IEEE COMSOC MMTC E-Letter

Adaptive Streaming in Mobile Cloud Gaming
Shervin Shirmohammadi$^{1,2}$

$^1$Distributed and Collaborative Virtual Environments Research (DISCOVER) Lab
University of Ottawa, Canada, shervin@eecs.uottawa.ca

$^2$Machine Learning Lab (MLL)
Istanbul Sehir University, Turkey, shervinshirmohammadi@sehir.edu.tr

1. Introduction
Cloud gaming leverages the well-known concept of cloud computing to provide online gaming services to players, including mobile players who have more computational or download restrictions than players with dedicated game consoles or desktop computers. The idea in cloud gaming is to process the game events in the cloud and to stream the game to the players. Cloud gaming can be single player, where a user plays the game on his/her own, or multiplayer, where multiple geographically distributed users play with or against each other. Since it uses the cloud, scalability, server bottlenecks, and server failures are alleviated in cloud gaming, helping it become more popular in both research and industry, with companies such as OnLive [1], StreamMyGame [2], Gaikai [3], G-Cluster [4], OTOY [5], Spoon [6], CiiNOW [7], and others providing commercial cloud gaming services.

One of the challenges in cloud gaming is adaptive streaming of the game to the players. By adaptive, we mean that the server has to adapt the game content to the characteristics and limitations of the underlying network or client’s end device. These include limitations in the available network bandwidth, or limitations in the client device’s processing power, memory, display size, battery life, or the user’s download limits as per his/her mobile subscription plan. While some of these restrictions are becoming less problematic due to rapid progress in mobile hardware technologies, battery life in particular and download limit to some extent are still problems that must be seriously considered. Also, consuming more bandwidth or computational power means consuming more battery. In this article, we present some approaches for adaptive streaming in cloud gaming to reduce battery, processing, and bandwidth consumption. We start this by looking at the types of streaming that can be done in cloud gaming.

2. Game Streaming in the Cloud
There are currently three types of game streaming in the cloud: graphics streaming, video streaming, and their combination.

In graphics streaming, the game objects are represented by 3D models and textures, and these are streamed to players as needed. Rendering of the game is done at the client, but the game logic runs in the cloud and the state of the game (position and orientation of objects, as well as actions and events) is streamed to clients as an update message every 10 to 280 msec, depending on game genre and activity levels [8]. The advantage of graphics streaming is that except for the textures, the 3D object models and the update messages are small and do not require much bandwidth. So once textures are received, which happens only one time, game streaming uses very little bandwidth. Depending on the client’s storage capacity and the nature of the game, textures can be even preloaded at the client when the game is installed, reducing bandwidth consumption even more during game play.

In video streaming, the cloud not only executes the game logic, but also the game rendering. The resulting game scene is then streamed to clients as video, typically in the 20 frames per second range, and in some cases up to 50 frames per second [9]. The advantage here is that as long as the client can display video, which pretty much all smartphones and tablets and most other mobile devices today do, the user can play the game without needing 3D graphics rendering hardware or software. The disadvantage is that video is much bulkier than 3D graphics or updates, and requires substantial bandwidth [10], and although bandwidth is becoming more affordable, the battery life problem due to bandwidth consumption, mentioned in section 1, has to be dealt with.

It is also possible to use a hybrid approach and to simultaneously mix graphics streaming with video streaming, as is done in CiiNO, for example.

With the above explanations in mind, let us now present some approaches for doing adaptive streaming in cloud gaming for graphics streaming and video streaming, respectively.

3. Adaptations in Graphics Streaming
The network and client limitations mentioned in section 1 essentially imply that we can stream only a limited amount of information at a time. Streaming more information than that will not be possible since either the network cannot transport it or the client does not have enough capacity for it. As such, one approach to
adaptive streaming is to prioritize game objects and to first stream the most important objects in the context of gameplay. Traditionally, this has been accomplished by using Area of Interest management or similar region based techniques, whereby objects that are closer to the player or within the player’s viewing scope, strictly in terms of distance, are streamed first and with higher quality [11][12]. But such approaches do not consider the context under which the game is played. For example, when fighting an enemy in a jungle, the enemy has a higher priority than the trees and surrounding bushes, and should be updated with higher quality in terms of both resolution and update frequency, even if the enemy is further away from the bushes and trees. Distance-based approaches on the other hand would simply render with higher quality whatever is closer to the player, which is not necessarily the most important object for the player. Therefore, it is more logical to adopt a context aware approach whereby we classify actions in a game as activities like walking, running, aiming, shooting, etc., and we determine how important a specific game object is in the context of that activity, building a one-time apriori importance matrix [13]. Using this matrix, an object selection threshold can then be set, according to the bandwidth or client limitations (for example download limit), so that less important objects in the scene compared to the threshold will not be streamed, freeing up resources for more important objects. It has been shown that, compared to traditional distance based approaches, such an approach leads to a higher quality of gameplay in terms of game score [13] and that the object selection threshold can be set automatically using optimization [14].

Of course not streaming less important objects at all could have a negative effect in the visual quality of the game, as can be seen in Figure 1(I). It is therefore preferable to still stream the less important objects, but with a lower graphics quality, as shown in Figure 1(II).

To stream an object, we must consider that an object consists of two elements: 3D mesh model, and image textures. 3D mesh models are relatively small and in most cases can be downloaded quickly. In cases where the mesh model itself is large, progressive mesh streaming approaches can be used such that with each update message, the visual quality of the object increases [15].

Textures are the most bandwidth and rendering consuming elements of objects. To facilitate texture streaming, we can design adaptive approaches that would stream textures efficiently according to the mobile device’s battery life and/or bandwidth restrictions and the importance of textures, with textures having different importance levels. For instance, in the previous enemy in the jungle example, the textures of the enemy can have importance level 1 (most important), bushes and trees can have importance level 2, and the sky with clouds or the water in the lake can have importance level 3 (least important). We can then design an optimization algorithm that would stream textures according to their importance and bandwidth/client restrictions, in a progressive manner so that with each update message the resolution and quality of the texture improves up to a certain threshold determined by the said restrictions [16].

![Game scene with the less important objects](http://www.comsoc.org/~mmc/)

Figure 1. Game scene with the less important objects (I) not screamed at all (indicated in white), and (II) streamed with lower quality. [16]

We can further save battery life by limiting lighting effects and doing smarter control of game object brightness [17]. Specifically, we can choose the lighting effects that are the most computationally expensive effects, such as specular highlights, reflection, transparency, and shadows, and not apply them to the less important objects. In addition, we can make darker the less important objects and save display energy, hence increasing battery life without significantly affecting the gaming experience. It has been shown that such approaches can lead to 20% to 33% extension of battery life [17].

The above approaches save battery life by smarter streaming of the game graphics. To further save battery life, we can enhance the above approaches with streaming techniques whereby the wireless interface of the mobile device is turned off for a time during which the status of an object can be more or less estimated and
so there is no need to transmit update messages for that object. Turning off the interface saves more battery life. Dead-reckoning has been used for decades in games and simulations to determine when an update message needs to be received for an object. This can be used to design dead-reckoning based game streaming protocols that allow the wireless interface to be turned off when no updates are expected, and turned on only when updates are expected. Such approach can save as much as 36.5% of battery life [19].

4. Adaptations in Video Streaming
We can apply some of the above approaches for graphics streaming to video encoding at the server side, and stream a more efficient video to the client. For example, we can use the concept of object importance and not encode the less important objects in the video, leading to as much as 8% bandwidth saving for the client and 7% reduction of encoding time in the cloud [20]. We can also apply smarter brightness control to the video, such that the less important objects appear darker, saving display energy and increasing battery life with little negative effect on the player’s experience.

For a more efficient streaming technique, we can combine the idea of object importance with visual attention models, such as Judd's SVM-based model that has been trained using a large database of eye tracking data. Visual attention models are used when it is essential to understand where humans look at in a scene. In gaming, the player most of the time looks at the most important part of the scene, so such saliency models can be used, in combination with object importance models, to label scene parts with various priority levels. We can then encode each scene part with a different quality level, according to the priority of that part, and stream the resulting video to the client. Specifically, in each frame of the gameplay, we can consider the importance of each object and visual saliency features to decide which regions of the frame are more important for the accomplishment of the player’s current activity [21]. Then, we encode each region of the game frame with a different QP value proportional to the importance of that region. To make the technique less time consuming, in practice only the first frames of each GOP can be analyzed to find the appropriate QP values.

This is shown if Figure 2, where the image on the top has been encoded normally with a QP value of 30, but the image at the bottom has been encoded with Saliency + Importance approach using three QP values of 30, 35 and 40 for the high, medium and low importance macro-blocks, respectively. In the figure, S1, P1, and P2 represent the high quality regions of the saliency map, and the high and medium quality regions of the importance map, respectively. It has been shown that such an approach can reduce the streaming bandwidth by as much as 50% [21].

Figure 2. Game frame encoded by a single QP value (top) and with three QP values (bottom). [21]

5. Conclusion
In this article, we gave an overview of different game streaming techniques in the context of mobile cloud gaming. We presented application-level adaptation approaches, specifically for game streaming, which reduce battery or bandwidth consumption and therefore cope with the limited battery life and download limit of mobile players. As cloud gaming become more popular, supporting mobile players becomes more important and approaches such as those presented in this article need to be implemented to improve the quality of mobile gaming.

References
IEEE COMSOC MMTC E-Letter


Shervin Shirmohammadi received his Ph.D. degree in Electrical Engineering from the University of Ottawa, Canada, where he is currently a Full Professor at the School of Electrical Engineering and Computer Science. He is Co-Director of both the Distributed and Collaborative Virtual Environment Research Laboratory (DISCOVER Lab), and Multimedia Communications Research Laboratory (MCRLab), conducting research in multimedia systems and networking, specifically in gaming systems and virtual environments, video systems, and multimedia-assisted biomedical engineering. The results of his research have led to more than 200 publications, over a dozen patents and technology transfers to the private sector, and a number of awards and prizes. He is Associate Editor-in-Chief of IEEE Transactions on Instrumentation and Measurement, Associate Editor of ACM Transactions on Multimedia Computing, Communications, and Applications, and was Associate Editor of Springer’s Journal of Multimedia Tools and Applications from 2004 to 2012. Dr. Shirmohammadi is a University of Ottawa Gold Medalist, a licensed Professional Engineer in Ontario, a Senior Member of the IEEE, and a Professional Member of the ACM.