16-bit, stereo stream has a bandwidth of 256 Kbit/s = 64 KB/s. This bandwidth divided by 50 audio packets is 1280 byte/packet. Finally, the packet header due to LLC layer, IP, UDP and RTP protocols is a total of 52 bytes per packet, which adds up to the total payload calculated above.

4. Methodology and testing environment

The methodology followed is similar to that described in [14]. The Inter-Arrival Time (IAT) between consecutive generated packets and packet size has been chosen as a study parameter to achieve a micro scale modelling of the OWL client network traffic.

The first step is a preliminary study of the parameter by calculating the Empirical Cumulative Distribution Function (ECDF) and evaluating its graphical representation to determine the best probability distribution to model the empirical data.

Depending on this first evaluation, an analytical expression will be determined: it may be simply deterministic or a common probability distribution. In those cases where an important divergence between the empirical and proposed models is observed, a split distribution may be required.

Once the probability distribution has been chosen, its parameters, if any, are calculated using Maximum Likelihood Estimation (MLE) [33,34]. Any needed data filtering will be also described, evaluating its impact over the fitting results.

Finally, depending on the model proposed, it will be accompanied by correlation and deviation estimators to evaluate the goodness of fit, as well as plots that allow to visually estimate the fitting, such as Q-Q plots that compares the empirical data against the analytical model proposed [35].

4.1. Data filtering

The filtering criteria for the different types of OWL traffic are as follows. Object synchronisation traffic is composed of TCP packets that can be identified by certain tokens contained in their payload. Specifically, the update request contains "MOVE_REQUEST" tokens and their confirmation packets the "MOVED" token. Audio traffic is composed of UDP packets. Unlike object synchronisation traffic, these packets do not contain any specific traffic, so they are identified based on whether their IP source or destination port is contained in the port ranges defined for audio traffic.

The game traffic associated to user login and game initialisation have been ignored as they are of little interest for the current study.

Regarding packet sizes, only their payload have been considered, overhead bytes due to the different transport protocols, TCP and UDP, have been omitted.

4.2. Testing environment

Several gaming sessions were performed in a controlled OWL environment. The server machine runs OWL v0.5 nightly build, corresponding to June 30th 2011. The OWL instance was deployed onto a LAN environment based on a single Ethernet switch 10/100. The use of wired technology and ISO Layer 2 device aims to provide an optimal network scenario, minimising network delay and removing any extra traffic not related to OWL. Thus, the whole bandwidth of the network is available for the application.

All the traffic was captured by using the packet analysis tool Wireshark v1.6.1. The tool was executed on the server side, sniffing all the incoming and outgoing traffic.

Three independent gaming sessions were performed, using 2, 3 and 5 concurrent clients respectively. The traffic captures were started after all the clients had logged into the gaming session and their views of the virtual world were already loaded. Thus, packets associated to the initialisation processes were avoided. These packets are only present during the initial stage of the gaming session to provide a copy of the virtual world environment to each client, and are not significant for the stationary scenario. During all the gaming session, all the players have intentionally performed a high activity rate, moving all the time to maximise the client traffic.

Tables 1 and 2 give information about the hardware resources and software configuration of each client.

5. Object synchronisation traffic

This section focuses on object synchronisation traffic generated by OWL clients. The following sections deal with a preliminary study of the distribution of IAT values throughout the sessions, the influence of user activity over the IAT, propose an IAT model and make an estimation for its parameters. Finally, the packet size for this kind of traffic is modelled.

Table 1
Hardware specifications of the testing machines.

Role	Processor	RAM	GPU
Server	AMD Phenom x4 2.6 GHz	4GB DDR2	GForce 8200
Client 1	Intel Dual Core x2 3 GHz	2GB DDR2	Radeon X300
Client 2	Intel C2D x2 2.53 GHz	3GB DDR2	GMA 4500 MHD
Client 3	Intel C2D x2 2.5 GHz	4GB DDR2	GeForce 8600 GT
Client 4	Pentium IV HT 3.2 GHz	512MB DDR	Radeon 9700
Client 5	Intel C2Q x4 2.4 GHz	6GB DDR3	GForce 9700

5.1. Preliminary study of the empirical packet IAT

The preliminary study of Inter-Arrival Time (IAT) reveals similar patterns for all their outgoing object synchronisation traffic. An example of this behaviour can be observed in Fig. 3, which shows the Empirical Cumulative Distribution Function (ECDF) for two clients from different testing sessions. Specifically, the ECDF plots belong to client 2 from the 2-client session and client 3 from the 5-client session. Details about these clients and testing environment are shown in Tables 1 and 2. Each ECDF curve in Fig. 3 is represented by solid line and, accompanied by a dashed one. The latter is the plot of a Cumulative Distribution Function (CDF) $F(x)$ for an exponential distribution calculated by Maximum Likelihood Method (MLE) and suggests an exponential nature of the IAT values below 0.5 s. The rate parameter λ determines the exponential behaviour and can be interpreted in the IAT context as the number of packets per second generated by a OWL client.

The ECDFs for the rest of clients, not attached due to space, follow patterns similar to those in Fig. 3. All the clients' ECDFs show two differentiated sections divided at 0.5 s.

Values below 0.5 s seem to follow an exponential distribution. This exponential nature is plausible if we assume the fact that outgoing client traffic can be considered as a Poisson Process where the waiting time between consecutive packets follows an exponential probability distribution [36]. Fig. 3 includes the CDF (dashed line) for the fitted exponential distributions calculated by Maximum Likelihood Estimation (MLE). The rate parameter λ for each fitted exponential curve are 4.52 and 2.68 respectively, both approximations show a good degree of fitness for values below 0.5 s. Although they are rudimentary fitting distributions, they support the hypothesis of an exponential distribution of some of the IAT values.

There is a probability saturation around 0.5 s with an important percentage of packet IAT observations concentrated around a narrow range centred on this value. This percentage varies between clients; in the example shown in Fig. 3 the ECDF from Client 2 has a lower step around 0.5 s than that of Client 3. In both cases, values greater than 0.5 s are relatively unlike and their frequency also varies between clients. This suggests that during high activity and continuous movement periods, the client tends to fix the time between consecutive requests to 0.5 s to avoid an overflow of synchronisation packets. During inactive periods, the IAT values increase due to the lack of updates.

Values greater than 0.5 s represent only a small percentage that varies between clients and sessions. Subsection 5.2 discusses the nature of these values and their relationship with user activity during the gaming session.

Although the ECDF curves have very similar shapes, they are not completely the same. The differences in the IAT distribution depends on session parameters such as user activity, number of players or client resources. Their influence will be discussed later.

The OWL technical information suggested a similar behaviour for the object synchronisation IAT. According to the information published in the OWL community forum [29], OWL clients limit the synchronisation traffic to avoid rates greater than five packets per second. This would imply a minimum IAT value of 0.2 s. However, we found that this value is not the most common in the testing results. Instead, there are values below 0.2 s following an exponential distribution and a probability saturation around 0.5 s. A specific explanation for such a behaviour is out of the scope of this study.

5.2. Impact of the client activity over the packet IAT

The previous section mentioned the low percentage of values below 0.5 s and its variability between clients. The OWL documentation suggests that there is a relationship between the user activity and the percentage of high IAT values. Inactivity periods would imply the absence of object synchronisation traffic and therefore high IAT values. On the other hand, active clients generate synchronisation traffic constantly, only limited by the traffic limitations suggested above, so high waiting periods and high IAT values would be quite unlike. Experimental results suggest that 0.5 s is the bound that separates traffic due to high or low activity rates.

Determining more accurately the impact of the client activity on the IAT distribution requires a testing environment where the rest of session parameters remain constant. For this reason two extra single-player sessions were performed in identical environments, only changing the activity rate of the players. The first client moved constantly, while the second one stood inactive for long periods of time. According to the figures shown in Table 3, the inactive player has a noticeable percentage of IAT values equal or greater than 0.5 s (11%), while the active player has few values over 0.5 s, less than 1%. This table contains the values from 89th to 99th percentile from each of the IAT data sets from the single-player sessions.

Table 2
Software specifications of the testing machines.

Role	OS	Architecture	JDK
Server	Debian 6.0 (testing)	64-bits	Sun JDK 1.6.0
Client 1	Windows XP Prof.	32-bits	Sun JDK 1.6.0
Client 2	Windows XP Prof.	32-bits	Sun JDK 1.6.0
Client 3	Windows 7 Prof.	64-bits	Sun JDK 1.6.0
Client 4	Windows XP Prof.	32-bits	Sun JDK 1.6.0
Client 5	Windows 7 Prof.	32-bits	Sun JDK 1.6.0

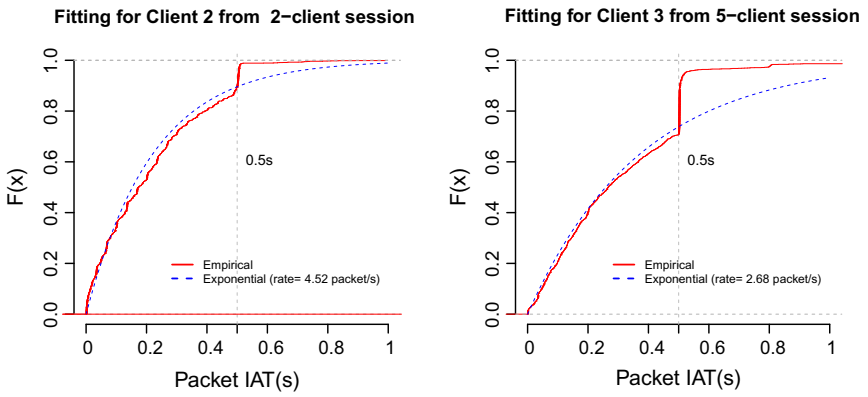

Fig. 3. Packet IAT ECDF for two OWL clients.

The ECDFs from the single-player traces differ in the distribution of values around 0.5 s; Fig. 4 shows their plots. Each ECDF plot is accompanied by two exponential CDF. The one labelled as "Raw Fitting" (dashed line) results from applying Maximum Likelihood Estimation (MLE) to the whole set of IAT values from each one of the single-player sessions. The "Filtered Fitting" CDF (dotted line) results from applying MLE to the set of IAT values after removing those over 0.5 s. The inactive client shows "Raw" and "Filtered" fitting distributions that are very different between them ($\lambda_{RAW} = 0.776$ and $\lambda_{FILT.} = 4.66$), while those two for the active client are very similar ($\lambda_{RAW} = 4.12$ and $\lambda_{FILT.} = 4.08$).

According to the previous conclusions, percentiles from Table 4 calculated from the multi-player traces show that 96% of all the IATs were below 0.56 s, as it can be observed in column $Q_{0.96}$, and 98% were below 1 s. Although the IAT distribution varies between clients, their percentages of high IAT values are consistent with a high user activity. This player behaviour was intentionally performed during the testing sessions.

Given the relationship between the user activity and the upper tail of the IAT distribution, it is necessary to determine a threshold in order to identify these upper tail values. A narrow range around 0.5 s was studied to determine this "activity threshold", t_{ACT}. The data set used to determine t_{ACT} is composed of the union of all the packet IAT values from the testing multi-player sessions. The range around 0.5 s comprises values within [0.45, 0.55]. The width of the range was set manually according to the graphical representation of the Probability Density Function (PDF) shown in Fig. 5. The threshold $t_{ACT} = 0.523$ s comprises 99% of those IAT values in Fig. 5, which means that 99% of all the values within [0.45, 0.55] are less or equal than $t_{ACT} = 0.523$ s. The median of the data set (0.501 s) was highlighted and will be discussed later for modelling purposes.

5.3. IAT model proposed for object synchronisation

Section 5.1 shows the division of IAT values into two ranges. A split distribution may account for this behaviour, divided in exponentially distributed values, a probability saturation around 0.5 s and a low percentage of values above this value. Assembling three models into one single split distribution may produce a clumsy expression [14]. On the other hand, Section 5.2 shows the relationship between the IAT upper tail and inactive periods. High IAT values due to inactivity are of little

Table 3
Packet IAT Quantiles for active and inactive clients.

Quantile	Inactive client(s)	Active client(s)
$Q_{0.89}$	0.5010	0.5003
$Q_{0.90}$	0.6840	0.5003
$Q_{0.91}$	0.7609	0.5004
$Q_{0.92}$	0.9104	0.5005
$Q_{0.93}$	0.9913	0.5006
$Q_{0.94}$	1.1978	0.5007
$Q_{0.95}$	2.2352	0.5008
$Q_{0.96}$	5.7679	0.5009
$Q_{0.97}$	11.4161	0.5010
$Q_{0.98}$	17.8715	0.5011
$Q_{0.99}$	32.0450	0.5016

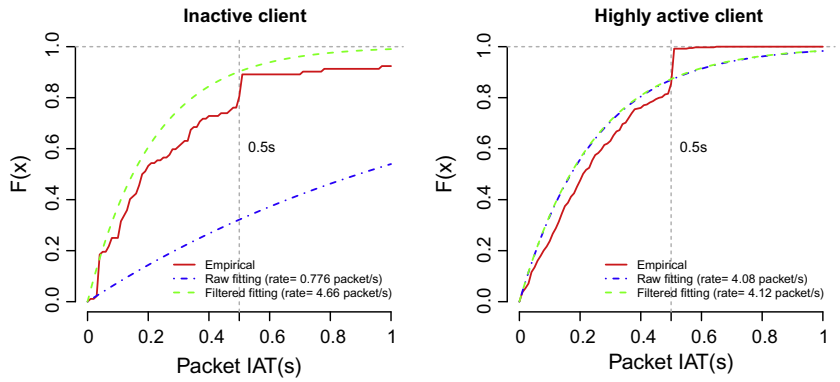

Fig. 4. Packet IAT ECDF for different activity rates.

Table 4
Packet IAT Quantiles for *n*-client sessions.

Session	Client	$Q_{0.95}$	$Q_{0.96}$	$Q_{0.97}$	$Q_{0.98}$	$Q_{0.99}$
2-client	01	0.5030	0.5040	0.5080	0.5141	0.6545
	02	0.5038	0.5048	0.5061	0.5081	0.6208
3-client	01	0.5058	0.5072	0.5100	0.5224	0.7631
	02	0.5046	0.5056	0.5073	0.5120	0.6931
	03	0.5028	0.5176	0.6980	0.8027	0.8398
5-client	01	0.5068	0.5098	0.5127	0.6419	0.7600
	02	0.5049	0.5064	0.5082	0.5119	0.6682
	03	0.5211	0.5510	0.7350	0.8051	1.6982
	04	0.5117	0.5139	0.5177	0.6530	0.7644
	05	0.5014	0.5030	0.6203	0.7957	0.8039

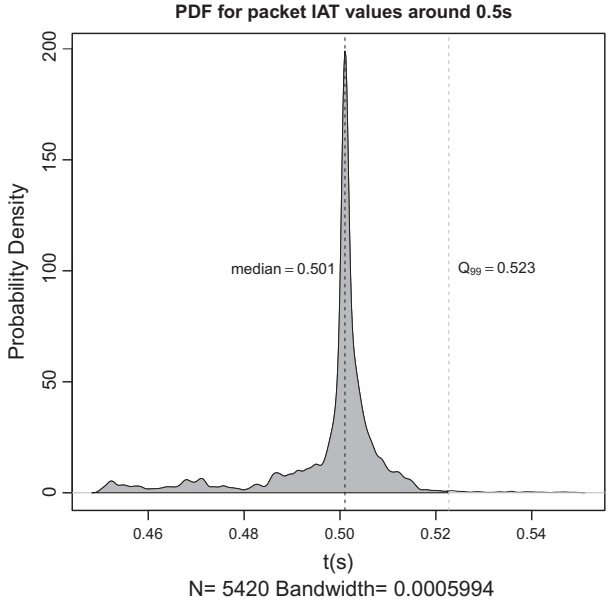

Fig. 5. PDF for packet IAT values $\in [0.45, 0.55]$.

interest to our model from the point of view of networking requirements, so the inactive client case will not be considered any more in this study. Future works can focus on modelling such a behaviour.

Ignoring these high IAT values implies removing the ECDF upper tail. The threshold value, $t_{ACT} = 0.523$, calculated in Section 5.2 will be used to distinguish "activity" from "inactive" derived traffic, the later will be ignored. This simplification also allows the use of a truncated distribution to model IAT values.

The threshold t for the truncated distribution can be visually determined around 0.5 s. The range associated to Fig. 5 from Section 5.2 was used to determine this value more accurately. The median is an estimator less sensitive to the tails so it has been chosen as an estimator of this range and as t threshold, resulting in $t = 0.501$. Therefore, we can safely assume that the truncated exponential has its threshold at 0.5 s.

Eq. (1) describes the model proposed for synchronisation IAT. It remains to delimit the range of values for the rate parameter λ and determine the goodness of the fitting of the expression.

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

5.4. Determination of the exponential rate and goodness of fit

The parameter rate λ from the IAT model in Eq. (1) determines the behaviour of the model proposed. The preliminary study of the IAT ECDFs for different gaming sessions revealed that sessions with the same number of players have similar ECDFs. Thus, the number of concurrent users would determine the distribution of the IAT values. Three values for λ were calculated, one for each multi-player session. These rate parameters provide the best goodness of fit of the IAT model for the 2, 3 and 5-client sessions.

Maximum Likelihood Estimation (MLE) was used to determine the λ value that provides the best fitting for the IAT model. This value was calculated using Least Squares as MLE estimator, which is used to minimise the summation of the square of the distances between the empirical and theoretical values. Eq. (2) describes the function whose minimisation returns λ, where y_i is the x_i probability returned by the data ECDF, $y(x_i$ lambda) is the x_i probability returned by the model $F(x|\lambda)$.

$$\sum_{i=1}^{n} (y_i - y(x_i|\lambda)^2) \tag{2}$$

The goodness of fit was determined using two metrics. The Pearson's correlation coefficient, where r measures the trend relative magnitude of the fit [37,38]. A r_{xy} value equal to 1 indicates that the equation describes the relationship between x and y perfectly, while a r equal to 0 implies the lack of correlation between empirical data and the model. The Pearson's r is accompanied by r^2, which indicates the proportion of the total variance that is explained by the prediction. The deviation from the exact data location has been measured by Root Mean Square Deviation (RMSD) [39,40], described in Eq. (4).

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{3}$$

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(y_i - y(x_i|\lambda))^2}{n}} \tag{4}$$

The MLE and goodness of fit results are shown in Table 5. λ values decrease when the number of concurrent players increase in the gaming sessions although the user activity do not experience significant variations. Regarding the goodness of fit, the correlation between the model and the data sets is close to 1, which implies that the model and different λ values give a good description of the relationship between the IAT values and their probability.

Although there are not enough data sets to make a detailed study of the evolution of λ values, they are within the range [3.1,4.2] and their evolution is not linear with respect to the number of players in the gaming session. Values close to $\lambda = 4.2$ model the inter-arrivals for a highly active client within a gaming session with 2 or 3 concurrent players. On the other hand, values near $\lambda = 3$ describe a highly active client involved in a gaming session comprising 5 or more concurrent players. This may be explained by the fact that an increase of users and its associated data traffic involve an increase of the overall workload in the OWL system. The server must deal with a higher rate of updates and their propagation while clients have their packet rate and inter-arrivals penalised. A detailed study about the influence of the number of users over the packet rate and overall OWL performance will be the subject of future work.

Table 5
MLE and Goodness of Fit figures for synchronisation IAT.

Session	λ	r	r^2	RMSD
2-client	4.168	0.9981	0.9962	0.0219
3-client	3.687	0.9973	0.9946	0.0340
5-client	3.146	0.9946	0.9893	0.0481

In addition to the metrics to evaluate the model fit, there are several graphs in Fig. 6 that support the quality of the fit. Left column of graphs in Fig. 6 shows the IAT ECDF for each multi-player session (solid line). Each graph also displays the CDF for the model in Eq. (1) (dashed line), using the λ from Table 5 calculated to its respective session. Right column in Fig. 6 shows the corresponding Q–Q graphs, which displays a very good correlation between the empirical and theoretical quantiles provided by the IAT model.

5.5. Object Synchronisation packet size

All the outgoing client traffic related with object synchronisation followed the same distribution of packet sizes. Table 6 shows a summary including the number of packets measured and the relative frequencies for the most common packet sizes. Only the packet payload was considered, the overhead due to TCP protocol represents about 54 B per packet and it is not

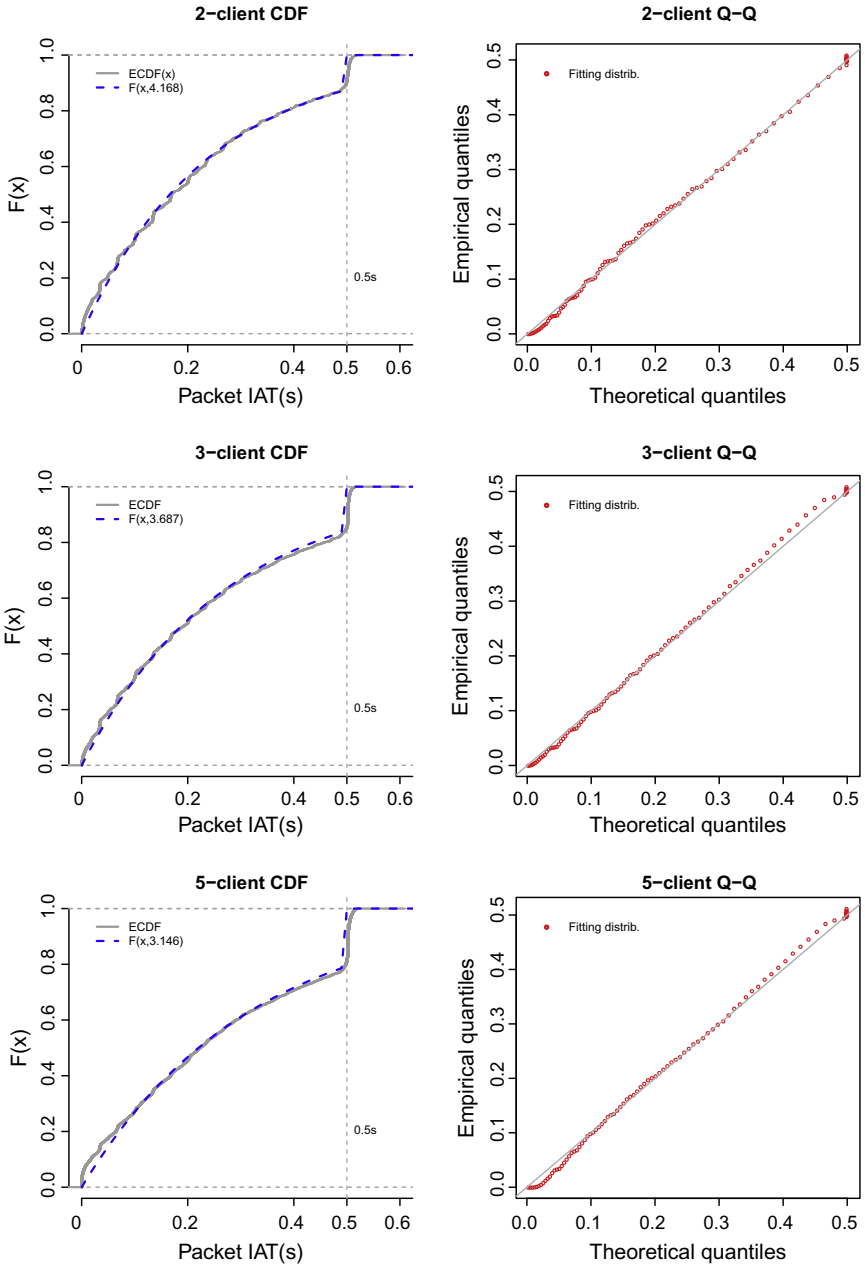

Fig. 6. Goodness of fit for the synchronisation IAT model and MLE λ values.

reflected in Table 6. The most common payload size is 239 bytes, comprising more than 96% of all the object synchronisation packets observed. The next three most common values account for only 3% of all the observations. Finally, the remaining observations are less than 0.5%.

Only the four most common size values were used to model the packet payload size behaviour, the remaining 0.45% corresponding to other payload sizes was omitted for simplicity. The probability for each payload size value was defined according to the empirical relative frequencies and fitted after removing the percentage of packets corresponding to "Other values" following a simple rule of three. The relative frequencies are calculated by multiplying the values shown in Table 1 by 0.9955, which is the sum of the relative frequencies for the four most common packet sizes.

The relative frequencies calculated above are used as probability values for the discrete random variable in Eq. (5), which describes the distribution of packet sizes observed in Table 6.

$$P(x) = \begin{cases} 0.9704, & x = 239 \text{ bytes} \\ 0.0195, & x = 478 \text{ bytes} \\ 0.0057, & x = 536 \text{ bytes} \\ 0.0044, & x = 717 \text{ bytes} \end{cases} \tag{5}$$

6. Audio traffic

Despite audio traffic generation is theoretically described in jVoiceBridge as a periodic process, a brief study of the audio traces captured was performed to compare the specifications with the testing results, derived from a more complex environment where jVoiceBridge interacts with the underlying operating system, network stack and other software entities.

Among the audio configurations available for OWL, the clients involved in all the test sessions used the 16 K PCM 16 bit stereo configuration. The jVoiceBridge is the component that transmits audio and, according to its documentation, it attempts to send audio samples each 20 ms, so the expected audio IAT values should be close to this value. The study of the experimental IAT values partially confirms the above statement. Although a significant percentage of audio IAT values concentrate around 20 ms, there are also a discrete set of values that gathers the rest of IAT observations. At this point it is necessary to make a distinction between Windows XP and Windows 7 clients.

On the one hand, audio IAT values for Windows XP clients are largely consistent with jVoiceBridge documentation, showing a high percentage of values concentrated around 20 ms. Fig. 7 displays the density function for the experimental results from all the Windows XP clients. While values around 20 ms are the most common, there is a significant percentage of observations distributed around other IAT values (highlighted by dashed vertical lines). The plot shows the local maxima for the IAT range $[0, 0.08]$, which comprises more than 98% of the IAT samples. The most significant maximum values were used to define the discrete random variable in Eq. (6) which models the IAT behaviour for audio packets in Windows XP.

On the other hand, audio IAT values for Windows 7 clients show a different distribution from the XP clients. Fig. 8 shows the density function for the IAT values measured for the Windows 7 clients during the testing gaming sessions. The local maxima were found and used to define the discrete random variable in Eq. (8) for Windows 7.

For simplicity, the audio IAT distribution was modelled as a discrete random variable. Due to the differences between Windows XP and 7 clients, a specific discrete random variable is proposed for each type of client. Defining a more comprehensive set of client operating systems is out of the scope of the present study.

$$P(X = k) = \begin{cases} 0.1426, & k = 0.0114 \text{ s} \\ 0.6405, & k = 0.0221 \text{ s} \\ 0.0723, & k = 0.0340 \text{ s} \\ 0.1446, & k = 0.0442 \text{ s} \end{cases} \tag{6}$$

$$E[X] = \sum_{i=1}^{n} x * p_X(x_i) = 0.0246s \tag{7}$$

Table 6
Relative frequency of object synchronisation packet payload sizes.

Session	No. packets	239 bytes	478 bytes	536 bytes	717 bytes	Other
2-client	5099	0.9674	0.0212	0	0.0082	0.0032
3-client	7768	0.9649	0.0206	0.0050	0.0053	0.0042
5-client	12,286	0.9663	0.0180	0.0084	0.0022	0.0051
Total	25,153	0.9661	0.0194	0.0056	0.0044	0.0045

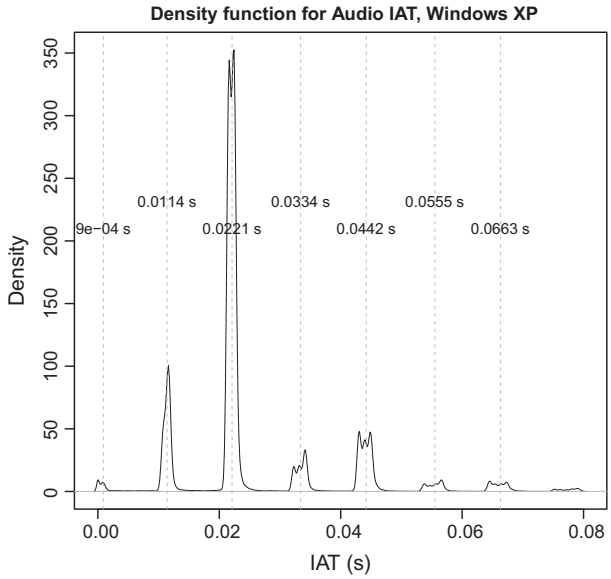

Fig. 7. Density function for Audio IAT, Windows XP client.

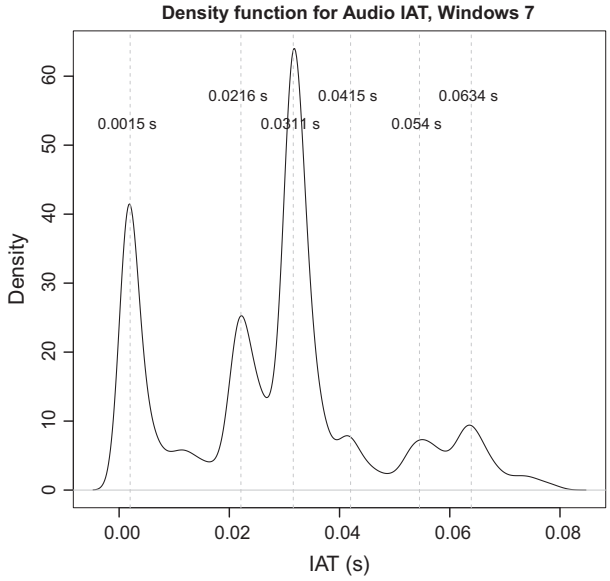

Fig. 8. Density function for Audio IAT, Windows 7 client.

$$P(Y=k)=\begin{cases}0.2342, & k=0.0015\text{ s}\\ 0.1820, & k=0.0216\text{ s}\\ 0.3656, & k=0.0311\text{ s}\\ 0.0709, & k=0.0415\text{ s}\\ 0.0576, & k=0.0540\text{ s}\\ 0.0896, & k=0.0634\text{ s}\end{cases} \tag{8}$$

$$E[Y]=\sum_{i=1}^{n} y * p_Y(y_i)=0.0273s \tag{9}$$

Table 7
Audio bandwidth.

Session	Client	OS	Time (s)	No. packets	Size (MB)	BW (KB/s)
2-client	01	XP	590.19	20,681	26.215	45.48
	02	XP	596.77	21,705	27.522	47.23
3-client	01	XP	696.70	25,053	31.760	46.68
	02	XP	697.05	25,083	31.805	46.72
	03	7	696.95	21,421	27.146	39.88
5-client	01	XP	782.14	31,596	40.078	52.47
	02	XP	782.10	28,543	36.196	47.39
	03	7	782.12	25,921	32.863	43.03
	04	7	782.13	28,639	36.322	47.55
	05	XP	770.21	28,267	35.923	47.76

Table 8
Relative frequency of audio packet UDP payload sizes.

Session	N. Packets	1292 bytes	467 bytes	481 bytes	52 bytes	Other
2-client	5099	0.9954	0.0000	0.0000	0.0009	0.0037
3-client	7768	0.9952	0.0010	0.0010	0.0010	0.0019
5-client	12,286	0.9950	0.0014	0.0014	0.0009	0.0014
Total	25,153	0.9951	0.0010	0.0010	0.0009	0.0020

Despite the different IAT distribution for Windows XP and 7 clients, their respective bandwidths remain similar regardless of the underlying operating system. The similarity between the client audio bandwidth can be observed in Table 7 which shows a summary of the audio traffic figures.

According to the technical specifications, an audio packet for the 16k PCM, 16-bit stereo configuration has a payload of 1280 bytes plus a header of 54 bytes, which means a total of 1334 bytes. In optimal conditions, an OWL client would generate an audio packet every 20 ms, which translates into a maximum theoretical bandwidth of 50 packet/s $*$ 1334 byte/packet = 66,700 byte/s = 65.14 KB/s. The Bandwidth column (BW) in Table 7 shows values below 65.14 KB/s which are consistent with the results shown in Figs. 7 and 8, where the clients are not sending audio packets every 20 ms. Instead, the sending rate is 0.0221 ms for XP and 0.0216 ms for Windows 7. This, combined with the existence of other probable IAT values, reduces the testing audio bandwidth.

The expectation for discrete random variable X ($E[X]$) in Eq. (6) is 0.0246 s. The expectation for random variable Y ($E[Y]$) in Eq. (8) is 0.0273 s. Despite the fact that they are different, their expectations have similar values. Using the expectation of the above discrete random variables and considering an audio packet size of 1334 bytes (including all the protocol headers), the expected audio bandwidths are 52.89 KB/s and 47.72 KB/s for Windows XP and 7 respectively (Eq. (10)), which is a more realistic approximation to the testing results than the theoretical audio bandwidth.

$$Bandwidth(KB/s) = \frac{PacketSize(B)}{E[X](s)} \frac{1KB}{1024B} \tag{10}$$

6.1. Audio packet size

The previous audio packet size estimation of 1280 bytes corresponds to the RTP payload which has a fixed 12-byte header. The RTP protocol analysed is built on UDP, so the UDP payload is 1292 bytes. Table 8 shows the most common UDP payloads observed during the testing sessions. Regarding this table, the packet size can be considered as fixed in 1292 bytes, this value comprises more than 99.5% of all the packet sizes measured. Thus, for modelling purposes, the audio UDP payload size is considered as fixed in 1292 bytes.

The RTP protocol allows defining the payload type in a field of its 12-byte header. RFC 3551 [41] defines a set of values associated to certain predefined audio/video configurations. Values ranging from 96 to 127 are defined as *dynamic* and their audio/video configuration and quality parameters can be defined according to the client application needs. In the OWL context, jVoiceBridge uses the identifiers 103 and 104 for the audio transmission depending on the context.

The study audio traces from extra single-client sessions reveal that in this specific situation, despite the stereo default configuration, the server and the only client negotiate a 16K PCM 16 bit mono audio stream, classified as RTP Payload Type 103, and described in the SIP REQUEST packets as Media Attribute 103 PCM/16000/1. This implies RTP payloads of 640 bytes, half the size of stereo ones. On the other hand, the multi-player sessions use 16k PCM 16 bit stereo streams, coded as SIP Payload Type 104 and described as PCM/16000/2; its associated RTP payload is 1280 bytes.

7. Conclusions and future work

The present study was focused on the two main traffic sources of Open Wonderland clients: audio and object synchronisation. Several network traffic traces were studied and used as a basis for modelling the packet Inter-Arrival Time (IAT) and size, following a micro scale approach.

On the one hand, the Inter-Arrival Time for object synchronisation packets has proven to be exponentially distributed for values below 0.5 s; at this point there is a probability saturation. Client inactivity periods determine the proportion of values above 0.5 s. The micro scale model proposed describes the IAT distribution accurately. The packet rate generation, determined by λ in the model, decreases as the number of concurrent users increases in a gaming session. The λ values calculated by Maximum Likelihood Estimation for several testing cases show a very high goodness of fit. The packet size for object synchronisation traffic takes only a discrete set of values, so it was modelled as a discrete random variable.

On the other hand, the IAT for audio traffic takes a discrete set of values that partly match the description provided by OWL documentation for the audio software components. There are differences in the packet IAT distribution depending on the client OS, although the client bandwidth remains similar regardless of the underlying OS. Audio IAT is not affected by the number of concurrent users or their rate of activity. Audio packets also show a fixed size which only depends on the selected audio quality.

IAT and size models and the empirical results lay the grounds for future work focused on the development, verification and validation of a network simulation tool, which will aim to evaluate the network requirements for OWL instances with an arbitrary number of concurrent users.

Further work may deal with a more comprehensive analysis of OWL client traffic behaviour, considering different rates of user activity or the inclusion of additional traffic sources, like the OWL integrated applications.

Acknowledgements

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, Morgan Kaufmann Series in Interactive Technologies, Morgan Kaufman Publishers, 2003.
[2] S. Romero, L. Fernandez-Luque, J. Sevillano, 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 27 (2010) 84–87.
[3] D. Cascado, S. Romero, S. Hors, A. Brasero, L. Fernandez-Luque, J. Sevillano, Virtual worlds to enhance ambient-assisted living, in: 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2010, pp. 212–215. doi:10.1109/IEMBS.2010.5627880.
[4] N. Yankelovich, W. Walker, P. Roberts, M. Wessler, J. Kaplan, J. Provino, Meeting central: making distributed meetings more effective, in: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, ACM, 2004, pp. 419–428.
[5] M. Gardner, A. Ganem, J. Van Helvert, J. Scott, C. Fowler, Designing and building immersive education spaces using project Wonderland: from pedagogy through to practice, 2008, pp. 1–8, jisc.ac.uk.
[6] O.W. Project, Open Wonderland Project, <http://www.openwonderland.org/> (accessed 15.03.11).
[7] O.W. Project, Open Wonderland Project: Developer Resources, <http://www.openwonderland.org/resources/developers> (accessed 30.12.11).
[8] R. Project, RedDwarf Project: about information, <http://www.reddwarfserver.org/?q=content/open-source-online-gaming-universe> (accessed 30.12.11).
[9] O.W. Project, Wonderland Tutorial, <http://code.google.com/p/openwonderland/wiki/OpenWonderland>.
[10] D. Gracanin, Y. Zhou, L. DaSilva, Quality of service for networked virtual environments, IEEE Communications Magazine 42 (4) (2004) 42–48.
[11] T. Henderson, 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.
[12] T. Fritsch, H. Ritter, J. Schiller, The effect of latency and network limitations on MMORPGs: A field study of Everquest 2, in: Proceedings of the Fourth ACM Network and System Support for Games (NetGames) Workshop (Hawthorne, NY, Oct. 10–11), ACM Press, New York, 2005.
[13] E. Asensio, Analyzing the network traffic requirements of multiplayer online games, in: The Second International Conference on Advanced, 2008, pp. 229–234. doi:10.1109/ADVCOMP.2008.15.
[14] M. Borella, Source models of network game traffic, Computer Communications 23 (4) (2000) 403–410, http://dx.doi.org/10.1016/S0140-3664(99)00197-8.
[15] S. Ratti, B. Hariri, S. Shirmohammadi, A survey of first-person shooter gaming traffic on the Internet, IEEE Internet Computing 14 (5) (2010) 60–69, http://dx.doi.org/10.1109/MIC.2010.57.
[16] P. Branch, G. Armitage, Towards a general model of first person shooter game traffic, Swinburne University of Technology, Tech. Rep. 050928A, 2005, pp. 1–11.
[17] W. Feng, F. Chang, 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.
[18] G. Armitage, An experimental estimation of latency sensitivity in multiplayer Quake 3, in: The 11th IEEE International Conference on Networks, 2003 (ICON2003), IEEE, 2003, pp. 137–141.
[19] A. Cricenti, P. Branch, ARMA(1,1) modeling of Quake4 Server to client game traffic, in: Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games – NetGames '07, 2007, pp. 70–74, doi:10.1145/1326257.1326270.
[20] T. Lang, P. Branch, G. Armitage, A synthetic traffic model for Quake3, in: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology – ACE '04 (cycle 1), 2004, pp. 233–238, doi:10.1145/1067343.1067373.
[21] T. Lang, G. Armitage, A ns2 model for the Xbox system link game HALO, Traffic 1 (2000) 3.
[22] S. Zander, G. Armitage, A traffic model for the Xbox game Halo 2, in: Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video – NOSSDAV '05, 2005, p. 13, doi:10.1145/1065983.1065987.

[23] M. Claypool, D. LaPoint, J. Winslow, Network analysis of counter-strike and starcraft, in: Conference Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, IEEE, 2003, pp. 261–268.
[24] N. Sheldon, E. Girard, S. Borg, M. Claypool, 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.
[25] J. Färber, Traffic modelling for fast action network games, Multimedia Tools and Applications 23 (1) (2004) 31–46, http://dx.doi.org/10.1023/B:MTAP.0000026840.45588.64.
[26] J. Font, D. Cascado, J. Sevillano, G. Lopez, S. Romero, G. Jimenez, Network requirements evaluation of a multi-user virtual environment, in: 2011 International Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS), 2011, pp. 90–97.
[27] O.W. Project, Wonderland Communications Architecture, <http://www.openwonderland.org/>.
[28] J. Lui, Constructing communication subgraphs and deriving an optimal synchronization interval for distributed virtual environment systems, IEEE Transactions on Knowledge and Data Engineering 13 (5) (2001) 778–792, http://dx.doi.org/10.1109/69.956100.
[29] O.W. Project, Open Wonderland Forum, <http://groups.google.com/group/openwonderland>.
[30] M. Hadžić, Supporting Distributed Software Teams with 3D Virtual Worlds. Master's Thesis. Institute for Information Systems and Computer Media (IICM), Graz University of Technology (Austria), 2010.
[31] O.W. Project, OWL Firewall Configuration, <http://openwonderland.org/>.
[32] jVoiceBridge Project, jVoiceBridge Developer Documentation, <http://www.openwonderland.org/>.
[33] J. Pfanzagl, Parametric Statistical Theory, Walter de Gruyter, Berlin, 1994.
[34] L. Le Cam, Maximum likelihoodan – introduction, ISI Review 58 (2) (1990) 153–171.
[35] J.M. Chambers, W.S. Cleveland, B. Kleiner, P.A. Tukey, Graphical Methods for Data Analysis, Chapman and Hall, New York, 1983.
[36] S.M. Ross, Stochastic Processes, Wiley, 1995.
[37] R.A. Fisher, Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population, Biometrika 10 (4) (1915) 507–521, http://dx.doi.org/10.1093/biomet/10.4.507.
[38] S.J. Devlin, Robust estimation and outlier detection with correlation coefficients, Biometrika 62 (3) (1975) 531–545, http://dx.doi.org/10.1093/biomet/62.3.531.
[39] C.D. Schunn, D. Wallach, Evaluating goodness-of-fit in comparison of models to data, Universitât des Saarlandes, Saarbrücken, 2005, pp. 115–154.
[40] M.P. Anderson, Applied Groundwater Modeling: Simulation of Flow and Advective Transport, 2nd ed., Academic Press, 1992.
[41] H. Schulzrinne, S. Casner, RTP Profile for Audio and Video Conferences with Minimal Control, RFC 3551 (Standard), July 2003, <http://www.ietf.org/rfc/rfc3551.txt>.