Network Aware Traffic Adaptation for Cloud Games

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

HAL Id: lirmm-02148499
https://hal-lirmm.ccsd.cnrs.fr/lirmm-02148499

Submitted on 8 Sep 2022

Network Aware Traffic Adaptation For Cloud Games

Richard Ewelle Ewelle
and Yannick Francillette
LIRMM, University of Montpellier
France, CNRS

Abdelkader Gouaïch
and Ghulam Mahdi
LIRMM, University of Montpellier
France, CNRS

Nadia Hocine
and Julien Pons
LIRMM, University of Montpellier
France, CNRS

Abstract—**Current cloud gaming systems have very strong requirements in terms of network resources. For pervasive gaming in various environments like at home, hotels, internet cafes, or area with limited or unstable network access, it is beneficial to avoid the necessity of having a costly steady fast speed internet connection to be able to run these cloud games.**

We present a network aware adaptation technique based on the level of detail (LoD) approach in 3D graphics. It maintains a bidirectional multi-level quality of service for game entities at runtime, lessening the game's communication cost when it notices a decrease in network resources and maintains an optimal communication frequency otherwise. It therefore reduces the impact of poor and unstable network parameters (delay, packet loss, jitter) on game interactivity while improving user's quality of experience (QoE). The evaluation of our approach through a pilot experiment shows that the proposed technique provides a significant QoE enhancement.

Keywords—*Level of detail, quality of experience, cloud gaming, network, communication, accessibility, ubiquity.*

I. Introduction

Cloud computing describes a broad movement toward the use of wide area networks, such as Internet to enable interaction between IT service providers of many types and consumers. According to [1], a cloud is a hardware and software infrastructure which is able to provide services at any traditional IT layer: software, platform, and infrastructure, with minimal deployment time and reduced costs. With the tremendous popularity of video games, moving games in the cloud will immensely increase their accessibility and ubiquity, by enabling them to reach geographically separated storage or data resource with even cross-continental-networks.

Cloud gaming, also called gaming on demand, is the type of online gaming that works on the same principle as does video on demand which is another data-intensive application. It allows direct and on-demand streaming of video games on a computing device through the use of a thin client. Here, the actual game is stored on the operator's cloud platform and it is directly streamed to the device. Due to these potential advantages many companies like Onlive [2], StreamMyGame[3], Gaikai [4] offer cloud gaming services.

As promising as it is, cloud gaming is also facing many challenges that, if not well resolved, may impede its fast growth. These data-intensive cloud services have very strong requirements in terms of network resources. In fact in order to ensure the delivery of quality of service (QoS) for the bulk data transfer generated by cloud games, a certain amount of network resources should be available. These constraints reduce the accessibility and ubiquity of such services, because devices with little or unstable bandwidth capabilities and people located in area with limited or unstable network access, cannot take advantage of these cloud services. In addition, the number of connected users increases every day, creating more processing demand, and network congestion, thus limiting the use of cloud services.

In the context of cloud gaming, to preserve game accessibility and ubiquity, we need not only to weaken these network constraints, but also maintain the quality of the game. The objective of the paper is then to make it possible for devices with poor or unstable network capabilities, to run cloud games with an acceptable QoE. The paper achieves this goal with the proposition of an organization based network aware traffic adaptation for cloud games achieving accessibility and player QoE maintenance. We focus in our approach on the organization of game entities as a means to express different priorities among entities depending on their importance in the game. Whenever the required network resources for game state synchronisation is not meet by the available network configuration, our framework ensures that the entities with the high priority are given more network resources than the ones with low priority, thus enables to meet the limited network constraints.

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

How to build and run video games efficiently over the cloud by optimizing the interaction between the client and the server for an increased player experience and game accessibility?

User experience in games and more generally in interactive entertainment systems has been a real focus in the research community. In cloud gaming, the game response time influences the user quality of experience as explained in [5]. The game response time, is the total delay from the user control command occurring, to the corresponding graphic or video frame displaying on the device screen. It is influenced by network parameters like (bandwidth, delay, packet loss and jitter). This paper focuses on optimizing the game system in other to minimize the effect of poor network parameters for a better game interactivity and an increased user experience.

The remaining of this paper is organized as follows. We first give the definition of the cloud gaming and its two main types in Section II. Then, we propose a new network aware multi-level traffic adaptation scheme with efficient game state synchronisation in Section III. In Section IV, we describe the experimental setup and the game used to validate our approach;

we then analyze the results of our pilot experiment with player evaluation in terms of QoE. Section V gives the related work on network resources optimisation in cloud gaming and networked games in general and finally, we conclude this paper in section VI by presenting conclusion and some future directions for this work.

II. Cloud Gaming

Cloud gaming can be defined as just executing games in a server instead of users' devices. In a client-server setup with the cloud gaming paradigms, two main trends exist: Video streaming and game state synchronisation.

A. Video Streaming

The most used solution in industrials cloud gaming systems are based on video streaming [2], [6], [3], [7]. In this configuration, only the server maintains a representation of the game model; the client simply gathers inputs from the user and sends them to the server; and receives video from the server and displays it to the user's screen. When launched the client contacts the server to establish the connection and start the game. As explained in [8], usually there are two connections: one connection to send the user's controller input to the server and one connection to receive the video from the server. The figure 1 shows this game loop.
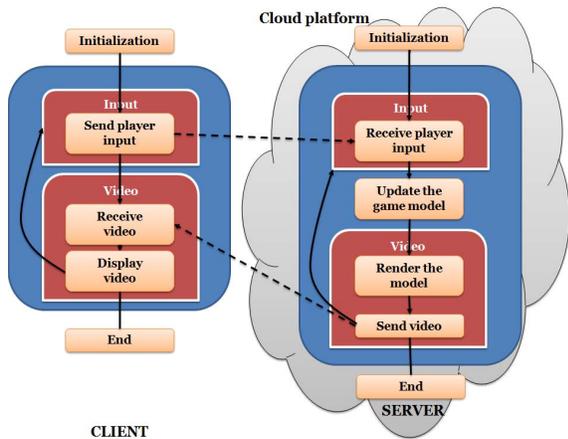

Fig. 1.   Cloud game loop: Video streaming

The client game loop run two tasks: the video task and the input task. With the input task, the client polls for input events from the player. When an event is received it is written into a packet and sent to the server. The video task enables the client to receive 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.

A typical server game loop also runs two tasks. An input task that receives inputs from the client application and passes those inputs to the game model, and a video streaming task that captures frames from the game, encodes them into a video, and then sends the video output to the client.

B. Game State Synchronisation

The client and the server have a representation of the game model. The client side periodically synchronizes his local game model with the server side remote game model which is the central one. This is done by receiving update messages about state change in the game model. This game update loop is presented in the figure 2. The game usually 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 input phase and the update phase. The major difference is that, with this configuration, the video rendering is done on the client side, and the server just sends state updates in smaller packets.
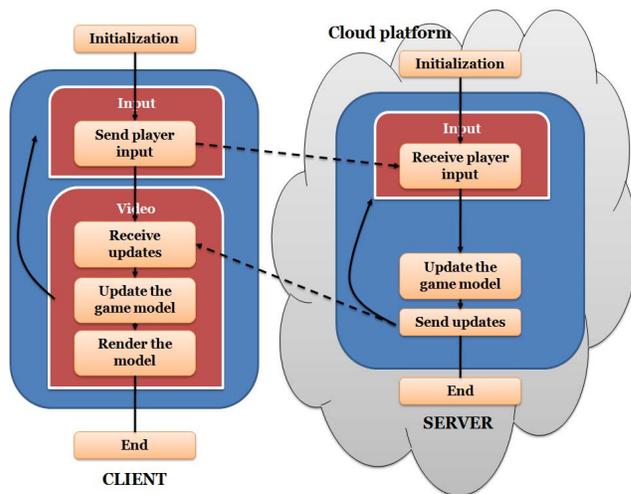

Fig. 2.   Cloud game loop: Cloud Game loop: State synchronisation

In this paper, we will consider the game state synchronisation approach as an alternative to video streaming in cloud gaming because it does not have strong requirements on network resources.

III. Our Traffic Adaptation Approach

A. Main Idea

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 the state synchronization between the client and the server. We make the assumption that there are different synchronization needs per game entity. Some need a small update frequency and for others a larger update frequency 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 that takes this synchronisation needs in consideration for better performances. That is why; we apply a LoD inspired mechanism to prioritize some updates over others.

B. Level of Detail

The LoD technique has been widely used in 3D graphics and simulations. The basic purpose of the technique is to modulate the complexity of a 3D object representation according to

the distance from which it is viewed (or any other criteria) [9]. As introduced by James Clark [10], this technique is meant to manage processing load on graphics pipeline while delivering an acceptable quality of images. The technique suggests to structure the rendering details of a 3D object in order to optimize its processing quality if the object's visible details are proportional to the distance from which the object is viewed.
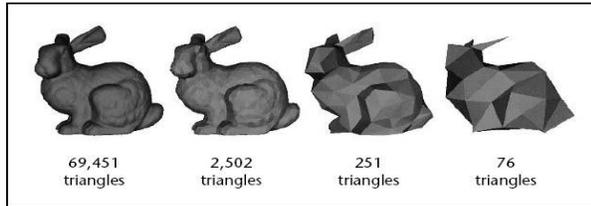

Fig. 3. Basic concept of LoD: An object is simplified by different representations through varying number of polygons [9]

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.

The figure 3 shows how the number of polygons affects the rendering quality of an image's graphical representational. With these different representations of a model on hand (as shown in the figure3), 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 frequencies for each game entity, and select one at a particular time based on certain criteria. This way, only more important entities will get the maximum amount of network 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.

C. Overview of Our approach

The approach maintains a bidirectional multi-level quality of service for game entities at runtime. Meaning that the adaptation needs to lessen the game's 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 frequency otherwise. This is made possible using organizations.

Here each entity belongs to a synchronization group (with a specific communication frequency) and has a specific role within that group. The group assignment for an entity is done according to its significance 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 use the functional importance of an entity in the game as significance.

D. Organization of Entities

Our organization model is inspired by the AGR model [11]. The AGR (Agent, Group, Role) model advocates organization as a primary concept for building large scale and heterogeneous systems. The AGR model does not focus on the internal architecture nor the behaviour of individual agents but it suggests organization as a structural relationship between the collection of agents. The AGR model defines three main concepts as its basis for an organizational structure: agent, as an active and communicating entity; a group is a set of agents which is defined by tagging them under a collection; finally a role defines an agent's functional representation in a group. The organizational model we use in our approach matches the AGR model as follows:

- **Agent:** An agent represents a game entity involved in the game scene. Each entity has a significance regarding its communication requirements.

- **Role:** The role represents the reason why the entity is in the game. Each entity in the scene has a role and, a role can be shared by several entities. The role can also be used to express entity's communication significance.

- **Group:** A group is a set of entities with the same communication needs. These entities are synchronized at the same frequency. An entity can move from one group to another at any moment according to the observed network settings and his significance in the game.

E. Communication Groups

A communication group is characterized by a communication frequency and a score coefficient (see score coefficient below) threshold, that the entities should satisfy to be assigned to the group as shown in the figure 4. Each update transmission uses some communication resources for its completion. We describe a score coefficient as an abstract measurement unit for the notion of importance regarding communication resources for weighting entities communication requirements. An entity's score coefficient is calculated at runtime using a combination of the actual network settings (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. This score coefficient of an entity is computed using the following formula:

$$ScoreCoefficient = Significance * CurrentNetworkCongestion$$

This generic notion of ScoreCoefficient is a value that defines whether the entity is important at the current time or not. 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 role by the game

designer through a configuration file. In case of congestion, entities are reassigned to groups.

F. Group Assignment Policy

The server side of the game engine maintains a collection of communication groups. When the game notices a drastic change in network load, the score coefficient of each entity is recalculated and entities are reassigned to new communication groups if needed.

An entity of reference (E.R) is used to trigger the group reassignment process for all the entities. The reassignment process is ordered only if the newly computed expected group for the E.R is different from its current group as explained in the figure 4. As general rule for getting the expected group for an entity, these conditions must be respected:

- **C1**: The entity's score coefficient is lower than the group's score coefficient's threshold.

- **C2**: The group is the one with the lowest score coefficient threshold among the groups respecting the condition **C1**.

This reassignment process is attempted every 5 seconds during the game session.
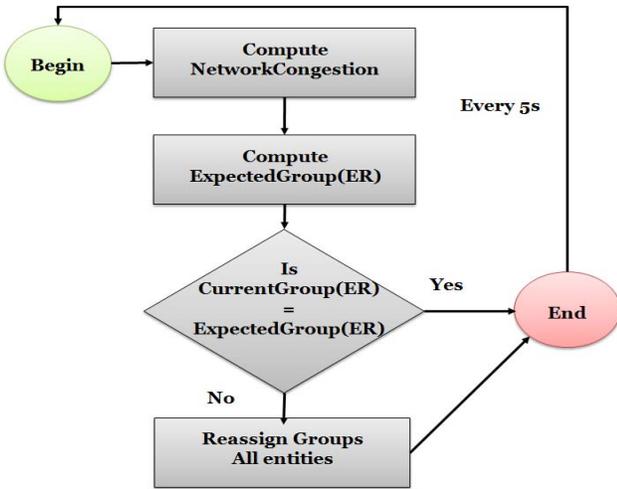

Fig. 4. Group assignment policy

The algorithm 1 shows the group reassignment algorithm for all entities in the game scene.

1) Traffic Congestion Computing: Our proposition is a generic framework that uses as adaptation metric a score coefficient calculated at runtime. The current congestion of the network is the main parameter for the calculation of this score, and represents the network overhead created by the traffic load at running time. Because our message passing is done using the UDP protocol, as a representative of network congestion, we monitor the packet loss in the network since in case of congestion some packets will be lost. Knowing that with UPD, there is no guaranty on the reception of the packets.

To monitor the packet loss at runtime, the server periodically sends a number of monitoring packets (100 packets, one

---

Algorithm 1: Assignment algorithm for all entities

```
/* Loop for all entities in the game
   scene                           */
for entity in entities do
   /* Compute the expected group of
      the entity                   */
   scoreCoefficient =
   significance(entity)*currentNetworkCongestion;
   expectedGroup = getExpectedGroup(entity,
   scoreCoefficient);
   /* Add the entity in the
      appropriate group            */
   if currentGroup != expectedGroup then
      expectedGroup.add(entity);
      currentGroup.remove(entity);
   else
      donothing();
   end
end
```

every 50 ms) 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 trigger the reassignment of communication groups for the entities changing the communication profile of the game.

G. A Short Example

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

- Cloud: represents clouds that are moving in the background. Less important; 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 duck as E.R and the formula in III-E to compute entities' 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 10ms, reducing the overall traffic load.

IV. EVALUATION OF OUR ADAPTATION APPROACH

The objective of this experiment is to evaluate the impact of LoD based adaptation in cloud gaming on the player's experience. In order to evaluate this impact, we observe and compare the reaction of players during a game session with and without the proposed approach.

A. My Duck Hunt

The video game *My Duck Hunt* has been developed to conduct this experiment. It is a competitive shooting game inspired from the traditional Duck Hunt video game [12]. The rules of the game are the following: five kinds of entities evolve in the game scene: the *reticle*, the *ducks*, the *flamingos*, the *gombas* and the *clouds*. The player controls the reticle and should point the reticle on the target and shut to kill. The game is divided in 5 rounds or waves of ducks. The player has to achieve the following goals:

- Kill as many ducks as he can. For each duck killed, the player gains points.

- Do not kill flamingos.

- Protect flamingos from gombas by killing gombas. Each flamingo killed result in point loss.

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

B. Participants

The test was conducted with 9 participants between 21 and 30 years old with an average age of 25.33. The distribution of players, based on their playing frequencies; it is given in the table I. Only one participant reported that he does not play video games. The other participants play games at least one time per week.

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

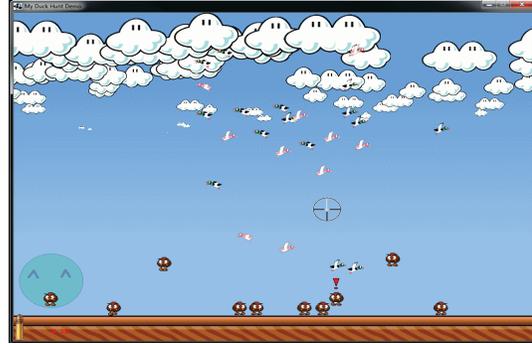TABLE I. PLAYING FREQUENCIES DISTRIBUTION

Fig. 5. "My Duck Hunt" video game

C. Protocol

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

- The candidates get a quick introduction of the game's rules through a demo version of the game.

- The candidates play one version then the other (this order is random for all the players). The candidates are not informed about the difference between the two versions. During the game, the candidates have to report when they perceive any type of bad quality of experience or interactivity shortage. They do so by holding the space key.

- At the end of each round, the candidates evaluate the quality of experience for the round. They give a mark between 1 and 5. 1 indicating a bad game experience and 5 indicating a good game experience.

- At the end of the experiment, the candidates are individually interviewed and asked to rate the global game experience for each game version by giving a mark between 1 and 5.

D. Network Configuration and Congestion Simulation

To be able to control the network environment, the pilot experiment is performed on a Local Area Network. In this LAN we have the game server machine, the client machine. Since we didn't deploy the server on a real cloud with WAN (Wide Area Network) connections, we need to simulate poor network settings of WAN. That is why we implemented a proxy. The proxy is used for network congestion simulation through delay, jitter 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 state updates, the proxy simulates a congested network by ignoring all the packets received while a 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 network congestion for the game, we just need to change this threshold value. 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 II. From 0 to 30 seconds, the proxy forwards 6000 packets per second; from 30s to 60s, it forwards 3000 packets per second, denoting a 50.

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

TABLE II.   PROXY CONFIGURATION FOR NETWORK CAPACITY

The game server is a Dell Precision M6500 with the following configuration: an Intel Core i7 Q 720 CPU and 4 Go of RAM. This is an experimental setup with a mono player game, meaning that the processing cost of the game can be handled by a server machine with these specifications. The only bottleneck we have is the one simulated by the proxy on the network link. The server platform is configured with the four following communication groups sorted by other of importance:

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

- **Enhanced group**: Entities with relatively high communication requirements but lesser than those in the optimal group. Frequency = 35 ms, threshold = 15.

- **Medium group**: Entities with average needs in network resources. Frequency = 40 ms, threshold = 70.

- **Degraded group**: Entities with the lowest communication needs. Frequency = 75ms.

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 during round.

- **H.B.0** There is no difference concerning the global game experience between the two versions of the game.

- **H.C.0** There is no difference in the ratio of time spent holding the space key per the session duration between the two versions of the game.

F. Experimental Results

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 rounds with $p-value < 0.5$. The difference between the game experience during each round when using LoD and without LoD is statically significant. The results of the t-test are summarized in the table III.

The hypothesis **H.B.0** is rejected by the t-test. The difference of the global game experience 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 also rejected by the t-test.

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

TABLE III.   RESULTS

The different in the ratio of the time spent holding the space key per the playing 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

According to the above analysis, we can see the effect of our approach on the players' interactivity with the game, therefore their QoE. In fact all the three hypotheses were rejected meaning that the players have perceived a significant gain on the game experience with the adaptation technique not only for each round but also for the overall game session. The instability introduced by changing the congestion degree at the proxy had less impact on the game quality for all the participants. Finally, these results validate our approach on improving the overall quality of interactivity of the game, showing that adapting the game traffic to the available network resources significantly increase the perceived quality of the game.

This approach is not perfect, one limitation is that, the whole adaptation can be very subjective, since the configuration of the platform (entity's weight, score coefficient threshold) is done manually by the game designer. 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.

V.   RELATED WORK

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

A. Packet Compression and Aggregation

Packet compression [13] tries to speed up transmissions by reducing bandwidth requirements. Aggregation 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.

B. Interest Management

To reduce both the traffic load in the network and the computational cost to process each game event, Interest Management techniques have been very rewarding [14]. In some game scenario, events generated are relevant only for a small fraction of the users. 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 [15]. 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.

C. Optimistic Algorithms

In order to be less depending on the real responsiveness of the network, optimistic algorithms for synchronizing game state at servers can be implemented in order to avoid delay perception at destination. 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.

D. Dead reckoning

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 r [16]. 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 impact on interactivity and playability of the game.

E. Games@Large

In Games@Large [17], A. Jurgelionis et al., 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 reduce the amount of data to be transmitted since only graphics commands are sent to the client, thus reduces 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.

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 analysed work has tried to couple the traffic reduction with a scheduling mechanism for an efficient message passing (messages with different importance) 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 efficient event delivery in client-server games with the cloud paradigm.

VI. CONCLUSION

In this paper, we have presented a new network aware adaptation technique for game state synchronisation in cloud gaming. We tackle the challenges of instability and shortage of network resources required to enjoy a classic cloud game. Based on LoD principles, our proposed approach uses entities organization model in order to minimize the effect of low or unstable network capabilities in maintaining game interactivity and improving player's QoE.

Future work is to support wider range of significances for an entity. We will need to implement a version of the adaptation scheme using a topology heuristics (distance to camera) as the significance of the entities. We will also perform a large scale experimentation in a multi-player cloud gaming environment where each player has his own network resources, his own camera and sees different angle of the game scene.

REFERENCES

[1] T. M. C. Geoffrey Raines, "Cloud computing and soa," 2009, [Online; accessed 31-Octobre-2012]. [Online]. Available: http://www.mitre.org/work/tech$_p apers/tech\_papers$_09/09_0743/09_0743.pdf

[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] 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].

[5] 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

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

[7] T5-Labs, "T5-labs official web page," http://www.t5labs.com, 2012, [Online; accessed 31-Octobre-2012].

[8] 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

[9] 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.

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

[11] 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$_2$

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

[13] 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

[14] 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

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

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

[17] 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