Level of detail based network adapted synchronization for cloud gaming

Richard Ewelle Ewelle, Yannick Francillette, Ghulam Mahdi, Abdelkader Gouaich, Nadia Hocine

▶ **To cite this version:**

Richard Ewelle Ewelle, Yannick Francillette, Ghulam Mahdi, Abdelkader Gouaich, Nadia Hocine. Level of detail based network adapted synchronization for cloud gaming. CGAMES 2013 - 18th International Conference on Computer Games, Jul 2013, Louisville, United States. pp.111-118, 10.1109/CGames.2013.6632616 . lirmm-02148503

HAL Id: lirmm-02148503

https://hal-lirmm.ccsd.cnrs.fr/lirmm-02148503

Submitted on 5 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Level Of Detail Based Network Adapted Synchronization for Cloud Gaming

Richard Ewelle Ewelle, Yannick Francillette, Ghulam Mahdi, Abdelkader Gouaïch
, Nadia Hocine and Julien Pons
LIRMM, University of Montpellier, France, CNRS
Email: {ewelleewel, francillet, mahdi, gouaich, hocine, jpons}@lirmm.fr

***Abstract*—Video games are considered as a major sector of popular entertainment and digital culture. With the arrival of cloud gaming, more video games become ubiquitous as they can be hosted on centralized servers and accessed through the Internet by thin clients with limited capabilities. Cloud computing within the game context has attracted significant attention due to its principal characteristics of scalability, availability and computational power. However, current cloud gaming systems have very strong requirements in terms of bandwidth and network resources. Thus, when devices have limited bandwidth and/or people are located in areas with limited network connectivity, they can not take advantage of these services.**

This paper presents an adaptation technique inspired by the level of detail (LoD) approach in 3D graphics. It is based on a cloud gaming paradigm for the minimization of the effect of poor network parameters (delay, loss, jitter) in order to enhance the game interactivity and improve the player quality of experience (QoE). A pilot experiment has been carried out to evaluate this approach through a game prototype. The results of this experiment show that the LoD based adaptation in cloud gaming provides a significant QoE enhancement on poor or congested networks.

***Index Terms*—cloud gaming, quality of experience, level of detail, client-server game**

I. Introduction

Video games are gaining increasing attention both from entertainment industry and from scientific community. Beginning with basic and toy games, they quickly evolved into rich interactive and animated environments approaching movies in terms of visual excitement.

A key aspect that will increase the use of video games in general, is to enable them to be accessible from every platform, and from everywhere in a way that maintains their realism and speed, the high quality graphics or any other aspect that can influence the players' quality of experience (QoE).

This objective has been partialy achieved by using *cloud computing* model for video games. Gaikai's [1] vision is that *"When video games can be accessed as easily as movies and music, we believe they will become the number one form of entertainment in the world"*.

Cloud gaming, also called "gaming on demand", is the type of online gaming that works on the same principle as video on demand. It allows direct and on-demand streaming of video games on a computing device using only a thin client. Here, the actual game is stored on the operator's server and directly streamed to the device. Given its advantages, cloud gaming can be considered as a new paradigm that may change the way computer games are delivered and played. That is probably why many companies such as Onlive [2], StreamMyGame [3] and Gaikai [1] have started offering cloud gaming services.

However, these cloud gaming systems have very strong requirements in terms of network resources. This restricts access to these services to small devices with limited network capabilities and to people living in areas where the network bandwidth is still limited. Besides, as the number of connected players increases, the server-side network bandwidth has to be increased, otherwise it will become the bottleneck of the overall system.

In this paper, we suggest taking advantage of cloud gaming platform to build video games that are accessible on devices with limited network resources, while maintaining the player's QoE.

The main contribution of this paper consists in an organization to adapt game entities' states synchronization in client-server games. We focus on the organization of game entities as the means to express different priorities depending on their importances in the game scene.

Indeed, whenever the generated network traffic for the game exceeds the capabilities of the actual network (bottleneck), our framework ensures that the entities with the high priority are given more network resources than the ones with low priority. This can therefore enable us to reduce the overall traffic load, meeting the limited network constraints.

This approach focuses on the interaction between video games and the cloud platform. Thus the general question guiding our study is:

How to build and run video games efficiently over the cloud while maintaining an acceptable player's QoE?

In cloud gaming, many factors can affect the player's quality of experience. Following the study in [4], the quantitative measurement of player experience mainly depends on the subjective factors: response time and graphics quality or received video quality. The game response time refers to the total delay from the occurrence of the player input to the display of the resulting graphic or video frame on the device. The received game video quality is influenced by the image quality in each frame and the smoothness of all the frames. The subjective factors are affected by a number of objective factors, which can be categorized into two groups: video settings (resolution, data rate, frame rate, codec) and network parameters (bandwidth, delay, packet loss and jitter). For this paper, we focus on

optimizing these objective factors in order to enhance the player experience and maintain it in an acceptable range when the network bandwidth decreases.

The rest of the paper is organized as follows: Section II discusses the background of this work. section III presents the related work which is followed by our proposed adaptation approach in section IV. Section V describes the experimental framework and the game used to validate our approach; we then analyze the results of our pilot experiment with player evaluation in terms of QoE and, we conclude this paper in section VI by presenting conclusion and future directions.

II. Backgrounds

A. Cloud computing

According to Geoffrey et al in [5], a cloud is a hardware and software infrastructure which is able to provide services at any traditional IT layer: software, platform, and infrastructure. The claimed advantages are minimal deployment time and reduced costs thanks to an on-demand and pay-per-use model. Here is a little exploration of the layers in the cloud computing stack from the bottom up, as presented in [5]:

1) *Infrastructure as a service (IaaS)*: Provides the distributed multi-site physical components to support cloud computing.
2) *Platform as a service (PaaS)*: Provides computational resources via applications and services that can be developed and hosted.
3) *Software as a service (SaaS)*: Provides applications /services using a cloud platform, rather than providing cloud features themselves.

Cloud Computing offers to application providers the possibility to deploy a product as a SaaS and to scale it on demand, without building or provisioning a data center. With respect to this definition, cloud gaming can be viewed as offering games as applications through the cloud to gamers.

B. Cloud gaming

In the simplest form, cloud gaming is just executing games on a server located in the cloud instead of users' devices. When launched the client contacts the server to establish the connection and starts the game. As explained in [6], usually there are two connections: one connection to send the player's controller input to the server and one connection to receive the video or commands from the server.

Figure 1 explains this communication flow. The client runs two processes: the main (video) process, and the input process. The video process receives packets from the server; these packets are used to decode the video stream. When a complete frame has been decoded, it is displayed in the application's window. The input process polls for input events from the player. When an event is received it is written into a packet and sent to the server. A typical server also runs two processes: a video streaming process that captures frames from the game, encodes them into a video, and then sends the video output to the client; and an input process that receives inputs from the client application and passes those inputs to the game.

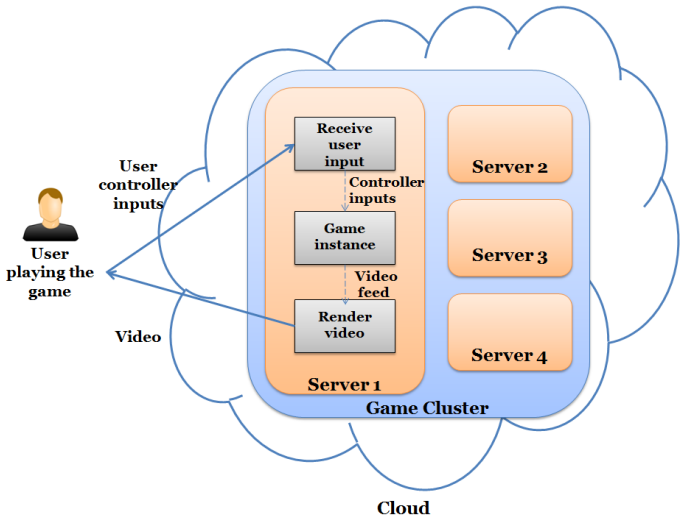

Fig. 1. Cloud gaming architecture

For an interactive experience, such a system has some requirements such as low end-to-end delay, high compression efficiency, and low encoding complexity. Wang et al. in [4], have divided these requirements in two groups: the video parameters and the network parameters.

For instance, the perceived loss of quality in the image can be caused by an inefficient video compression process from the source or by a poor network configuration such as a congested network.

The quality of the video and the interactivity of the game is therefore influenced by network parameters e.g. packet loss rate, delay and jitter. With video streaming, the game response time typically consists of the codec delay, caused by video encoding and decoding, the network round-trip delay and the client's play out buffering delay. To guide our state of the art, we are going to review studies aiming to optimize these network parameters.

C. Client-server game loop

A game loop is presented in the figure 2. The game starts with an initialization stage where all the objects and entities of the game model are initialized. After this phase, the game loops between the update and rendering phase. In the update phase, all the entities in the game model are updated and in the rendering phase, the game produces the graphic output.

In a client-server setup with the cloud gaming paradigms, the phases of this game loop are distributed among the client and the server. Two approaches are generally found in the literature:

- The client and the server have a representation of the game model. The client side periodically synchronizes its local game model with the server side's remote game model which is the central one. This is done by receiving update messages about state change in the game model.
- The most used solution in industrial cloud gaming systems is based on video streaming [2], [7], [3]. Here, the

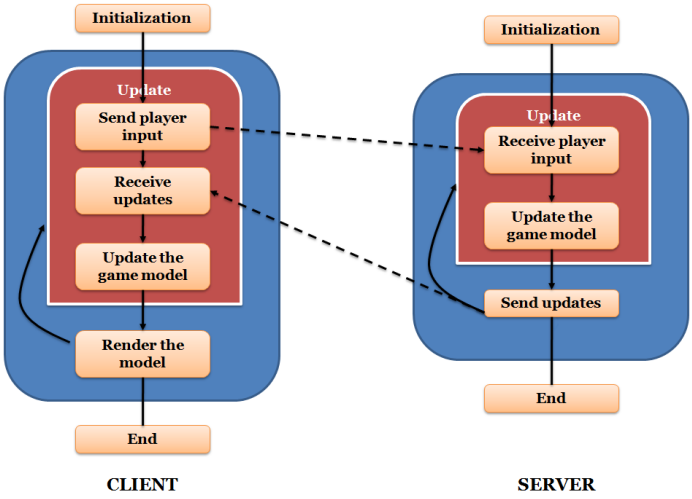

Fig. 2. Client-Server Game loop

server executes the game logic and the graphical output is produced and transmitted to the client as a video stream.

III. Related Work

In the cloud gaming paradigm, to improve the interactivity performance of a game architecture, two main causes for delays have to be analyzed: network latencies and video processing costs. Several research works have already brought contributions to the optimization of the video processing mechanism, but in our study we are focusing on network optimization.

For the network settings optimization context, A. Jurgelionis et al. in Games@Large [8], propose a distributed gaming platform for cross-platform video game delivery. The system executes games on a server PC, located at a central site or at home, captures the graphic commands, streams them to the end device, and renders the commands on the end device allowing the full game experience. For end devices that do not offer hardware accelerated graphics rendering, the game output is locally rendered at the server and streamed as video to the client. The advantage of this approach is that the architecture is transparent to legacy code and allows all type of games to be streamed. For devices with rendering capabilities, it discharges the server from rendering and encoding the game output, enhancing the server performance, it also reduces the amount of data to be transmitted since only graphics commands are sent to the client, thus reducing the game latency. For users with low end devices, the latency of video encoding and video transmission remain the same as in other cloud gaming architecture like Onlive. Sending only graphic commands reduces the traffic load in the network, but the generated traffic is still important and may not be supported in some network configurations.

In the attempt to reduce the network load in state synchronization for client-server games in general, several works have contributed to the development of efficient synchronization schemes. In particular, packet compression [9] tries to speed up transmissions by reducing bandwidth requirements. Indeed, minimizing the number of bits needed to represent a game information is a proficient method to diminish the traffic present in the network. Aggregation [10] is another technique attempting to reduce the overhead associated with each transmission therefore limiting the bandwidth required by the application. Specifically, before being transmitted, packets are merged in larger ones thus reducing the overhead. Both schemes, however, pay the latency benefits achieved with an increment in computational costs. Information compressed and aggregated, needs to be recovered with decompressing and disaggregating algorithms at the receiving end, thus increasing the time required to process each single event. Moreover, aggregation can origin further waste of time if a transmission is delayed while waiting for having available other events to aggregate.The packet compression technique is also often used in video game streaming.

To reduce both the traffic load in the network and the computational cost to process each game event, Interest Management techniques have been used [11]. In some multi-player game scenario, events generated are relevant only for a small fraction of the players. Therefore, implementing an area-of-interest scheme for filtering events, as well as a multi-cast protocol, could be put in good use to match every packet with the nodes that really need to receive it and, consequently, to reduce both the traffic and processing burden at each node [12]. Games having interest areas occupying a significant portion of the global virtual environment could hence be further delayed if Interest Management schemes would be implemented.

In order to be less dependent from the real responsiveness of the network, optimistic algorithms for synchronizing game state at servers can be used in order to avoid delay perception at destination [13], [14]. In case of lousy interactivity between client and server, in fact, an optimistic approach executes events before really knowing if ordering would require to process other on the way events first. Game instances are thus processed without wasting any time in waiting for other eventually coming packets. On the other hand, this performance gain is paid with some occurrence of temporary consistency loss. In our work, we are dealing with synchronous real-time games, therefore maintaining the consistency of the game is a must.

Dead reckoning is another method that can help to minimize the effects of latency, but it can also introduce some temporary incoherence between the factual game state and the assumed one at the server [15]. In fact, attempting to limit the bandwidth required by the application, this scheme utilizes a reduced frequency in sending update packets while compensating the lack of information with prediction techniques. Obviously, predicted movements and actions are not always trustful. These eventual restoring actions further impacts on interactivity and playability of the game.

All these mechanisms propose enhancements that can improve the performance of a real-time networked game by reducing the traffic load between the client and the server.

Nonetheless, this traffic reduction introduces some computation expenses contributing to the lag or other incoherence in the game. Techniques such as interest management and dead reckoning are widely used to reduce both the network load and the computational cost for a better player quality of experience. None of the analyzed work have tried to couple the traffic reduction with a scheduling mechanism for an efficient message passing (messages with different importances) between the client and the server, nor have tried to adapt the traffic generated by the game to the actual capacity of the network.

We present here a novel synchronization adaptation technique, specifically designed for an efficient event delivery synchronization in client-server games with the cloud paradigm.

IV. Adaptation Framework

In this section, we introduce the LoD principles and present our proposed approach for client-server synchronisation.

A. Level of detail

The graphical Level of detail (LoD) technique has been introduced by James Clark [17] to manage the processing load on graphics pipeline while delivering an acceptable quality of images.

The technique is based on the idea that structuring the rendering details of a 3D object can optimize the processing quality, if the object's visible details are proportional to the distance from which the object is viewed.

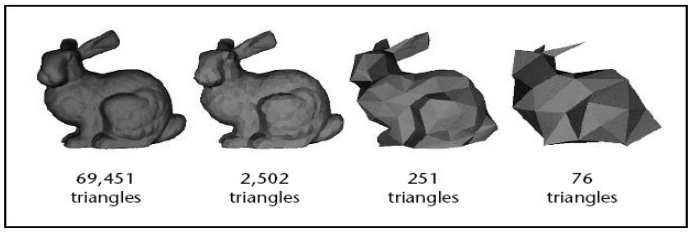

Fig. 3. Basic concept of LoD: An object is simplified by different representations by varying number of polygons [16]

The figure 3 presents the rational of Clark: by changing the number of vertices, we see the change in the quality and the visualization of a sphere.

Geometric datasets are usually too large in data size and complex (in terms of time and computational resource demands) so their rendering can become a tedious and time consuming process. The LoD approach suggests different representations of a 3D object model by varying in the details and geometrical complexity. The geometrical complexity of an object is determined by the number of polygons used for his representation. The more complex an object is, the more time consuming its rendering will be.

Although there can be other factors involved in the complexity and resource demand of a graphical model of an object, the relations between polygonal quantity and resource consumption are generally considered as established ones [18]. For example, one can determine the difference in rendering quality by observing the figure 3. This figure also makes it possible to draw a general conclusion on how the number of polygons affects the rendering quality of an image's graphical representational.

Once these different representations of a model are on hand (as shown in the figure 3), the LoD technique will suggest their selection at a particular time point based on certain positive selection bias. The latter can be their size, camera distance or any other criteria.

The application of this technique for our work is the ability to have different synchronization needs for each entity, and select one at a particular time based on certain criteria. This way, only certain entities will get the maximum amount of communication resources while others get less. These entities can see their communication resources changed when the network situation changes or when their importance in the game changes.

B. Overview

In classical networked game architecture with the cloud gaming paradigm, all the game entities of a scene are updated in a synchronous basis. We aim at optimizing these state synchronizations between the client and the server. We make the assumption that there are different synchronization needs per entities. Some need a small update rate and for others larger update rate will be enough. For example, background entities need less synchronization than target entities in a shooting game. We therefore need an efficient message passing protocol for better performances. That is why, we apply a LoD inspired mechanism to prioritize some updates over others. This enables us to lower the game's communication requirements.

The approach maintains a bidirectional multi-level QoS for game entities at runtime. Meaning that the adaptation needs to lessen the games communication requirements when it notices a bottleneck in the network (materialized by an increase in the response time, or packet loss in case of UDP packets) and maintain an optimal communication rate otherwise.

The proposed framework uses entity organizations. Here entities belong to a a synchronization group (with a specific synchronization or communication rate) and have specific roles within that group. The group association is done according to the significance of the entities in the organization. The significance of an entity can be defined in many different ways depending of the game designer settings. For this paper, we used the functional importance of an entity in the game as significance.

C. Organizational model of entities

We use organizations for several reasons:

- Organizing and representing different communication levels of entities, so that a change in the network load steers the use of a particular communication level for an entity.
- Evaluating the relative importance of the entity in real time in other to select the most appropriate communication level.

Our organization model is inspired by the AGR model [19]. Here we briefly define the AGR model. The AGR (Agent, Group, Role) model advocates organization as a primary concept for building large scale and heterogeneous systems. The model does not focus on the internal architecture nor the behavior of individual agents but suggests organization as a structural relationship between collection of agents. The AGR model defines three main concepts as its basis for an organizational structure : agent, as an active and communicating entity ; groups are comprised of agents in the set by tagging them under a collection ; finally an agents functional representation in a group is given by defining its role.

Our proposition matches the AGR model with the same basic components:

- **Agent:** An agent representing a game entity in the game scene.
- **Role:** A role representing the reason why the game entity is in the game. Each entity in the scene has a role and a role can be shared by several agents.
- **Group:** A group is a set of entities that are synchronized at the same frequency. In our approach, several group with different update frequencies are defined. Each group also has a score coefficient (see score coefficient below) threshold, that the entities should satisfy to be associated to the group.

An entity can move from one group to another at any moment according to the observed network settings and his significance in the game. The game engine ensures the message passing using a module called network updater.

Each update transmission uses some communication resources for its completion. We describe a score coefficient as an abstract measurement unit for the notion of communication resources for weighting entities communication requirements. Each entity has a significance regarding its communication requirements.

An entity's score coefficient is calculated at runtime using a combination of the actual network setting (here for UDP the packet loss percentage) and the entity's significance. This way the entity's importance is proportional to the network load at running time. At any time. This score coefficient of an entity is computed using the following formula:

$$ScoreCoefficient = Significance * CurrentNetworkLoad$$

This generic notion of significance is a value that defines whether the entity is important at the current time or not. This score allows us to sort which are the most important entities at the current time. As the significance of an entity is depending to the game rules and the function of the entity, this notion can be exploited in many ways. For the game prototype developed in this paper, the significance is a weight defined on each roles by the game designer through a configuration file.

In case of congestion, entities are reassigned to groups. As general rule, an entity goes in a group only if its score coefficient is lower than the group's score coefficient's threshold.

D. Implementation

1) Game Engine: The game engine maintains a collection of communication groups. Here, when the game notices a drastic change in network load, the score coefficient of each entity is recalculated and entities are assigned the communication group matching the required score coefficient's threshold.

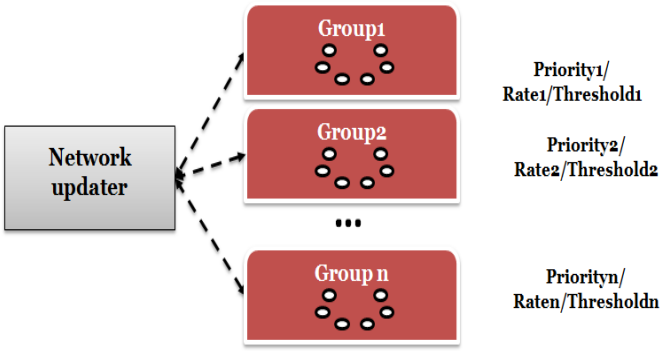

Fig. 4. Adaptation framework

2) Network load computing: The current network load is the main parameter for the calculation of the score coefficient, and represents the traffic load generated by the game at running time. Because our message passing is done using the UDP protocol, as a representative of network traffic load, we monitor the packet loss in the network since in case of congestion some packets will be lost. Knowing that with UDP, there is no guaranty on the reception of packets.

To monitor the packet loss at runtime, the server periodically sends a number of monitoring packets to the client and simply counts the number of responses it receives back. Each missing response is marked as packet loss. Thus in case of network congestion, the amount of packet loss will increase, and therefore our adaptation scheme will recalculate the score of the entities changing the communication profile of the game.

E. How it works

To illustrate our approach, here is a simplified version of a shooting game, "My Duck Hunt", developed to evaluate the approach. You will find a more complete description in the subsection V-A. Suppose that we have 3 types of entities: clouds, ducks and a reticle. Lets say the objective is to point the reticle on the ducks and shoot. These entities have different functional importances in the game, therefore different significances as defined in the subsection IV-C. Here the significance is a weight value associated with each entity.

- Cloud: represents clouds that are moving in the background. Less importance; weight: 1.5.
- Duck: represents ducks that are flying in the game scene. Ducks are the targets. Medium importance; weight: 1.
- Reticle: represents the player's weapon's pointer in the screen. More important, weight: 0.5.

Suppose that we have 3 synchronization groups with different communication rates and score thresholds:

- Degraded: Less important entities; rate: every 100ms.

- Medium: Semi important entities; rate: every 50ms.
- Optimal: More important entities, rate: every 10ms.

When the game starts and when any drastic change is noticed in the network load, entities are redistributed to the groups.

- Without LoD: All the entities will always be attributed to the optimal group. Their updates will be sent every 10ms and then will evenly compete for network resources, therefore will be impacted the same way by any network congestion.
- With LoD: if there is no congestion on the network, all the entities will be on the optimal group. In case of congestion, the entities are redistributed to the groups using the formula in IV-C to compute their current score coefficient. So if the groups thresholds are well set, the clouds will belong to the degraded group with updates sent every 100ms, the ducks will belong to the medium group with updates sent every 50ms, and the reticle will belong to the optimal group with updates sent every 50ms, reducing the overall traffic load.

V. Pilot Experiment

The objective of this experiment is to evaluate the impact of the LoD based synchronisation on the player's experience. For that, we observe and compare the reaction of players during a game session with and without the proposed approach.

A. My Duck Hunt video game

The video game "*My Duck Hunt*" has been developed to conduct this experiment. This is a competitive shooting game inspired from the traditional Duck Hunt video game [20]. The rules of the game are the following : the player sees a scene where five kinds of entities evolve: the **reticles**, the **ducks**, the **flamingos**, the **gombas** and the **clouds**. The player controls the reticle and has to achieve the following goals :

- The game is divided in 5 rounds or waves of ducks.
- The player should kill as many ducks as he can. For each duck killed, the player gains points.
- The player must not kill flamingos.
- The player should protect flamingos from gombas by killing gombas. Each flamingo killed result in point loss.

The clouds are in the scene as decoration elements. The figure 5 shows a screen shot of the game. In this screen shot, the ducks are the entities with the black body and the green head. The flamingos are pink and the gombas are the brown entities on the floor.

B. Protocol

The study follows a repeated-measures design. The candidates have to play two versions of My Duck hunt. One that includes the our LoD inspired proposition and the other without the proposition. In the later game, all the entities are updated at the same rate.

The experiment proceeds as follows :

Fig. 5. Screen shot of "My Duck Hunt" video game

1) The candidates get a quick introduction about the rules of the game through a desktop demo version of the game.
2) The candidates play one version then the other (the order is random for all the players). The subjects are not informed about the difference between the two versions. During the game, each subject has to report when she/he perceives a "lag" or interactivity shortage by holding the key space.
3) At the end of each wave, each candidate evaluates the quality of experience during this wave. He gives a note between 1 and 5. 1 indicates a bad game experience. 5 means an good game experience.
4) At the end of the experiment, the candidates are individually interviewed.

C. Network configuration

In order to be able to control the network environment, the experiment is performed on a Local Area Network. In this LAN we have the game server machine, the client and a proxy for delay and packet loss simulation. The proxy forwards all the packets from the client to the server and vice-versa. Since we are using UDP connections to send the updates, the proxy simulates a congested network by ignoring all the packets received while a certain threshold of packets sent per second is reached. This threshold represents the capacity of the network or the available bandwidth: number of packets to forward per second. So to drastically change the capacity of the network for a game, we just need to change this threshold value.

The game server is a *Dell Precision M6500* with the following configuration : a *Intel Core i7 Q 720* CPU and 4 Go of RAM. The server is configured with the four following groups ranged by other of importance:

1) **Optimal group**: Entities with the highest communication requirements. In this group the update frequency of entities is 5 ms and the threshold score to stay in this group is 7.

2) **Enhanced group**: Entities with relatively high communication requirements but lesser than those in the optimal group. Update frequency = 35 ms, threshold = 15.
3) **Medium group** Entities with average needs in network resources. Update frequency = 40 ms, threshold = 70.
4) **Degraded group** Entities with the lowest communication needs. Update frequency = 75 ms.

The proxy is started at the same time as the game and it is launched with a configuration file dictating the network capacity variations during the game. This first 210 seconds of this configuration is given in the table I. From 0 to 30 secondes, the proxy forwards 6000 packets per second; from 30s to 60s, it forwards 3000 packets per second, marquing a 50% network quality degradation, etc...

Time	0s	30s	60s	90s	120s	150s	180s	210s
Pkts/s	6000	3000	5000	2900	7000	2500	3500	3100

TABLE I
PROXY CONFIGURATION FOR NETWORK CAPACITY

D. Participants

The pilot test was conducted on 9 subjects. These participants are between 21 and 30 years old with a mean of 25.33 The distribution of players based on their playing frequencies, it is given in the table II. Only one participant has reported that he does not play video games. The other participants play games at least once a week.

Never	**1 per year**	**1 per month**	**1 per week**	**everyday**
1	1	2	5	0

TABLE II
PLAYING FREQUENCIES DISTRIBUTION

The distribution of players based on their games genre preferences is given by the table III. Real time video games are video games where all of the players can send their inputs at the same time. For example, First Person Shooter, Fighting and Racing games. Turn based games are games where the players have to wait their turn to send their inputs. According to the table, most of the participants prefer real time video games.

Real time games	**Turn based games**
6	3

TABLE III
PREFERRED GENRE DISTRIBUTION

E. Hypothesis

In order to evaluate the impact of the proposition we have stated the following hypotheses :

- **H.A.0** There is no difference in the game experience between the two versions of the game for each wave.

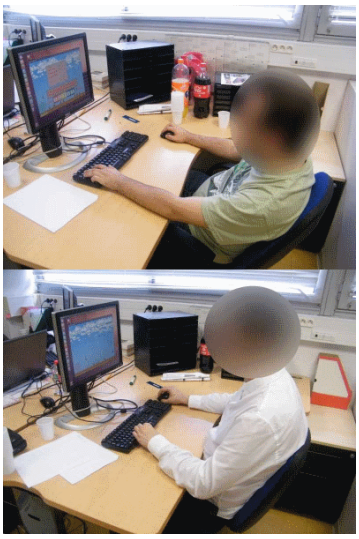

Fig. 6. Pictures of the experiment

- **H.B.0** There is no difference concerning the global game experience for the entire game session between the two versions of the game.
- **H.C.0** There is no difference in the ratio of, the amount of time the space key is pressed and the game session duration, between the two versions of the game.

F. Result

We use the paired t-test to reject the three hypotheses. The statistical analysis was performed using R http://www.r-project.org version 2.15.0.

The hypothesis **H.A.0** is rejected for the five waves with $p-value < 0.5$. The difference between the game experience when using Lod and without Lod is statically significant. The results of the t-test are summarized in the table IV.

Wave number	**M**	**t(8)**	**p-value**
Wave 1	1.2222	4.4	0.002287
Wave 2	1.555556	6.4236	0.0002039
Wave 2	1.666667	3.7796	0.005391
Wave 4	1.222222	3.0509	0.0158
Wave 5	1.666667	3.0151	0.01668

TABLE IV
RESULTS

The hypothesis **H.B.0** is rejected by the t-test. The difference of the global game experience for the entire game session between the game with Lod based adaptation and the game without Lod is statically significant with a mean M = 1.777778, t(8)=8.6298, $p-value = 2.521e-05$

The hypothesis **H.C.0** is rejected by the test. The different between the ratio of, the amount of time the space key is pressed and the game session duration between the two game versions is statically significant with a mean M = -0.1197444, t(8) = -2.4535 and $p-value = 0.03972$.

G. Discussion

The data gathered during the experimentation and the statistical study show the effect of our approach on the players' interactivity with the game, therefore their QoE. In fact all the three hypothesis were rejected meaning that the players have perceived a significant gain of the game experience with the adaptation technique not only for each wave but also for the overall game session.

The t-test results have also rejected the hypothesis H.C.0. but with a higher $p - value$ than the others. Because we choose a $p - value$ threshold of 0.5, the hypothesis is still rejected but we can see that the results are not strong as in the two first hypothesis. A reason for this is that during the experiment, not all participants were following the guideline we gave them about pushing the space key every time they notice a loss of interactivity or a decrease in the playability of the game, instead they were complaining verbally. Thus for some participants the data we gathered from the space key, does not reflect the quality of experience they actually experienced.

Finally, these results validate our approach on improving the overall quality of interactivity of the game, showing that adapting the network load generated by the game to the actual network capacity significantly increase the perceived quality of the game.

Our approach is not perfect, here are some limitations:

- Since the setting the configuration parameters (entity's weight, score coefficient threshold) of the platform is done manually by the game designer, the whole adaptation can be very subjective. A bad configuration of this system, can result to a very bad player's QoE. In our work, we suppose that the game designer knows what is doing(the significance of each entity) and the parameters are well set.
- It is also important to note that, while this approach reduces the bandwidth needed for a good quality game, it does not eliminate the requirement of a minimum bandwidth for an enjoyable game. It for sure requires less bandwidth than the classic cloud gaming services.

VI. Conclusion and future works

In this paper, we proposed a level of detail inspired synchronization scheme used in cloud gaming. The challenges in this context are due to the variability and sometimes the shortage of network connectivity required to enjoy a classic cloud gaming service using video streaming. For that we propose a new adaptation technique using entities organization to: (1) minimize the effect of these network settings for maintaining game interactivity and an improved player's quality of experience (2) get benefit of the elasticity, the availability and scalability of the cloud in video games.

The next objective is to implement our version of the adaptation scheme using a topology heuristic (distance to camera) as the significance of the entities coupled with a multiplayer environment, where each player has a different camera and sees different angles of the game scene. We will then to carry out a large scale experiment in a multiplayer cloud gaming environment.

References

[1] Gaikai, "Gaikai chooses new nvidia geforce grid to fuel explosive growth in cloud gaming," http://www.nvidia.com/content/PDF/NVIDIA-GeForce-Grid-Gaikai-Case-Study-HR.pdf, 2012, [Online; accessed 31-Octobre-2012].

[2] Onlive, "Onlive official web page," http://www.onlive.com, 2012, [Online; accessed 31-Octobre-2012].

[3] StreamMyGame, "Streammygame official web page," http://www.streammygame.com, 2012, [Online; accessed 31-Octobre-2012].

[4] S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *Proceedings of the 28th IEEE conference on Global telecommunications*, ser. GLOBECOM'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 4231–4237. [Online]. Available: http://dl.acm.org/citation.cfm?id=1811982.1812085

[5] T. M. C. Geoffrey Raines, "Cloud computing and soa," 2009, [Online; accessed 31-Octobre-2012]. [Online]. Available: http://www.mitre.org/work/tech_papers/tech_papers_09/09_0743/09_0743.pdf

[6] D. R. D. Barievi, , and M. Chandrashekar, "Gameon: Analysis and implementation of cloud gaming," 2011, [Online; accessed 31-Octobre-2012]. [Online]. Available: http://www.cs.ucsb.edu/~manasa/cs276.pdf

[7] G-Cluster, "G-cluster official web page," http://www.gcluster.com, 2012, [Online; accessed 31-Octobre-2012].

[8] A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J. P. Laulajainen, R. Carmichael, V. Poulopoulos, A. Laikari, P. Perälä, A. De Gloria, and C. Bouras, "Platform for distributed 3d gaming," *Int. J. Comput. Games Technol.*, vol. 2009, pp. 1:1–1:15, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/231863

[9] B. S. An, M. Lee, K. H. Yum, and E. J. Kim, "Efficient data packet compression for cache coherent multiprocessor systems," in *Proceedings of the 2012 Data Compression Conference*, ser. DCC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 129–138. [Online]. Available: http://dx.doi.org/10.1109/DCC.2012.21

[10] T. Ferrari, "End-to-end performance analysis with traffic aggregation," 2000.

[11] K. L. Morse, L. Bic, and M. Dillencourt, "Interest management in large-scale virtual environments," *Presence: Teleoper. Virtual Environ.*, vol. 9, no. 1, pp. 52–68, Feb. 2000. [Online]. Available: http://dx.doi.org/10.1162/105474600566619

[12] S. E. Deering, "Host extensions for ip multicasting," United States, 1989.

[13] D. D. J. S. Steinman, J. W. Wallace and D. Elizandro, "Scalable distributed military simulations using the speedes object-oriented simulation framework," in *Proceedings of Object-Oriented Simulation Conference (OOS'98)*, ser. OOS'98, 1998, pp. 3–23.

[14] L. Gautier, C. Diot, and J. Kurose, "End-to-end transmission control mechanisms for multiparty interactive applications on the internet," in *In IEEE Infocom*, 1999, pp. 1470–1479.

[15] S. K. Singhal, "Effective remote modeling in large-scale distributed simulation and visualization environments," Stanford, CA, USA, Tech. Rep., 1996.

[16] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney, *Level of Detail for 3D Graphics*. New York, NY, USA: Elsevier Science Inc., 2002.

[17] J. Clark, "Hierarchical geometric models for visible surface algorithms," *Communications of the ACM*, vol. 19, no. 10, pp. 547–554, 1976.

[18] M. F. Deering, "Data complexity for virtual reality: Where do all the triangles go?" in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, 1993, pp. 357–363.

[19] J. Ferber, F. Michel, and J. Baez, "Agre: integrating environments with organizations," in *Proceedings of the First international conference on Environments for Multi-Agent Systems*, ser. E4MAS'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 48–56. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-32259-7_2

[20] Wikipedia, "Duck hunt," http://en.wikipedia.org/wiki/Duck_Hunt, [Online; accessed 12-October-2012].

View publication stats