Communication versus computation: A survey of cloud gaming approaches

COMMUNICATION VERSUS COMPUTATION: A SURVEY OF CLOUD GAMING APPROACHES

Robert J. Grigg
School of International Game Architecture & Design
NHTV Breda University of Applied Sciences
email: grigg.r@nhtv.nl

René Hexel
Institute for Integrated and Intelligent Systems
Griffith University
email: r.hexel@griffith.edu.au

KEYWORDS
Cloud gaming, Cloud Game Engine Architecture

ABSTRACT

There still remains many challenges in creating an effective and efficient cloud gaming operation able to handle the new release games of today. There are also many existing paths to creating a cloud gaming architecture. In this paper some of the different approaches, that are beyond that of just remote rendering, are analysed giving insight into the operational approach of each technology. Currently there is a growing number of initiatives in cloud game architectures that vary in significant ways. Although there are many varying technologies, with a lot of promises, the ultimate goal of a cloud game engine is something unique to what has been before. It really is about providing a modular and scalable approach but within a controllable and sandbox like environment.

INTRODUCTION

To date *Cloud Gaming* has experienced both success and failure in delivering cloud-based gameplay to gamers. These operations still face challenges in bandwidth usage, the effects of latency on gameplay experience and the costs involved when the numbers of users supported per server remains quite low.
Most games have a real-time requirement with very low latency to end-users, whom are quite sensitive to quality of the gaming experience. The game engines running these games tend to demand high amounts of resources to render successfully at the expected Frame-Rates (FRs).
Current game engines are *plugged-in* or *ported* to the cloud which, although feasible, often introduces many inefficiencies in an online networked environment that already has its own problems and challenges. The game engine integration approach also fails to leverage the mass amounts of computational power that may be available in a *cloud gaming* cluster.
In this paper the various methods that underpin current *cloud gaming architectures* are analysed with the goal to bring an understanding of the various approaches and their inherent advantages and disadvantages.

WHAT IS A CLOUD GAMING PLATFORM?

A typical view of a *cloud gaming system* (see Figure 1) is one that renders the game scene on the cloud cluster and then transmits the encoded scenes to an end-user device via the internet or similar broadband connection. User input is taken from the end-user device and transmitted to the *cloud gaming cluster* so it can update the game state and render the next frame (see Figure 2). This architecture means that game-code is stored and executed on the server with the benefit of being able to deliver the game in a generic fashion suitable for a wider selection of end-user devices.

Figure 1: Cloud-Gaming Architectural Overview

In the *cloud cluster* an important differentiating factor is whether the system is created generically so it can plug into as many game products as possible. Alternatively the cloud gaming platform can be built to leverage internal game engine information and functionality for better performance, but this ties the solution to a particular game engine and therefore the need for porting the game product to that cloud gaming platform (Beer 2013). Creating a generic game plug-in system, that allows a wide range of games to be *plugged-in*, can negatively impact performance due to the additional steps required to capture game output as a black box process.

CLOUD FRAMEWORKS

Currently there exists a number of approaches that deliver game services from the cloud. The aim is to bring benefits a cloud platform gives other services, to games. This section will present core approaches used for delivering games from the cloud and discuss the associated advantages and challenges of each approach. The

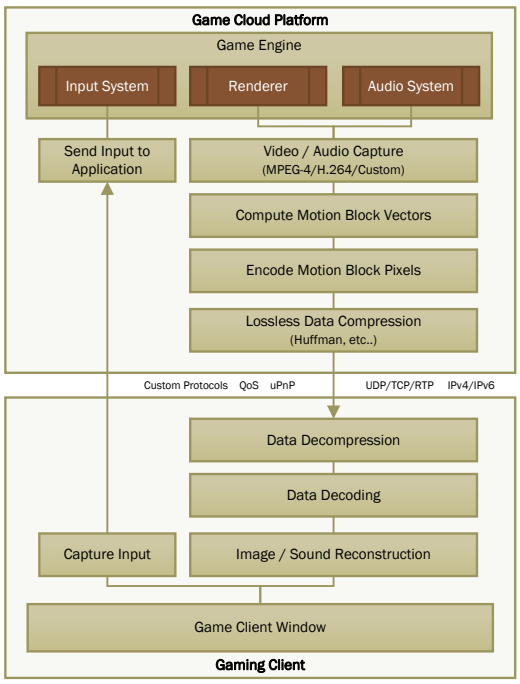

Figure 2: A Traditional Video Streaming Architecture

suitability and successful application of each technology may depend on the original application and whether it can handle user interaction and dynamic scenes. A lot of current work has leveraged remote rendering technologies of the past and a survey covering interactive remote rendering systems has been done (Shi and Hsu 2015). Cai *et al.* (2016) has completed a more broad study at defining cloud gaming, the ongoing research and challenges. This paper focuses on highly interactive and dynamic scenes, that is the cloud gaming category of the model (see Figure 3), and explore the options and challenges in creating a cloud gaming platform.

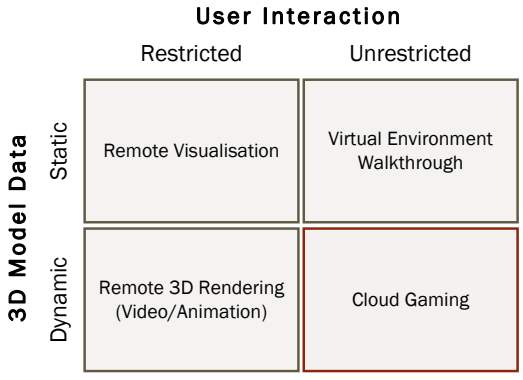

Figure 3: Categories of Remote Rendering Systems and Applications (Shi and Hsu 2015)

STREAMING MEMORY PAGES

The memory page level of an application is streamed by sending memory pages as required by the user application. The approach uses a statistical method to help predict what pages are required next by the client application. The handling of the memory pages is done by a Virtual Machine (VM) layer on the client-side which communicates with the server to retrieve the previously virtualized memory page. This technique makes it possible to be cross-platform for end-user devices that have a VM available. This process was called *Cloudpaging* by *Numecent* (Ahiska 2015) and brought to gaming by a spin-off company *Approxy* (Ahiska 2012) that was later reacquired by *Numecent*.

Cloudpaging allows for the download of the files as they are required and even the subcomponents of a large executable file transparently in the background. This means that, although the application does not start immediately the first time, the application will start after downloading less than 10 percent with an example such as Adobe Photoshop starting after 6.5 percent. *Cloudpaging* presents a way where the transferred *Cloudpages* may be proxied locally to allow for client devices, that share the local network, to make use of this if the same application is being used and even play offline if all relevant pages are available. This type of approach may be more compatible with existing Content Delivery Networks (CDNs) used today.

Security is by encrypting the *Cloudpages* and ensuring the end-user does not have the whole application at any point in time. This still sends all material to clients and therefore could be argued that it would still be vulnerable to piracy activities. Latency, conversely, is much less of an issue as the application is running upon the client device and slowed only by the VM layer it runs within.

STREAMING OF FILES

This approach works at the file level of an application and focuses on downloading files as required by the client-side application. The transfer of files can be defined by the developer, defined by the levels of a game, anticipated statistically or via prediction methods on what files are required and when. The aim for all these approaches is to minimise the size of the initial download before being able to play, as well as effectively stream in the background without negatively impacting the game (see Figure 4). Sometimes file streaming is done at the block level of a file, avoiding the transmission of large files when only a part is required, which can be especially useful when it comes to efficient patching. The streamed game then runs natively, or within a thin layer to handle file requests, and therefore is not impacted by latency.

Services such as *AWOMO*, *InstantAction*, and *Triton* are file streaming game services that are no longer in op-

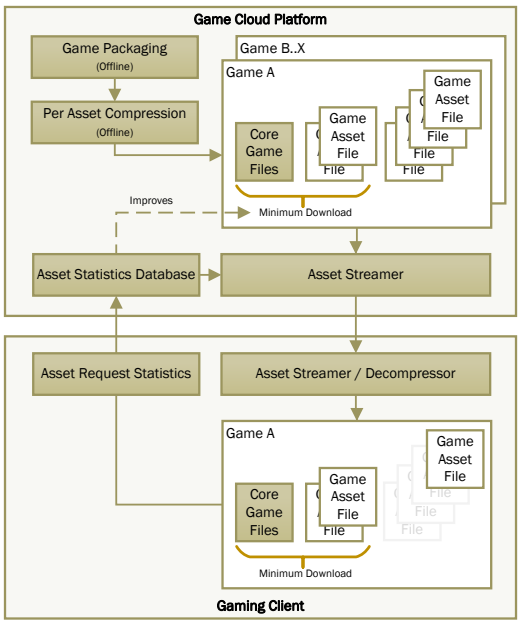

Figure 4: Cloud Gaming File Streaming Architecture

eration today. *InstantAction* was focused on being able to deliver games directly within a web-browser in websites such as Facebook (DiStream 2004; Walkden 2009). Exent, which had been involved with the *Games@Large* cloud gaming research (Jurgelionis et al. 2009; Nave et al. 2008), had primarily relied on file streaming for their services where they had technology to layer in-game advertisements to make games freely available.

Eximion (2006) provided a cloud gaming tool-set called *Kalydo*, allowing the packaging of an existing game so that it could be progressively downloaded, enabling players to commence gameplay as soon as the minimum number of files required was available. Access was provided directly through a user's web-browser and a small plug-in. Games like the Massively Multiplayer Online Role-Playing Game (MMORPG) *Runes of Magic* had a download size of over 8 GB, but it started playing after downloading only 200 MB of the game.

An even more effective way to deploy a file streaming solution is to use a *player prediction* based approach for file streaming as done by *Utomik* (2015), which uses player behaviour to build a model for what files are going to be required when. *Game Domain International* (2006) had used a statistical approach for a *A World Of My Own* (AWOMO) where games would be provided in a *Second-Life* like world that was aiming to be the best place to buy and sell games (The Steam Review 2007). Valve's *Steam* (2011) also depends on file/asset streaming at the core. Using a Distributed File System (DFS), the system includes manifest objects that determine the order of when files are required to be cached locally. If multiple players share a device then *Steam* will share the assets of the game as well. Also a move to deliver games, not just to the PC, but also to devices such as Smart TVs only required that the developer's game be compatible with a game controller (Newell 2003).

File streaming solutions aim to run natively and perform as though the gamer has installed them, once the initial download requirement is met. Transfer at the file level can mean specialist compression approaches that best suit the file-type, which can help maximise compression. Some file streaming solutions will run within a VM. The use of a VM may introduce a level of latency, but can enable playback of the game on different devices. Today there are a number of platforms in the cloud gaming space (see Table 1) that utilise asset streaming technologies and techniques in providing their service.

STREAMING OF GRAPHICS COMMANDS

The streaming of graphics commands, which is also referred to as *Graphics Streaming*, involves the intercept of graphics commands from OpenGL (Khronos Group 1997), Vulkan (Khronos Group 2017) or DirectX (Microsoft 2017) that are compressed and streamed to the client to be executed. The client is required to have a graphics processor capable of quickly rendering the image, which may not be appropriate for a Smart-TV, Set-Top Box (STB), or mobile device that has limited processing power. This is an advantage for a *cloud cluster* because it reduces the amount of processing it has to do and allows each server to support more users (Jurgelionis et al. 2009).

THINC used an approach of operating as a virtual video driver that captures, translates and sends render commands to thin-clients. THINC delivered superior performance over existing solutions of the time on both LAN and WAN environments to true thin-client systems where a Graphics Processing Unit (GPU) was not present (Baratto et al. 2005). Another example is the streaming graphics command architecture VirtualGL which intercepts OpenGL calls on the server to render the 3D parts on the server-side and then streams the resulting image frames to the client for playback (Commander 2012) using the GLS (OpenGL Stream Codec). The aim of the system was to allow for the visualisation of large datasets without having to send the data itself whilst also being able to share expensive rendering hardware more effectively amongst users. Pre-dating this were 2D streaming approaches on Unix-based X-server systems. Streaming graphics commands is a technology also explored in various projects to help handle complex scene renders by approaching this in parallel across multiple nodes in a cluster. Such research includes WireGL (Humphreys, Eldridge, et al. 2001) which led to the work on Chromium (Humphreys, Houston, et al. 2002). Later distributed approaches included OpenGL Multipipe SDK (MPK) (Bhaniramka et al. 2005) and ParaView (Cedilnik et al. 2006). Now with highly parallel servers, work may be more focused on having a larger

number of games render on a single node but the methods used may be of interest in addressing the challenges of the server to client communication and latency. Game audio is also a consideration and normally streamed via a capturing, encoding and streaming process which can create challenges when trying to re-sync the audio to the visual on the client device.
The streaming of graphics commands can be viewed as a custom communication protocol. This approach often means developing a proprietary *codec* to allow the streaming of graphics content with the advantage of very little encoding effort being required. The disadvantage is that bandwidth is subject to FR (Stegmaier et al. 2002). High quality games present more of a challenge with large numbers of commands per frame requiring methods of compressing and caching commands to reduce network throughput (Eisert and Fechteler 2008; Liao et al. 2016).
There is also a dependency on the client device being sophisticated enough to render the graphical commands which may require the addition of a more expensive GPU hardware within the client device. This can also impact how cross-platform the graphics command stream can be and introduces inconsistencies between different client devices operationally. Even with this consideration, however, the cost to the end-user is often less than purchasing all the modern day PC hardware required to run the game at a similar level of graphics quality.

STREAMING OF THE VIDEO IMAGE

The streaming of a video game, which is known as Video-Streamed Games On Demand (VSGOD), involves the cloud cluster rendering the 3D world to a 2D image that is then captured, encoded and compressed into a video format before transmission. The client receives the stream and decompresses using a decoder to replay the video. The client captures input from the user to return to the server (see Figure 5). The advantage of this approach is that thin-clients do not require special hardware, such as GPUs, although they may have cheap decoder chips that support quick video decoding (Cheng et al. 2004; Holthe et al. 2009; C. Huang et al. 2013b).
Research in the area of remote data visualisation is often informed by progress in fields like medicine where the use of MPEG approaches to help stream the rendered images is done and ways to improve the performance has been researched. The main focus was on the MPEG encoding and decoding process operating more effectively for 3D data visualisation. Of this the motion estimation algorithm component of MPEG was found to cost at least 50 percent of the time. Wallach *et al.* (1994) created an image representing the 2D optical flow of each pixel of the image to better the exhaustive search performed by the original encoder. Khan *et al.* (1996) presented a geometrical approach to calculating the motion vectors which resulted in a set of matrix operations based on the viewer movement that allowed motion vectors to be easily calculated. Cheng *et al.* (2004) reduced processing time with GPU assistance, using the z-buffer and camera position change between frames to project current frame pixels of a block into the projection space of the previous camera position and in doing so calculate the vector difference in movement. If the pixels of the block were found to be a poor match then it was marked as occluded and normal MPEG-4 motion estimation would take place whereby this location would be a good search starting point. Cheng's work viewed large static 3D models that algorithmically may not work as effectively given a dynamic and interactive game environment.
H.264/AVC-based video codecs, considered as a possible improvement for the work of Cheng's, was used in later research by De Winter *et al.* (2006) to stream graphical output of applications to thin-client devices. They combined a traditional thin-client protocol with real-time desktop streaming to handle 3D games with a performance that surpassed the classic thin-client protocol alone. The main problem with 3D gaming is the reliance on GPU hardware for acceleration, this being a missing component on thin-clients.
Earlier work surveyed latency and performance of video streaming to mobile devices (Lamberti and Sanna 2007). Shi *et al.* (2010) went as far as to say that the only disadvantage of video streaming is the interaction delay or latency. This problem largely comes from the

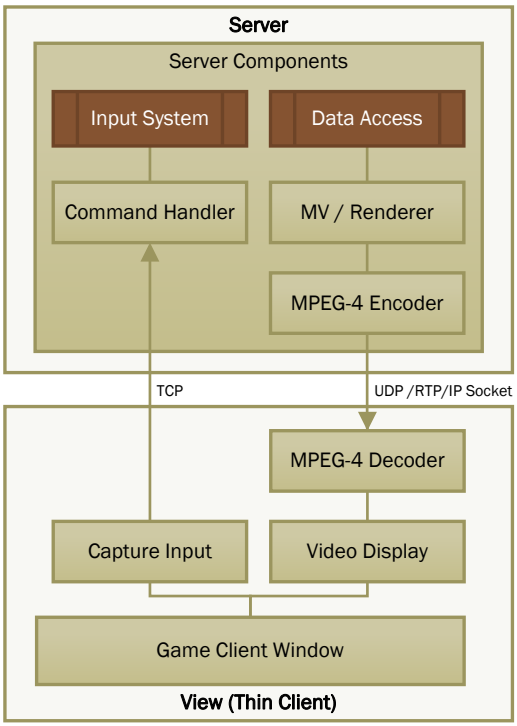

Figure 5: Cheng *at al.'s* Remote Viewing 3D Data-sets Architecture

on-demand nature of gaming that is unable to benefit from the use of large video playback buffers (Choy et al. 2012). This motivated work in how the Quality of Service (QoS) could be used to help ensure a good cloud gaming experience. Research such as Quality of Experience (QoE) (Jarschel et al. 2013) where downstream packet loss, followed by downstream delay, were found as the most important factors for a good gameplay experience. Jarschel highlighted that specific QoS settings are required for the type of game content being delivered and that cloud gaming's sensitivity to downstream problems greatly impact image quality, which is readily perceived by gamers. Wang *et al.* (2010) faced even higher research challenges in the mobile gaming area in providing a consistent QoE that led to methods of dynamically changing rendering and encoding parameters to control the computation and transmission requirements. This work in the Cloud Mobile Gaming (CMG) area identified factors impacting QoE in a model entitled Mobile Gaming User Experience (MGUE) (Wang and Dey 2012). This was later expanded to the model of Cloud Mobile Rendering User Experience (CMR-UE), which combined research with a Content-Aware Adaptive Rendering (CAAR) algorithm and demonstrated to work effectively over 3G mobile networks (Liu et al. 2014).

HYBRID APPROACHES

A number of successful *cloud gaming* operations have taken an approach that combines different methods to obtain the best result. In the following sections will be presented some of the hybrid approaches used in the past and present.

VIDEO STREAMING AND CLIENT PROCESSING

Another technique is to perform some of the rendering on the server and then post-rendering or post-processing operations, that are generally limited by the thin-client's processing power. The approach of video streaming with post-processing steps is somewhat of a hybrid between the streaming of video and 3D graphics commands. Rendering of 3D graphics is still performed on the cloud servers while a range of optional post-rendering options can be performed on the thin-clients.

MODEL STREAMING AND POINT BASED STREAMING

One client post-processing approach is for the server to pre-select structural scene information in an optimised fashion for streaming to the client to perform the final visualisation stages. Such techniques may send selected 3D model's Level of Details (LODs) to reduce the amount of data for lossless compression and transmission to the client (Engel et al. 1999; Prohaska et al. 2004). Extremely large data-set techniques, that store data in a pre-computed hierarchical format, may suit static elements of a game scene but may need further development to support highly dynamic or destructible scenery (Prohaska et al. 2004).

Another method is to construct a point-based representation of the data from the scene's mesh-based models and match the density of the points produced to the client's screen resolution (Duguet and Drettakis 2003; Shi, Kamali, et al. 2010). Duguet and Drettakis's work demonstrated filtered delivery from 4.7 million polygons on the server to 1.3 million points received by the mobile Personal Digital Assistant (PDA) client. The client still performed rendering which included tasks like shadow mapping at 2.1 F/Sec (Frames per second) which would not have been possible otherwise (2003).

ENVIRONMENT MAPS, DOUBLE AND 3D WARPING

Environment maps, that are commonly used in gaming, is another approach where the server is responsible for generating based on the 3D scene to represent any view angle from a fixed point. This is seen in QuickTime VR (S. E. Chen 1995) with the streaming of a chain of environment maps to allow navigation and used in games from Red Orb Entertainment (Simon 1997) and work by Boukerche *et al.* (2006). The method allows for highly detailed pre-rendered scenery to be displayed to the user. The drawback to this is the difficulty in adding dynamic objects to the scene or in changing the lighting dynamically within the environment.

Chen *et al.* (1993) proposed the method of asynchronous rendering where the client can perform intermediate rendering, such as *3D Warping*, independent of the server rendering step and therefore address latency issues. The approach used the z-buffer to re-render viewpoints that are a small change from the current position using an image warping algorithm that increased the responsiveness of the system but presented challenges in minimising image error (McMillan 1997). The technique required the server to send more information about the scene to the client but was found to have significant FR improvements where their examples present a server running at 5 F/Sec, whilst the client was running at 60 F/Sec and maintaining a high level of quality (Mark et al. 1997). The research led to work with low-powered mobile devices and the aim of enhancing their 3D graphics abilities (Bao and Gourlay 2003; Chang and Ger 2002).

Another approach involved warping within the framework of the H.264 encoder video stream. This also delivered an increased performance of up to twice the FR upon the client and suggested that future improvements may involve the development of a custom *codec* (Giesen et al. 2008). Work built upon this used prediction of the next point of view which increased the effective range that the image warping could take place within (Shi, Ka-

mali, et al. 2010). This approach is somewhat similar to *Double Warping* where two depth images are rendered and sent from the server to represent the current position and selected reference. The *3D Warping* process is done twice and the result is an interpolation between the two (Mark et al. 1997).
These various approaches discussed address latency in an asynchronous manner to the server but for highly interactive 3D game titles their use may be quite limited (Shi, Kamali, et al. 2010). The algorithms are still of interest as they may increase FR when dealing with client devices capable of high FRs and therefore the travel distance of in-game objects between frames being suitably minimised.

VIDEO STREAMING AND CODEC PERFORMANCE

At the heart of video streaming of games is the *codec* being used and how it is configured. There are a number of challenges with a *codec's* performance when an application is highly interactive like that of a video game. Visualisation applications of large data-sets are an area where there has been development of interesting algorithms to enhance the performance of *codecs* or generally increase the responsiveness of a system. Critically a lot of these methods find their limitations when faced with complex and highly dynamic scenes such as those found in modern games.
One consideration is whether the server checks the type of client device and then pre-scales the imagery to match the client device resolution. This can have an impact when there is a lot of devices connected with small resolutions. In this case you do not want to waste processing time creating and encoding detail that is not visible on the client (Jurgelionis et al. 2009).
Enhancements to encoding can lower the delay and complexity which helps reduce issues from latency (Nave et al. 2008). As an example H.264 can perform encoding with CABAC (Context-Adaptive Binary Arithmetic Coding), that uses a form of entropy encoding used in H.264/MPEG-4 AVC video encoding (Said 2004), which increases the compression efficiency of the *codec* but has the negative effect of increasing the computational time at the decompression stage on the client (Eisert and Fechteler 2008). The processing time required by the *codec* on the server needs to share processing time with more than one game where each may need several streams created, encoded and transmitted (Jurgelionis et al. 2009). Off-loading processing requirements and reducing bandwidth use are desirable when trying to maximise the number of users supported per server.
Codec algorithms are sometimes not suited to be implemented in parallel form and therefore restricted from execution on GPUs efficiently. Jules Urbach from *OTOY* (2011) developed their own proprietary codec called *ORBX* to run well on GPUs encoding a 3D scene. Other motivations include handling higher video resolutions and colour depths where open standards such as Google's VP10 (Sharabayko and Markov 2016) and Kronos Group's OpenMAX (Group 2015) are helping achieve better performance.

3D GAME WORLDS - LEVERAGING SCENE KNOWLEDGE

When encoding video using the H.264 encoder there is approximately 50 percent of the processing time in finding and calculating the movement of motion blocks. The game engine's scene information can be used to know where these blocks are moving to and therefore greatly reduce the processing time required in this stage of encoding (Jurgelionis et al. 2009). The speed of video encoding can be improved via encoding objects further away from the camera at lower levels of quality (Noimark and Cohen-or 2003). It is also possible to selectively omit game objects from rendering on the server altogether and therefore reduce the encoding time and Bit-Rate (BR) required for transmission (Hemmati et al. 2013). Some of these techniques and algorithms have also been developed from a client post-processing view-point which can put more processing requirements upon the end-user device.

CLOUD GAMING PLATFORMS

The area of *Cloud Gaming* has been developed largely privately, avoiding open-source development approaches. Contrastingly *NetFlix*, that operates in the area of on-demand streaming of media for TV and films online, builds upon a number of open-source projects for their business platform (Hastings 1997). Today there are a number of platforms in the cloud gaming space (see Table 1) that utilise various technologies and techniques in providing their service.
In this section eight examples will be presented that cover open-source platforms called *GamingAnywhere* and *LiveRender*, ex-industry player *OnLive*, current industry players *Sony's PlayStation Now*, *Blade's Shadow*, *AirConsole* and *GameFly* (previously known as *PlayCast*), and results from an older European research project called *Games@Large*.

GAMINGANYWHERE

Currently there exists very little in open-source options for cloud gaming platforms, but one called *GamingAnywhere* (2013) was made available by Huang *et al.* (2013). This follows a traditional approach of using readily available libraries to capture video and audio for streaming. The aim was for an extensible, portable and configurable platform made from open-source packages (see Figure 6).

Table 1: Cloud Gaming Operations

Platform	*Link*	*Stream Type*
Agawi (Google Buyout)[b]	-	Video
AirConsole	airconsole.com	Files
AWOMO (GDI Game Domain International) [b]	-	Files
CiiNOW (Google Buyout)[b]	ciinow.com	Graphics
Cloud Union	cloudunion.cn	Video
Community Cloud Development Kit[a]	github.com/monguri/CCDK	Video
Exent (FreeRide Games, GameTanium)	exent.com	Files
Gaikai (Sony Buyout)[b]	-	Video
GameFly (Merged with PlayCast)	gamefly.com	Video
GameNow (Ubitus)[b]	ugamenow.com	Video
Games@Large (Exent)	exent.com	Graphics/Video
GamingAnywhere[a]	gaminganywhere.org	Video
Geelix LiveGames	-	Video
GFace (Crytek)	gface.com	Video
G-Cluster[b]	-	Video
Happy Cloud	thehappycloud.com	Files
InstantAction (GarageGames)[b]	-	Files
Kalydo (Eximion)[b]	-	Files
LiquidSky	liquidsky.tv	Video
LiveRender[a]	github.com/llfjfz/LiveRender	Graphics
NDS's *Xtremeplay* (Cisco Buyout)	-	Video
NVIDIA GRID	nvidia.com	Video
Numecent (Merged with Approxy)	numecent.com	Memory/Files
OnLive 1/2 (Sony Buyout)[b]	-	Video
OTOY	otoy.com	Video
PlayGiga	playgiga.com	Video
PlayKey	playkey.net	Video
PlayStation Now/Remote Play (Sony)	playstation.com	Video
Shadow (Blade)	shadow.tech	Video
Steam (Valve)	steampowered.com	Files
StreamMyGame	streammygame.com	Video
t5 Labs	t5labs.com	Video
Triton/Game xStream (DiStream)[b]	-	Files
Ubitus	ubitus.net	Video
Utomik	utomik.com	Files

[a]Available for developers as a SDK or open-source platform

[b]Operation no longer available or discontinued

GamingAnywhere was found to have a per-frame processing delay of 34 ms which was three times faster than *OnLive* (Perlman 2011) and ten times faster than Tenomichi's *StreamMyGame* (Faria 2007) whilst transmitting lower levels of network traffic. The final image quality was better than *OnLive* and *StreamMyGame* at both 3 dB and 19 dB lower in final video quality respectively. The platform also supports clients that are *observers* which was a popular mode on services like *OnLive* where it could be delivered via a URL and viewed through players like VLC (VideoLAN 2010). The *observer mode* is where one encoded stream is shared with a number of users via a *unicast* method that is one of two modes *GamingAnywhere* supports. The other mode uniquely encodes streams for each game-player where the independent processing required may limit a server to a little over ten users comprising of twenty streams approximately (C. Huang et al. 2013b).

StreamMyGame offers end-users the ability create a MPEG-4 based streaming server of their own PC games to other devices in the home and includes abilities to record the gameplay and broadcast the games for spectators. The streaming typically has BRs of 4 Mb/s at XGA resolution. In the *GameAnywhere* platform communication can be layered on the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). UDP requires three streams which uses RTSP or Real-Time Streaming Protocol (Schulzrinne, Rao, et al. 2013) to deliver encoded frames over TCP along with two RTP or Real-Time Protocol (Schulzrinne, Casner, et al. 2003) over UDP streams for the audio and video. Over TCP alone the encoded frames, binary data and RTP/RTCP (Real-Time Control Protocol) packets for the audio and video are interleaved in the same stream

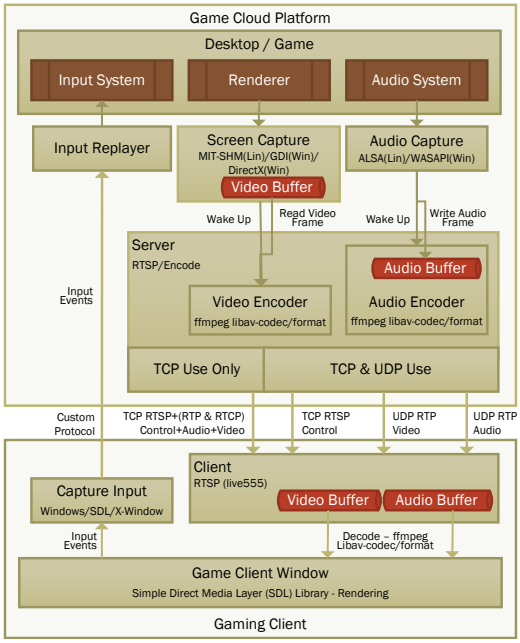

Figure 6: *GameAnywhere* System Architecture

and therefore only need one network connection instead of three (C. Huang et al. 2013b).

Huang *et al.* (2013) highlighted that they could improve their architecture by reducing the synchronisation overheads and by adding methods to better handle packet loss via a rate control algorithm for the varying quality of the remote player connections. Many approaches involve video encoding and decoding at various levels of compression and input resolution. The success of the compression, that if mismatched to the connection's feasible BR, can quickly lead to optical deterioration (Baun et al. 2010).

Later work involved the integration of *Live555* (Live Networks 1997) library replacing *FFmpeg* (Bellard 2000) to enable an easier implementation of a system to dynamically control the FR and BR of the codec depending on the user connection. This resulted in 30 to 46 percent better performance results over their baseline implementation in data centre bandwidth challenged situations (Hong et al. 2015).

LIVERENDER

LiveRender is an open-source project (Liao et al. 2014) handling the streaming of graphics commands, model data comprising of vertex and indices, and the texture data in an approach (see Figure 7) they call Compressed Graphics Streaming (CGS). This is a technique that streams graphical commands but it attempts to address the use of this method for modern games where the number of draw commands and associated data becomes quite large. Liao *et al.* (2016) states that a game like Trine (Frozenbyte 2009) would consume an average bandwidth of 15 Mb/s when streaming graphics with only basic lossless compression applied.

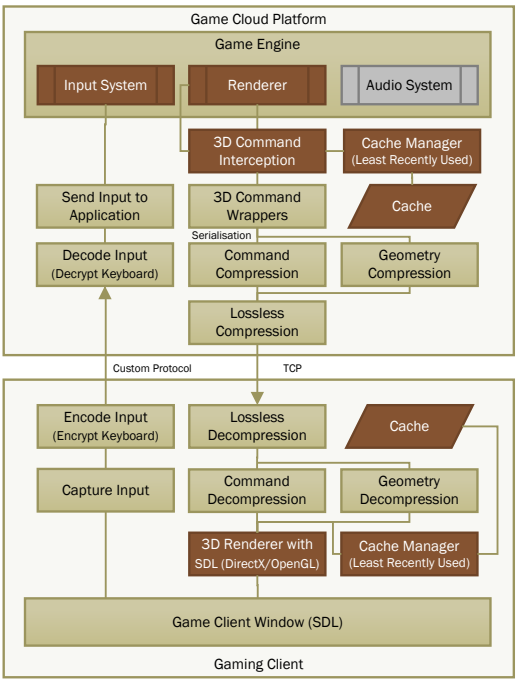

Figure 7: LiveRender Compressed Graphics Streaming Architecture

The *LiveRender* approach incorporated caching of recurring graphics commands, geometry and texture data where the Least Recently Used (LRU) caching algorithm resulted in the smallest cache of well under 5000 elements with a hit rate of approximately 95 percent, compared to *First In First Out* (FIFO) at 40 percent and Least Frequently Used (LFU) at 5 percent. The compression of graphics commands are done both intraframe which benefits static models, and interframe which is beneficial for animating models. With the geometry data *LiveRender* leverages progressive algorithms that represent the models at different LODs and which utilised the *QSlim* algorithm by Garland and Heckbert due to the algorithms advantages of speed while maintaining model quality (1997).

The application of their compression and caching approaches achieved a saving of 52 to 73 percent with no difference in video quality whilst having the expected latency introduced. When compared to video streaming this offered a BR reduction of 40 to 90 percent whilst maintaining better video quality and lower response delay.

The performance of *LiveRender* was superior in bandwidth limited situations where at 5 Mb/s latency was approximately 118 ms compared to normal graphic streaming taking approximately 39 ms longer, and video streaming having two to three times the latency of the compressed streaming approach. One of the reasons behind *LiveRender* being lower in latency than the com-

parable video streaming solutions is that as soon as the stream of graphics commands starts being received the client can commence processing each of the render instructions, and once the *end of frame* is received the image can be displayed immediately. An exception to this is in the situation of packet loss due to the use of TCP for communication, where a 10 percent loss would effectively double the impact of latency. Latency was impacted by their interframe interpolation method, this being where they could set how many derived frames (D frames) are created between original frames (O frames) where geometric data is not transmitted on D frames but instead interpolated. The results of having one D frame between each O frame was deemed optimal giving a bandwidth saving of 20 to 40 percent where any significant bandwidth savings disappears beyond two D frames.

ONLIVE

OnLive was the most famous of cloud gaming services starting in 2009, leading to insolvency in 2012 and then relaunching in 2014. In 2015 the company was brought out by *Sony*. The service delivered leading triple-A titles using their video streaming technology, although obtaining content from a number of publishers remained elusive (Hollister 2012; Lowensohn 2015). Financially they had faced challenges in the costly exercise of providing *OnLive* game consoles as well as localised data-centres throughout the world to maintain a responsive service. It was found that users need to be within a 1500 km radius of a data-centre and have a connection capable of 5 Mb/s to experience good gameplay. This includes the connection latency being under 150 ms for general games and under 80 ms down to 35 ms for games such as a First Person Shooter (FPS).
Shi *et al.* (2015) stated that the cloud game streaming service *OnLive* was measured at 6.49 Mb/s on average when tested with *Unreal Tournament 3* indicating that the bandwidth use was well above BRs such as with the x264 codec at 1.5 Mb/s for comparable quality. The research, furthermore, aimed to increase quality through the use of 3D warping and double warping techniques which they proposed should replace the original motion estimation stage of the *codec*. The *OnLive* operation relied on hardware encoders using their own proprietary compression technique.

PLAYSTATION NOW

Sony launched a new service called *PlayStation Now* in 2014 that allowed the delivery of older PlayStation 3 (PS3) games to be video streamed and played on the newer PlayStation 4 (PS4) console. The games are run and rendered in the cloud and sent to the PS4 requiring a connection with around 5 Mb/s in bandwidth (Sony 2014). At the same time technology allowing for games to be streamed and played on different devices, that they called PlayStation Remote Play, would stream from your console to other compatible devices in your the home.
Today gamers can use the service to play both PS3 and PS4 games that *Sony* has to offer. The games can also be streamed and played on PCs and *Sony* Smart-TVs as well not requiring the need for a console. The payment model is subscription based that currently costs around €17 per month and allows game progress to be maintained across devices.

SHADOW

A recent French start-up company *Blade* (2017) runs a service called *Shadow* that virtualizes Windows 10 desktops in high performance cloud powered hardware that consists of supplying a dedicated *NVIDIA* GTX 1070 GPU per user rather than *NVIDIA's GRID* (J. Huang 1993) division of GPUs amongst multiple users. The company raised $57.1 million for their subscription based service that ranges from €25 to €45 per month (Dillet 2017). The service requires that you have a Fibre To The Home (FTTH) connection and can support a bandwidth of 5 to 25 Mb/s enabling the delivery of HD images at over 100 F/Sec with a connection latency below 16 ms. Currently the average daily use per user is 2.5 hours for what is currently a more gaming focused subscription base. Like *Onlive's* later move they want to focus on attracting casual and business users as well.

AIRCONSOLE

AirConsole is a browser based service that started in 2015 that uses your smartphone as your game controller and turns your Smart TV, or web-browser, into the console. Due to the system relying on web-browser technologies, these games are written with tools like Flash, HTML5, Construct 2 or Unity3D, using their WebGL build option where game content is streamed as files to the web-browser cache. They currently have a free use and subscription, of $2.99 per month, that disables advertisements and enables access to special content and early access to new games. The service has games that range from 1 to 8 players with a console like party game focus, where a subscriber's privileges are shared with other players in the same session (N-Dream 2015).
AirConsole has been active with the Global Game Jam and with competitions encouraging content creators to create or port games to the platform. The technique is the smart use of existing technologies although feedback on how modern day smartphones feel as a controller are mixed due to the current lack of haptic feedback these devices currently offer and some games may demand. *AirConsole's* use of mobile devices as controllers presents a latency challenge that they approach in re-

liable and latency minimising steps of first using *long polling* where controllers send server web requests that remain unanswered until the server has new information which the controller receives and immediately opens the next request, secondly change to *WebSockets* if able in the background, and finally use the least latent option *WebRTC* if possible.

GAMEFLY

GameFly acquired *PlayCast* in 2015 who emphasised the importance of being able to help users and enhance everyone's gaming experience. They see this as a layer that can provide hints and tips as overlay graphics (Beer 2011, 2013). This important point highlights that these systems should interact effectively and may involve the input from several separate servers. The same could be applied to the Graphical User Interface (GUI) layer of a game, which needs to remain the most readable and in most cases changes the least would suit being sent separately from that of the in game footage layer.
PlayCast's original VSGOD service launched as *SingTel's ESC* for customers in Singapore with fibre or broadband connectivity of 10 Mb/s and above. The games were executed, rendered and streamed in *SingTel's* data centres as a MPEG stream to consumers to play games using a PC or *SingTel's exCite* TV service (Beer 2011).
De Beer (2013) states that in making a success in cloud gaming the major areas of improvement are in streaming efficiency and game porting efficiency. The financial impact can be significant given how many streams are supported per data-centre server, this can be seen from *PlayCast* cost per user stream based on 100k subscriptions (see Table 2). This highlights how important the scalability of the technological components are to the viability of the business model. This includes being able to deliver effectively to as many different end-user devices as possible. Finally, and most importantly, is the gaming experience of the user with the perceived smoothness of gameplay and responsiveness or minimal latency.

Table 2: Streams per Server - Cost per Stream

Players (Streams) per Server	20	15	10	5	1
Overhead per Stream per Month	$9	$12	$19	$37	$185
Overhead per Stream per Subscription	$0.5	$0.6	$0.9	$1.9	$9.3
Viable Operation?	Yes	Yes	Maybe	No	No

GAMES@LARGE

From 2006 to 2010 a European funded project called *Games@Large* (G@L) focused on producing a platform for streaming video games. This project aimed to be cross-platform and support a variety of devices from STBs and Handheld Devices (HDs) through to modern end-user devices including mobiles and computers. The *G@L* system communicates data for multiple games from a single server with low latency and employs end-user QoS. Depending on the end-user device this can be done either using a stream of graphical commands or a stream of image data where both are delivered using UDP based RTP (Jurgelionis et al. 2009; Laikari et al. 2010; Nave et al. 2008).

G@L GRAPHICAL COMMAND STREAMING

Graphical command streams rely on end-user device having the ability to handle things such as hardware acceleration in executing each of the DirectX or OpenGL commands successfully. The *G@L* project found this to be the lower latency approach between command and video streaming even with the use of TCP rather than UDP. The approach intercepted DirectX or OpenGL commands using delegate objects engineered to virtualize commands so as to avoid synchronous calls and therefore reduce the overall total number of commands required to serialise, compress and transmit (see Figure 8). They found games not to be playable when streaming graphics commands without compression and client-side caching mechanisms.

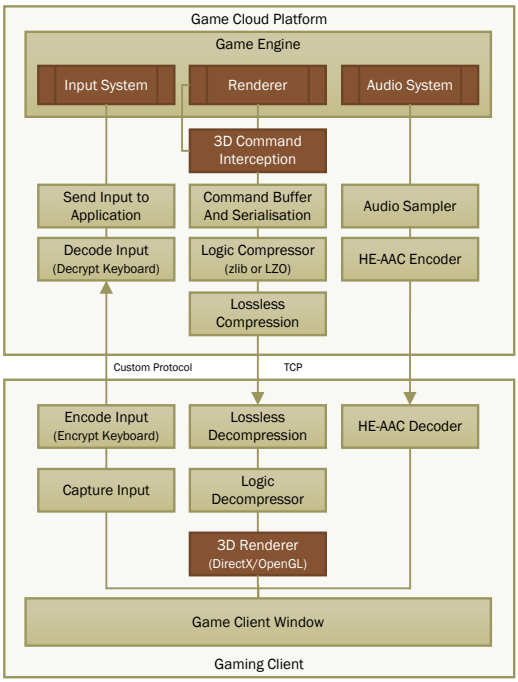

Figure 8: G@L OpenGL/DirectX Command Streaming Architecture

The end-user experience in *G@L* was found to clearly map to the FR of the game. With the streaming

of graphical commands the FR is related to the network bandwidth being used and for high FR games the bandwidth ranged from 6 Mb/s at 20 F/Sec through to 80 Mb/s at 2 F/Sec making this approach not as viable when streaming multiple high-end games from a single server whilst at high FRs. Later experiments had quite wide ranging bandwidth usage results from 5.2 Mb/*frame* to 847 Mb/*frame* which at decent FRs is high. To address the performance problems work was done on both the lossless compression of the commands as well as the virtual representation of the graphics card upon the client-side (Eisert and Fechteler 2008). Video streaming overcame the FR to bandwidth challenge allowing multiple streams to be sent from a single server.

G@L VIDEO STREAMING

G@L video streaming (see Figure 9) uses standard H.264 (Marpe et al. 2003) video and HE-AAC-v2-audio (Meltzer and Moser 2006) codec enabling delivery to a wide range of end-user devices where the time between creating the image frame and the user being able to view it is most critical. To synchronise the audio and video streams the RTCP sender report packets delivered in each channel and mapped to high-resolution NTP (Network Time Protocol) timestamps enabling the client to synchronise channels.

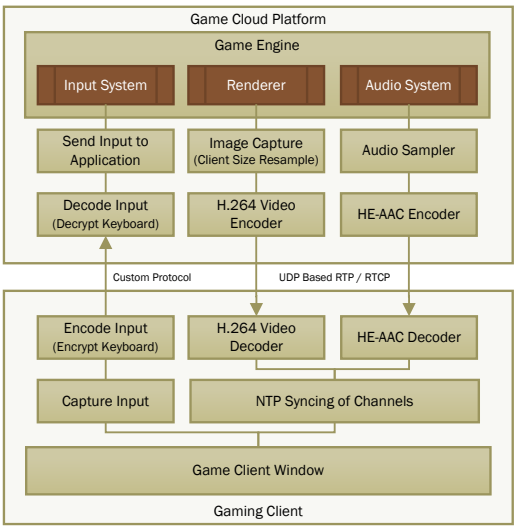

Figure 9: G@L Video Streaming Architecture

To help prioritise communication for low latency and high throughput, the *G@L* solution utilised WiFi Multimedia (WMM) subset of the *IEEE Wireless LAN 802.11e* (Mangold et al. 2002) standard for QoS approach to prioritise network traffic. This results in improving communication performance when low background network traffic is present due to the way QoS categorises the activity into four groups with appropriate prioritisation. Without QoS support on the network any other application would compete and detract negatively on the gaming experience, especially over a wireless network (Jurgelionis et al. 2009).

G@L aimed to achieve easy discovery of the networked services and used UPnP or Universal Plug and Play (Open Connectivity Foundation 2008) to achieve this. This resulted in the easy discovery of services available to client devices connected on the same network (Jurgelionis et al. 2009). The user input was encrypted on the client device using RSA (Rivest et al. 2009) encryption for the keyboard, otherwise data such as the mouse position would be sent unencrypted.

Now the *G@L* project only exists as publications but one of the main industry partners in the project, an Israel based company Exent (Levgoren 1992), is believed to be using the technology in a private business with their *GameTanium* mobile play service along with its *FreeRide Games* (Exent 2012) portal.

CLOUD CHALLENGES

The exploration of current research in the area of cloud gaming and reviewing what is currently best industry practise helps to form a picture of the challenges and problems faced by the world of cloud gaming. The immediate challenges of ensuring an acceptable gameplay experience motivated research into new services such as Quality of Experience (QoE) where downstream packet loss, followed by downstream delay, were found as the most important factors on the gameplay experience. Jarschel *et al.* (2013) highlighted that specific QoS settings are required for the type of game content being delivered and that cloud gaming's sensitivity to downstream problems is due to the impact on image quality which is readily perceived by gamers. In the following section the components of a cloud gaming architecture will be presented.

CLOUD GAMING COMPONENTS

From this survey there has been identified a number of challenges, opportunities and possible features that are presented as the components of cloud gaming (see Figure 10). The components cover functionality currently seen in existing cloud gaming operations through to proposed future features. There is still a major focus around bandwidth consumption and latency between the user's input to the response viewed on the user's screen.

Bandwidth demands can vary a lot depending on the amount of activity within the game but more importantly on the current and future resolutions of games where 720p (SD) needs approximately 5 Mb/s, 1080p (HD) at 10 Mb/s, 2160p (4K) at 25 Mb/s and 4320p (8K) at 85 Mb/s, where both 3D and VR versions doubling this. The increased speed of broadband will eventually be something that addresses this, but the need to have a very fast response between the time the

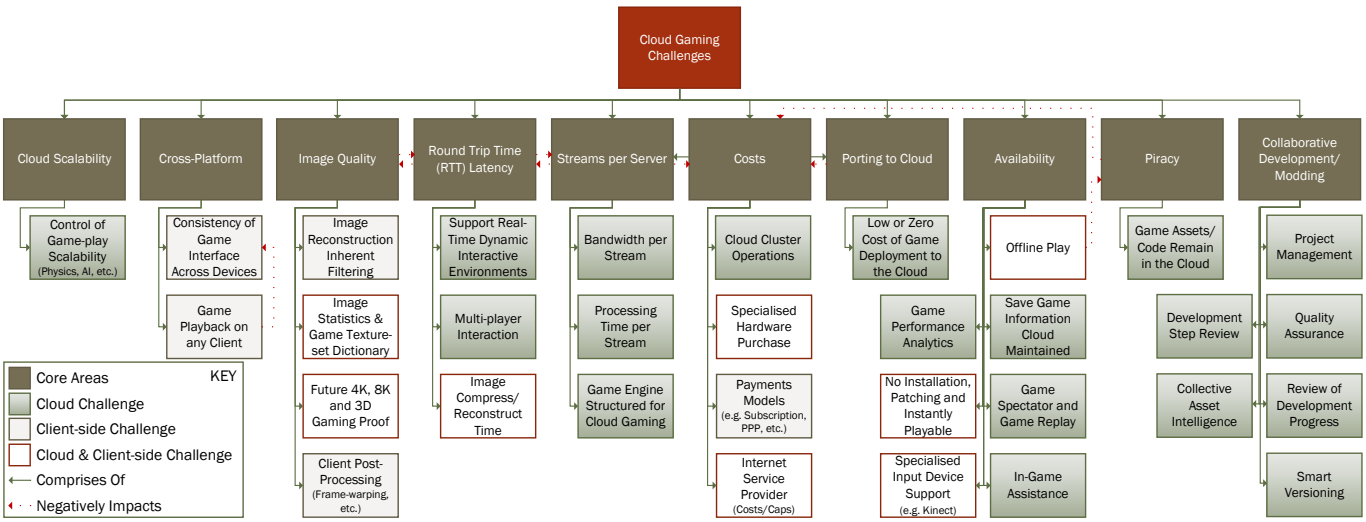

Figure 10: Overview of Cloud Gaming Challenges

user hits the key and the time the user sees the updated picture on their device is one that often means expensive set-ups of localised servers to ensure that latency is minimised. The latency can be greatly impacted by a bad connection, other traffic on the network or the use of a poor underlying protocol for transmission.

Costs for the operator is an important topic which encompasses all the aspects of running operational data-centres. This includes calculating the number of concurrent players, the geographical location of each of the data-centres, the quality of the game, frames-per-second and resolution.

THE IMPACT OF LATENCY

To deliver a great user-experience *latency* is highlighted as one of the big challenges to a cloud gaming system. For delivery of a game of the FPS genre a sub-100 ms Round Trip Time (RTT) is required as some studies found player accuracy and number of kills reduced by 50 percent with a latency of 75 to 100 ms, and players becoming annoyed at latencies around 200ms (Beigbeder et al. 2004). *OnLive* was found to differentiate server resources available on a game-genre basis to provide acceptable levels of latency based on the game requirements (K.-T. Chen et al. 2011). Research by Lee at al. (2012) produced a model of *cloud game friendliness* that analysed games on both how the screen changes spatially and the frequency of the player's input.

Chen *et al.* (2011) defined a methodology for evaluating the *real-timeliness* of cloud gaming systems to compare the performance of *OnLive* and *StreamMyGame*. This author found that *StreamMyGame's* level was not acceptable with encoding at 720p taking 400-500 ms and general performance twice as bad as *OnLive* which was acceptable if the network delay was not too great. The *OnLive* performance was thought to be due to specialist server hardware being used and emphasised the large investment cost for the hardware required.

Choy *et al.* (2012) further researched existing internet infrastructure and found that latency-sensitive cloud games need better infrastructure by equipping CDNs with the appropriate hardware. It was found that CDN edge-servers currently have low computational abilities and generally lack GPUs. Most CDNs are designed for large numbers of network connections and not the heavy processing required in running many instances of a game engine. *NVIDIA* appears to be supporting this view with their *NVIDIA Grid* aimed at supporting Cloud Gaming and the increased processing power through the addition of GPUs to this hardware.

Chen (2011) defined *real-timeliness* as the total of the following three components of *Network Delay or RTT:* time to deliver player commands to the server and return the matching screen to the client, *Processing Delay:* the time difference between a server receiving and responding to a client command, and *Play-out Delay:* the time difference from when a client receives information for a frame to when it is displayed on the screen.

Latency is one of the most talked about challenges. with the impacts of latency varying due to the differences in available bandwidth for each connected client (Shi, Hsu, et al. 2011). In this situation, being able to dynamically alter the BR via scalable image encoding is one approach that can help maintain responsiveness (Xu et al. 2013). One particular study compared *GamingAnywhere* with *OnLive* and the personal streaming service called *StreamMyGame*. The results clearly highlight how latency impacts the user's experience and therefore acceptance of a cloud gaming system. They also indicate how important latency improvements relate to the support of more users or streams per server and, therefore, making the service a feasible venture (see Table 3).

Table 3: Components of Latency

Challenge	*Factor*
Image Render Time	Server Game Engine Image Construction
Image Buffer Access/Copy	Server Image Data Made Available/Captured
Image Colour Conversion	Server Image Colour Space Transmission Suitable
Video/Audio Encoding	Server Codec Settings and Buffering
Video/Audio Compression	Server Compression Settings
Video/Audio Packetization	Server Transmission Protocol
Transmission Time	Network Distance Traversed
Bandwidth	Network Connection Capacity
Packet Loss/Competing Traffic	Network Quality, RTCP, QoS, QoE
Video/Audio Decompression	Client Decompression Settings
Video/Audio Decoding	Client Codec Decoding Settings and Buffering
Video/Audio Rendering	Client Renderer SDK/Library
Image Buffering	Client Buffering (Double/Triple/Quad)
Transmission to Display Time	Client Transmission to Display Device (HDMI/VGA/DVI)
Display Refresh Rate	Client 30/60/120Hz and Response Time of Monitor
User Perception Time	Gameplayer Time to Perceive Results
Input Device Response Time	Client Controller/Driver Input (Wired/Wireless)

LATENCY - SERVER-SIDE ASPECTS

On the server-side the *GamingAnywhere* performance experiments clearly identified that the overall latency was 25 percent of *OnLive's* service (see Figure 11). This was a large difference but given this experiment it forms both a benchmark for future experimentation as well as an example framework on which to firstly establish further data-collection.

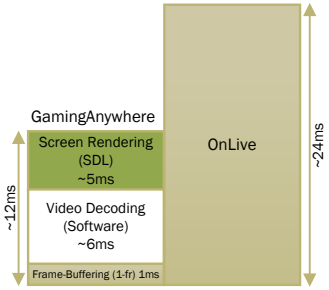

Figure 11: Huang's Server Comparison of *OnLive* to their *GamingAnywhere* Framework

It can be seen that there are several areas to improve outside of the game engine component itself. The actual retrieval of the image and conversion to a data format acceptable to the video encoding stage already costs 11 ms. The video encoding was the most substantial time consuming component at 14 ms but this was minimal given that the encoder acts more as a buffer rather than maximising *frame-by-frame* compression. The retrieval of the image from the game engine and conversion to the right colour space, costing 6 ms and 5 ms respectively, is an aspect that could be minimised if the system was built as part of the game engine itself.

LATENCY - CLIENT-SIDE ASPECTS

Although the *OnLive* client was far more competitive in performance it was still twice as slow as the *GamingAnywhere* client (see Figure 12). The system still requires 12 ms to go from the incoming data to a completed image displayed on the screen. The video decoding speed for the *GamingAnywhere* system, at just 6 ms, is considerably faster in producing a single frame. They used frame-buffering, for one frame, which consumed 1 ms while the image construction took almost as long as the decoding at 5 ms. Given this performance, the maximum FR would be around 80 F/Sec which is acceptable for all types of games.

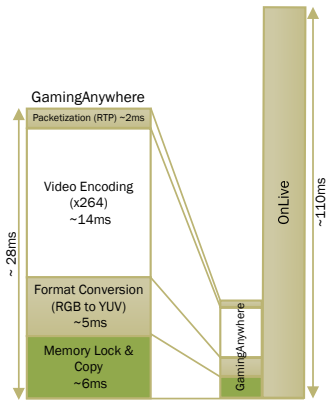

Figure 12: Huang's Client Comparison of *OnLive* to their *GamingAnywhere* Framework

The impact on latency can also be changed through how the display of the client system is set up. The use of a VGA or HDMI cable can provide different impacts to latency given the monitor (Pohl 2012). Monitors have varying response times, and when this is combined with

the screen refresh rate and the number of buffers used, it can have a dramatic impact to the overall latency measurements. Smart-TVs also have various image processing algorithms to enhance the picture that, in some cases, can be turned off via a *game mode* setting to decrease the TV response time. Disabling any *noise reduction* and *motion smoothing* options will also have the same impact.
Overall the framework offers an interesting structure and performance benchmark for initial developments. For future benchmarking, each of the components mentioned will need to be measured in creating a clear picture of where time is lost on the client.

MULTI-PLAYERS AND ADDITIONAL LATENCY

Multi-player games have an additional layer of latency to consider in coordinating interactions between users before rendering the results for streaming back to gameplayers. This can be addressed through geographically separating participants to reduce this overhead (Choy et al. 2012).

EFFICIENCIES OF MULTI-USER INTERACTIONS MANAGED CENTRALLY

By centralising all user interactions, in a MMO world for example, it is possible to complete multi-user interactions in complex environments that may lead to computational efficiencies (Baun et al. 2010). An example may be the pre-calculation of certain game environmental lighting that is not view-dependent.
Any data to do with the user, for example save games and character configurations, no longer needs to be saved to and transferred between devices (C. Huang et al. 2013b). This allows for players to instantly continue a game experience from any device supported without having to ensure that software or supporting data is available. The results also mean that games are often instantly playable from your last point and for single player experiences is identical to pausing and resuming a movie.

CONCLUSIONS

It is clear that the most important topics include how easily you can move a game onto a cloud platform along with the even more important topic of system responsiveness and user perceived quality of the service. The research gives insight to the design of the cloud game engine architecture along with the *GamingAnywhere* system providing a valuable benchmark for development evaluation.
The importance of platforms like *GamingAnywhere* and *LiveRender* enabling better research and benchmarks cannot be understated when large funded research projects, like *G@L*, have only an ageing collection of research papers left as evidence. The industry context presents quite closed approaches with companies like *OTOY*, *GameFly*, and *CiiNOW* developing private streaming systems. The contrasts of *Gaikai's* $380 million sale and OnLive's demise both now apart of *Sony* and the *PlayStation Now* service that is continuing to grow.
The bigger picture is how game engines of the future, that sit purely in the cloud, may leverage this position to address the challenges identified in more inventive ways.

REFERENCES

Bao, P and D Gourlay (2003). "Remote walkthrough over mobile networks using 3-D image warping and streaming". In: *Vision, Image Signal Process. IEE Proc.* Pp. 329–336.

Baratto, Ricardo A., Leonard N. Kim, and Jason Nieh (2005). "Thinc: A Virtual Display Architecture for Thin-Client Computing". In: *Proc. Twent. ACM Symp. Oper. Syst. Princ. - SOSP '05* 39.5, p. 277.

Baun, Christian, Marcel Kunze, Jens Nimis, and Stefan Tai (2010). "Informatik im Fokus". In: *Cloud Comput. - Cloud Gaming*. Springer Heidelberg. Chap. Ausgewählt, p. 71.

Beer, Guy de (2013). *Cloud Gaming TV*. Tech. rep. London: PlayCast.

Beigbeder, Tom, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool (2004). "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003". In: *ACM SIGCOMM* Workshops.October.

Bhaniramka, P., P.C.D. Robert, and S. Eilemann (2005). "OpenGL Multipipe SDK: A Toolkit for Scalable Parallel Rendering". In: *VIS 05. IEEE Vis. 2005.* November 2005, pp. 119–126.

Boukerche, Azzedine and Richard Werner Nelem Pazzi (2006). "Remote rendering and streaming of progressive panoramas for mobile devices". In: *Proc. 14th Annu. ACM Int. Conf. Multimed. - Multimed. '06*, p. 691.

Cai, Wei, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor C. M. Leung, and Cheng-Hsin Hsu (2016). "A Survey on Cloud Gaming: Future of Computer Games". In: *IEEE Access* 4, pp. 7605–7620.

Cedilnik, Andy, Berk Geveci, Kenneth Moreland, James Ahrens, and Jean Favre (2006). "Remote Large Data Visualization in the ParaView Framework". In: *Eurographics Symp. Parallel Graph. Vis.* March 2014, pp. 163–170.

Chang, Chun-fa and Shyh-haur Ger (2002). "Enhancing 3D Graphics on Mobile Devices by Image- Based Rendering". In: *PCM*, pp. 1105–1111.

Chen, Kuan-Ta, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei (2011). "Measuring the latency of cloud gaming systems". In: *Proc. 19th ACM Int. Conf. Multimed. - MM '11*, p. 1269.

Chen, Shenchang Eric (1995). "QuickTime VR - An Image-Based Approach to Virtual Environment Navigation". In: *SIGGRAPH Proc. 22nd Annu. Conf. Comput. Graph. Interact. Tech.* Pp. 29–38.

Chen, Shenchang Eric and Lance Williams (1993). "View interpolation for image synthesis". In: *Proc. 20th Annu. Conf. Comput. Graph. Interact. Tech. -SIGGRAPH*, pp. 279–288.

Cheng, Liang, Anusheel Bhushan, Renato Pajarola, and Magda El Zarki (2004). "Real-time 3d graphics streaming using mpeg-4". In: *Proc. IEEE/ACM* 1.August 2004, pp. 1–16.

Choy, Sharon, Bernard Wong, Gwendal Simon, and Catherine Rosenberg (2012). "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency". In: *2012 11th Annu. Work. Netw. Syst. Support Games*, pp. 1–6.

De Winter, D., P. Simoens, L. Deboosere, F. De Turck, J. Moreau, B. Dhoedt, and P. Demeester (2006). "A hybrid thin-client protocol for multimedia streaming and interactive gaming applications". In: *Proc. 2006 Int. Work. Netw. Oper. Syst. Support Digit. audio video - NOSSDAV '06* 5, p. 1.

Duguet, Florent and George Drettakis (2003). "Flexible point-based rendering on mobile devices." In: *IEEE Comput. Graph. Appl.* 24.4, pp. 57–63.

Eisert, P. and P. Fechteler (2008). "Low delay streaming of computer graphics". In: *Proc. - Int. Conf. Image Process. ICIP*, pp. 2704–2707.

Engel, Klaus, Rudiger Westermann, and Thomas Ertl (1999). "Isosurface Extraction Techniques for Web-based Volume Visualization". In: *IEEE Vis.* Vi, pp. 139–147.

Garland, Michael and Paul S Heckbert (1997). "Surface Simplification Using Quadric Error Metrics". In: *SIGGRAPH*, pp. 209–216.

Giesen, Fabian, Ruwen Schnabel, and Reinhard Klein (2008). "Augmented Compression for Server-Side Rendering". In: *Proc. Interna- tional Fall Work. Vision, Model. Vis. tion.*

Hemmati, Mahdi, Abbas Javadtalab, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, and Tarik Arici (2013). "Game as video: Bit rate reduction through adaptive object encoding". In: *ACM Work. Netw. Oper. Syst. Support Digit. Audio Video*, pp. 7–12.

Holthe, Ole-Ivar, Ola Mogstad, and Leif Arne Rønningen (2009). "Geelix LiveGames : Remote Playing of Video Games". In: *IEEE Consum. Commun. Netw. Conf.* Pp. 3–4.

Hong, Hua-Jun, Chih-Fan Hsu, Tsung-Han Tsai, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu (2015). "Enabling Adaptive Cloud Gaming in an Open-Source Cloud Gaming Platform". In: *IEEE Trans. Circuits Syst. Video Technol.* 25.12, pp. 2078–2091.

Huang, Chun-Ying, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen (2013b). "GamingAnywhere : An Open Cloud Gaming System". In: *Proc. ACM Multimed. Syst. 2013*.

Humphreys, Greg, Matthew Eldridge, Ian Buck, Gordan Stoll, Matthew Everett, and Pat Hanrahan (2001). "WireGL". In: *Proc. 28th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '01*, pp. 129–140.

Humphreys, Greg, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchner, and James T. Klosowski (2002). "Chromium: a stream-processing framework for interactive rendering on clusters". In: *ACM Trans. Graph.* 21.3, pp. 693–702.

Jarschel, Michael, Daniel Schlosser, Sven Scheuring, and Tobias Hoßfeld (2013). "Gaming in the clouds: QoE and the users' perspective". In: *Math. Comput. Model.* 57.11-12, pp. 2883–2894.

Jurgelionis, Audrius, Philipp Fechteler, Peter Eisert, Francesco Bellotti, David Haggai, Jukka-Pekka Laulajainen, Richard Carmichael, Vassilis Poulopoulos, Arto Laikari, Pekka Perälä, Alessandro De Gloria, and Christos Bouras (2009). "Platform for Distributed 3D Gaming". In: *Int. J. Comput. Games Technol.* 2009, pp. 1–15.

Khan, Javed I (1996). "Motion Vector Prediction in Interactive 3D Rendered Video Stream". In: *World Congr. Adv. IT Tools*, pp. 533–539.

Laikari, Arto, Philipp Fechteler, Benjamin Prestele, Peter Eisert, and Jukka-pekka Laulajainen (2010). "Accelerated Video Streaming For Gaming Architecture". In: *3DTV-Conference True Vis. - Capture, Transm. Disp. 3D Video.* IEEE.

Lamberti, Fabrizio and Andrea Sanna (2007). "A streaming-based solution for remote visualization of 3D graphics on mobile devices." In: *IEEE Trans. Vis. Comput. Graph.* 13.2, pp. 247–60.

Lee, Y.-T., K.-T. Chen, H.-I. Su, and C.-L. Lei. (2012). "Are all games equally cloud-gaming-friendly? An electromyographic approach". In: *2012 11th Annu. Work. Netw. Syst. Support Games*, pp. 1–6.

Liao, Xiaofei, Li Lin, Guang Tan, Hai Jin, Xiaobin Yang, Wei Zhang, and Bo Li (2016). "LiveRender: A Cloud Gaming System Based on Compressed Graphics Streaming". In: *IEEE/ACM Trans. Netw.* 24.4, pp. 2128–2139.

Liu, Yao, Shaoxuan Wang, and Sujit Dey (2014). "Content-aware modeling and enhancing user experience in cloud mobile rendering and streaming". In: *IEEE J. Emerg. Sel. Top. Circuits Syst.* 4.1, pp. 43–56.

Mangold, Stefan, Sunghyun Choi, Peter May, Ole Klein, Guido Hiertz, Lothar Stibor, and Q Bss (2002). "IEEE 802 . 11e Wireless LAN for Quality of Service". In: 11.

Mark, William R., Leonard McMillan, and Gary Bishop (1997). "Post-rendering 3D warping". In: *Symp. Interact. 3D Graph.* Pp. 7–16.

Marpe, Detlev, Heiko Schwarz, and Thomas Wiegand (2003). "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard". In: *IEEE Trans. Circuits Syst. Video Technol.* 13.7, pp. 620–636.

McMillan, Leonard Jr. (1997). "An Image-Based Approach to Three-Dimensional Computer Graphics". PhD thesis. University of North Carolina, pp. 1–191.

Meltzer, Stefan and Gerald Moser (2006). "MPEG-4 HE-AAc v2 - audio coding for todays media world". In: *EBU Tech. Rev.* 2.January, pp. 37–48.

Nave, Itay, David Haggai, Alex Shani, Arto Laikari, Peter Eisert, and Philipp Fechteler (2008). "Games@Large Graphics Streaming Architecture". In: *Consum. Electron.* ISCE, IEEE, pp. 1–4.

Noimark, Yuval and Daniel Cohen-or (2003). "Streaming Scenes to MPEG-4 Video- Enabled Devices". In: *Web Graph. IEEE Comput. Soc.* February, pp. 58–64.

Pohl, Daniel (2012). "Research and ideas regarding Cloud Gaming". In: Cloud Gaming USA.

Prohaska, S., A. Hutanu, R. Kahler, and H.-C. Hege (2004). "Interactive Exploration of Large Remote Micro-CT Scans". In: *VIS Proc. Conf. Vis.* Pp. 345–352.

Said, Amir (2004). *Introduction to Arithmetic Coding - Theory and Practice*. Tech. rep. Palo Alto: HP, pp. 1–62.

Schulzrinne, H., S. Casner, R. Frederick, and V. Jacobson (2003). *RTP: A Transport Protocol for Real-Time Applications*. Tech. rep. Network Working Group, pp. 1–89.

Schulzrinne, H., A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling (2013). *Real Time Streaming Protocol 2.0 (RTSP)*. Tech. rep. MMUSIC Working Group, pp. 1–302.

Sharabayko, M.P. and N.G. Markov (2016). "Contemporary video compression standards: H.265/HEVC, VP9, VP10, Daala". In: *2016 Int. Sib. Conf. Control Commun.* May, pp. 1–4.

Shi, Shu and Cheng-Hsin Hsu (2015). "A Survey of Interactive Remote Rendering Systems". In: *ACM Comput. Surv.* 47.4, 57:1–57:29.

Shi, Shu, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy H. Campbell (2011). "Using Graphics Rendering Contexts to Enhance the Real-Time Video Coding for Mobile Cloud Gaming". In: *Proc. ACM Multimed.* Pp. 103–112.

Shi, Shu, Mahsa Kamali, Klara Nahrstedt, John C. Hart, and Roy H. Campbell (2010). "A high-quality low-delay remote rendering system for 3D video". In: *Proc. Int. Conf. Multimed. - MM '10*, p. 601.

Stegmaier, Simon, Marcelo Magallón, and Thomas Ertl (2002). "A Generic Solution for Hardware-Accelerated Remote Visualization". In: *Eurographics - IEEE TCVG Symp. Vis.*

Wallach, Dan S., Sharma Kunapalli, and Michael F. Cohen (1994). "Accelerated MPEG compression of dynamic polygonal scenes". In: *Proc. 21st Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '94* 28, pp. 193–196.

Wang, Shaoxuan and Sujit Dey (2010). "Rendering adaptation to address communication and computation constraints in cloud mobile gaming". In: *GLOBECOM - IEEE Glob. Telecommun. Conf.*

– (2012). "Cloud mobile gaming: Modeling and measuring user experience in mobile wireless networks". In: *ACM SIGMOBILE Mob. Comput. Commun. Rev.* 16.1, pp. 10–24.

Xu, Yang, Chenguang Yu, Jingjiang Li, and Yong Liu (2013). "Video Telephony for End-Consumers: Measurement Study of Google+, iChat, and Skype". In: *IEEE/ACM Trans. Netw.* Pp. 1–1.

WEB REFERENCES

Ahiska, Dr Bartu (2012). *Approxy*. URL: `www.approxy.com`.

– (2015). *Numecent*. URL: `www.numecent.com`.

Beer, Guy de (2011). *Playcast*. URL: `www.playcast-media.com`.

Bellard, Fabrice (2000). *FFmpeg*. URL: `www.ffmpeg.org`.

Blade (2017). *Shadow*. URL: `shadow.tech`.

Commander, Darrell (2012). *VirtualGL*. URL: `www.virtualgl.org`.

Dillet, Romain (2017). *Shadow raises $57 million for its cloud computing service for gamers*. URL: `techcrunch.com/2017/06/14/shadow-raises-57-million-for-its-cloud-computing-service-for-gamers`.

DiStream (2004). *Triton (Game xStream)*. URL: `www.distream.com`.

Exent (2012). *FreeRide Games*. URL: `www.freeridegames.com`.

Eximion (2006). *Kalydo*. URL: `www.kalydo.com`.

Faria, Richard (2007). *StreamMyGame*. URL: `www.streammygame.com`.

Frozenbyte (2009). *Trine*. URL: `www.frozenbyte.com/trine`.

GDI Game Domain International (2006). *A World Of My Own (AWOMO)*. URL: `www.awomo.com`.

Group, Khronos (2015). *OpenMAX*. URL: `www.khronos.org/openmax`.

Hastings, Reed (1997). *NetFlix*. URL: `netflix.github.io`.

Hollister, Sean (2012). *OnLive lost: How the Paradise of streaming games was undone by one man's ego*. URL: `www.theverge.com/2012/8/28/3274739/onlive-report`.

Huang, Chun-Ying, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen (2013a). *GamingAnywhere*. URL: `gaminganywhere.org`.

Huang, Jen-Hsun (1993). *NVIDIA*. URL: `www.nvidia.com/object/cloud-gaming.html`.

Khronos Group (1997). *OpenGL - The Industry's Foundation for High Performance Graphics*. URL: `www.opengl.org`.

– (2017). *Vulkan*. URL: `www.khronos.org/vulkan`.

Levgoren, Zvi (1992). *Exent*. URL: `www.exent.com`.

Liao, Xiaofei, Li Lin, Guang Tan, Hai Jin, Xiaobin Yang, Wei Zhang, and Bo Li (2014). *LiveRenderProject*. URL: `github.com/llfjfz/LiveRender`.

Live Networks (1997). *Live555*. URL: `www.live555.com`.

Lowensohn, Josh (2015). *Sony buys streaming games service OnLive only to shut it down*. URL: `www.theverge.com/2015/4/2/8337955/sony-buys-onlive-only-to-shut-it-down`.

Microsoft (2017). *DirectX and UWP*. URL: `docs.microsoft.com/en-us/windows/uwp/gaming`.

N-Dream (2015). *AirConsole*. URL: `www.airconsole.com`.

Newell, Gabe (2003). *Steam Explained*. URL: `www.valvetime.net/threads/steam-explained.6373/%7B%5C#%7Dpost-132937`.

Open Connectivity Foundation (2008). *UPnP Industry Forum*. URL: `openconnectivity.org`.

Perlman, Steve (2011). *OnLive*. URL: `www.onlive.com`.

Rivest, Ron, Adi Shamir, and Leonard Adleman (2009). *RSA Laboratories*. URL: `apj.emc.com/emc-plus/rsa-labs`.

Simon, Ted (1997). *Red Orb Entertainment*. URL: `web.archive.org/web/20010118235200/http://www.redorb.com`.

Sony (2014). *PlayStation Now*. URL: `www.playstation.com/en-us/explore/playstationnow`.

The Steam Review (2007). *The Secret of AWOMO's Island*. URL: `steamreview.org/posts/awomo-rto`.

Urbach, Jules (2011). *OTOY*. URL: `www.otoy.com`.

Utomik (2015). URL: `www.utomik.com`.

Valve (2011). *Steamworks*. URL: `www.steampowered.com/steamworks/SteamworksBrochure2011.pdf`.

VideoLAN (2010). *VLC*. URL: `www.videolan.org`.

Walkden, Roger (2009). *'What Are You Waiting For?' GDI Reveals How Unique High Speed Video Game Download Technology AWOMO Works*. URL: `www.prweb.com/releases/awomo/games/prweb1906194.htm`.