A Cloud Gaming System Design Insights

Serban Ungureanu, Andrei Panu and Lenuta Alboaie
Faculty of Computer Science, Alexandru Ioan Cuza University, 16 General Berthelot, Iasi, Romania

Keywords: Cloud Gaming, Video Streaming, Remote Rendering.

Abstract: Cloud Gaming is a service that allows its users to run games without installing them locally. The game applications run remotely on a machine in the cloud, having the graphical output of the game compressed and sent back to the user's machine, where it is displayed. Within this paper, we explain how this technology works. We propose a cloud gaming platform solution, called InstantPlay, that we use to emphasize both advantages and disadvantages of this technology. Furthermore, we compare InstantPlay with a well known gaming platform, OnLive. Therefore, the scope of this paper is to explain the cloud gaming technology, present our implementation, and raise public awareness on this relatively new technology.

1 INTRODUCTION

The gaming industry has lately been on a constant rise. A very large number of new games, that offer experiences which are more and more closer to reality, are released each year. Game developers are in close relations with hardware producers, which release highest quality gaming gear. In order to fully experience latest games, users have to periodically purchase new hardware components which aren't always cheap. Furthermore, users have to support the costs of buying new games, costs which cannot be overseen. Considering all these costs, avid gamers are required to heavily invest in order to support their hobby.

Cloud gaming introduces a new alternative which does not require purchasing the latest hardware components. This technology aims to reduce the costs that must be supported by gamers. Cloud gaming, also known as Gaming as a Service or Gaming on Demand , is a service aimed at both casual and enthusiastic gamers and its purpose is to provide them games instantly, at the lowest possible cost. The most common types of cloud gaming are video streaming based and file streaming based. These two types of cloud gaming have almost identical intentions, which is running games instantly, but the advantages brought are different. In a cloud gaming system based on file streaming, the client application initially downloads the core part of the game, usually less than 5%. This is required for the game to run. While the user is playing the game, the rest of the game data is also downloaded on its machine. In a cloud gaming system based on video streaming, the game applications are stored and ran on the suppliers machines, in the cloud. On the user's machine, a small client application, also called thin client (Beal, 2015), is used to communicate with the supplier's servers. This thin client is designed to use a small set of the system's hardware resources, all the processing being done on the server. The thin client receives and displays a stream of video content which represents the graphical output of the game application that is ran on the server. Meanwhile, the client application registers and sends the user's commands to the server. The main difference between these two types of cloud gaming is represented by the machine which runs the game application. For file streaming based systems, the game application runs on the user's machine, as opposed to video streaming based system, where the game application runs on a server machine, "in the cloud". InstantPlay aims to implement a cloud gaming system based on video streaming, therefore all mentions of cloud gaming systems will refer to video streaming based cloud gaming.

The following section presents both advantages and disadvantages of using a cloud gaming service, introducing the reader into this domain. A couple of existing implementations will also be presented. The third section describes our own implementation of a platform of this kind, and a series of tests will be presented in the fourth section. In the last section we compare InstantPlay's performance with OnLive (OnLive, 2015), a commercial cloud gaming platform.

Ungureanu S., Panu A. and Alboaie L..
A Cloud Gaming System Design Insights.
DOI: 10.5220/0005515701440151
In *Proceedings of the 12th International Conference on e-Business* (ICE-B-2015), pages 144-151
ISBN: 978-989-758-113-7

2 CLOUD GAMING OVERVIEW

The cloud gaming technology offers a series of great advantages to it's users. Client-side hardware requirements reduction is considered one of the most important advantages present in cloud gaming systems. Since the graphics processing unit's (GPU) solely task is to display the video stream received from the cloud gaming servers, the user's system requirements are mainly related to the Internet connection. The central processing unit (CPU) and the RAM memory are also superficially used, as compared to running the game locally. Their main task is to run the thin client, which doesn't have excessive requirements. Taking these factors in consideration, the user doesn't need a high-performance personal computer for running games through a cloud gaming system. On the other side, in order to efficiently display the video stream without delays, a strong Internet connection is required.

While the games storing size grows larger and larger, cloud gaming systems solution is to store the games on the servers, relieving its users from the necessity of storing the game applications locally. Additionally, using cloud gaming systems, users can instantly run game applications, without losing time with lengthy installations or updates.

Another important goal achieved by cloud gaming systems is that using this kind of platforms, games become cross-platform. It is a difficult task for game developers to implement their product for every device and operating system. Game developers often develop platform-specific products, usually because of financial reasons, many gamers being deprived of these products. This issue is handled by cloud gaming systems, and both users and game developers benefit from this situation.

From a financial point of view, gamers may not be required to purchase the game entirely. Most cloud gaming platforms allow users to rent a game for a determined period of time, also providing the option to purchase the game entirely.

Running game applications in the cloud can also handle some piracy issues. If a game were to be available only through a cloud gaming system, illegal copying of that game would be theoretically impossible. Furthermore, having the option of trying the game before purchasing it may also reduce piracy, since users can try the game before purchasing. This way they can better decide if the game is worth buying. On the other hand, this strategy overlooks users with a slow or without an Internet connection.

Although cloud gaming services greatly benefit involved parties, a series of disadvantages can be identified. Instead of investing in buying high-performance computer hardware, gamers have to invest in a strong, stable Internet connection, in order to fully benefit from the advantages brought by cloud gaming. If a strong Internet connection is not present, users will notice a quality reduction of the service, usually in the form of high latency. Latency represents the time required for a network package to travel from source to destination. Latency is very important in these type of services, meaning that high latencies can lead to desynchronization between user input and server output. In this situation, the execution of user's commands on the server machine is delayed, which can be a great disadvantage for users. InstantPlay implementation aims to generate a minimum latency.

On the other hand, for cloud gaming suppliers, offering a high quality service can also be a difficult task. In order to satisfy their customers, cloud gaming suppliers have to design high quality, stable and scalable systems, in addition to buying high-end hardware components, factors which make implementing cloud gaming systems very challenging.

OnLive, launched in March 2010, was the first commercial implementation of cloud gaming services. Although OnLive was a huge success in the United States, there was no public document or available publication to explain their implementation. Another resonant success was Gaikai, which together with Electronic Arts, released its beta version in February 2011 (Brown, 2011). A year later, Sony purchased Gaikai (Newman, 2012) and this technology was integrated and released as Playstation Now (Brown, 2011), a service for Sony PlayStation consoles. Following OnLive example, neither Gaikai nor Sony ever released detailed information regarding their implementation of cloud gaming services. Inevitably, in April 2013, GamingAnywhere, the first open source cloud gaming platform, was released (Huang et al., 2013). GamingAnywhere's website contains extensive documentation on their implementation, as well as different documents containing performance and quality of service measurements. As of April 2015, important parts of OnLive technology were bought by Sony in order to also be integrated in Sony's Playstation Now cloud gaming platform (Hachman, 2015), further strengthening Sony's position in the cloud gaming scene.

3 INSTANTPLAY ARCHITECTURE AND DESIGN

InstantPlay represents our approach on designing and implementing a cloud gaming platform. As of this moment, there are only a few existing documents

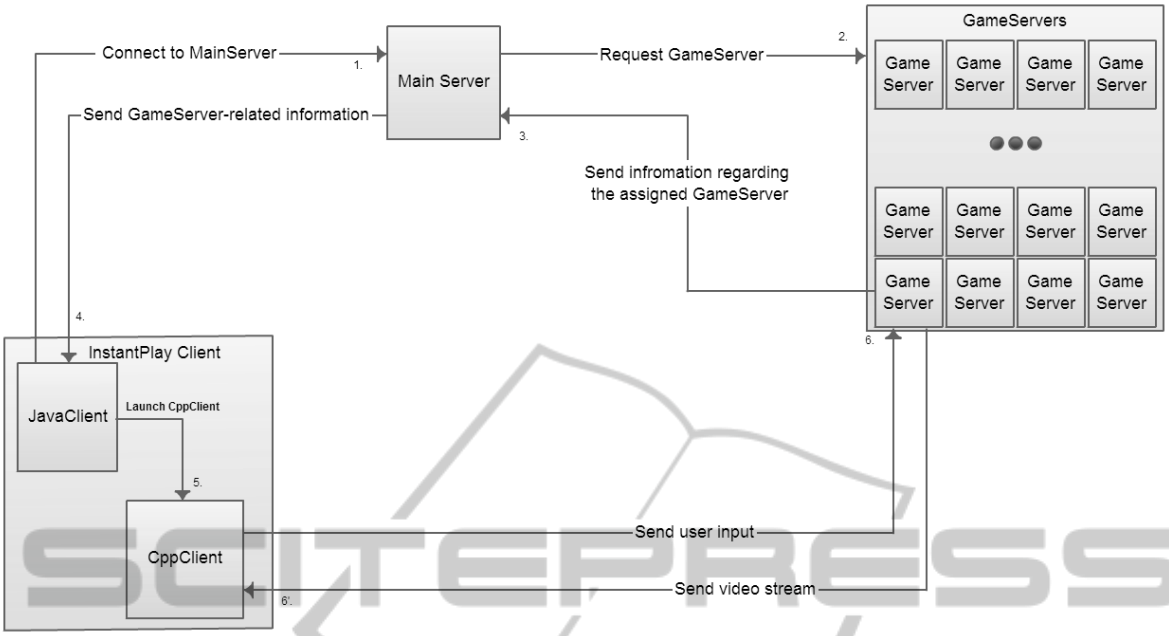

Figure 1: InstantPlay Architecture.

publicly available that present technical details regarding cloud gaming systems. When we designed InstantPlay, we aimed to provide a clear and concise solution, while offering all of the advantages of cloud gaming systems. InstantPlay's architecture is composed of the following modules: InstantPlay Client, MainServer and GameServers. The arrows between the modules, depicted in Figure 1, represent the communication between them, and the messages are chronologically ordered.

MainServer is the management module, built on top of an TCP/IP concurrent server application. This server awaits connections from clients and then manages the communication between a client and the corresponding game server. GameServers module manages the server machines which run the game applications. An application, used to communicate with the clients, is installed on these type of machines. InstantPlay client is the component located on the user's computer. This module is composed of two applications, one used for communication with the MainServer module, and the other one used for communication with the corresponding game server. Currently, InstantPlay Client module has a Windows-specific implementation.

The following subsections contain detailed explanation of how each module works.

3.1 MainServer Module

The MainServer module is comprised of a C++ application which implements a concurrent TCP/IP server. This application listens for client connections, and accepts clients, creating a new thread for each one. This way the application can communicate with multiple clients simultaneously. The server's configuration parameters are read from a configuration file.

The MainServer module plays an intermediate agent role, administrating connections between clients and the assigned game servers. Initially, when a client is connected, the MainServer module sends a request to the GameServers in order to allocate a game server for the connected client. Afterwards, the module simultaneously communicates with both the client and the assigned game server so that vital information, like IP addresses and respective ports at which certain services will run, is interchanged. The MainServer module assures that this type of information properly reaches its destination.

3.2 InstantPlay Client Module

InstantPlay Client represents the component that is installed on the user's computer. This component is composed of two applications: JavaClient and CppClient.

JavaClient is an application which contains graphical user interface (GUI) elements. This application is launched by the user, and its purpose is to initiate a connection between the client and the MainServer module. CppClient is the application that handles the communication between the client and the assigned

game server and is the core component of the Client module. CppClient is a Windows application which maintains and manages communication with the assigned game server. The main tasks of this application is displaying the video stream, recording and sending user's input to the respective game server (Russinovich et al., 2012). In order to determine which commands the user has entered, CppClient uses two hook procedures, implemented inside two Dynamic Link Libraries (DLL), that are dynamically loaded at runtime. A hook procedure is a function that intercepts a certain type of operating system event. CppClient installs two global hook procedures, one for intercepting mouse events and the other one for intercepting keyboard events. CppClient intercepts these types of events, encapsulates them in packets and sends them to the game server. The events described above may also be interpreted by other applications installed on the user's machine. For displaying the video stream received from the assigned game server, InstanPlay Client uses MPlayer (MPlayer, 2015), an open-source application.

When the user launches JavaClient, the application will establish a connection with the MainServer. This connection is used to receive information regarding the assigned game server. After this information is received, JavaClient launches CppClient application, passing it information regarding the assigned game server. CppClient saves this information in registries, in order to be easily accessed by the hook procedures. After initializing registry values, CppClient launches MPlayer with specific arguments for displaying the video stream received from the game server. The two DLLs mentioned above are loaded into memory. The application uses different WinAPI functions to be able to install the hook procedures. After the hook procedures are installed, when mouse and keyboard events are fired, the operating system will invoke the implemented hook procedures. These procedures will serialize and encapsulate event-specific information into network packets. These packets are then sent to the assigned game server, where they will be executed.

From the user's point of view, the client application displays a MPlayer window, which plays the video stream received from the assigned game server. Meanwhile, each user command is serialized and sent to the game server.

3.3 GameServers Module

The GameServers module consists of an application which is installed on the game server machines (either virtual or physical), machines on which game applications are ran. The main tasks of this application is assigning a game server instance for each client, recording, compressing (Salomon, 2004) and streaming the graphical output of the game application to the respective client (Thornhill et al., 2002). The GameServers application also receives and executes the user input (Microsoft, 2015), in the game context.

A game server instance is launched when a specific request is received from the MainServer. This request contains information regarding the IP address and port to be used for receiving user input information. Firstly, the GameServer instance launches the requested game and then runs the FFmpeg application (FFmpeg, 2015), which will handle media streaming. Although obtaining a minimum latency is desired, the quality of the video stream must not be overlooked. FFmpeg reaches a compromise between the quality of the video stream and compression speed, using MPEG-4 codec for compressing captured images from the game server machine. With the purpose of obtaining maximum speed, the packets are sent to the client using the User Datagram Protocol(UDP). On the grounds that additional verifications are not required within this context, the UDP protocol is best suited for this situation.

The Game server application's next action is to setup a server for receiving UDP messages, containing user input information, from the client. Considering that the packets containing user input information have to reach destination as fast as possible, the UDP protocol is yet again appropriate. The application creates an UDP socket for continuously receiving packets from the client. A packet of this kind is composed of an initial byte for specifying the type of message: keyboard message, mouse message or finishing message. For finishing messages, the application closes all open connections and ends the session. For other types of messages, the application will then receive additional information specific to each type of event:

- for keyboard events, the application receives 2 bytes representing the key and if the key was pressed or released;
- for mouse events, the application receives 9 additional bytes. The first byte represents the type of mouse event, for example mouse movement or mouse button click. The latter 8 bytes represent the x and y coordinates used to determine the mouse position where the event was fired.

Having this information, the game server application can simulate, within the context of the running game, the described command. Windows API's SendInput function is used to execute mouse and keyboard commands.

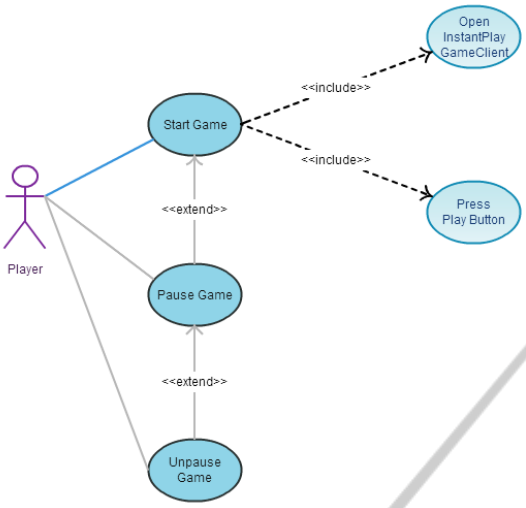

Figure 2: Use-case diagram.

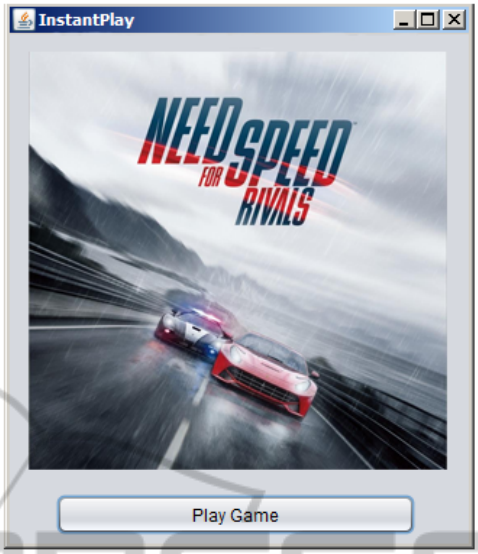

Figure 3: InstantPlay Client's main window.

4 USE-CASE SCENARIO

This section presents an InstantPlay use-case scenario. For this testing scenario, two computers were used, one used for both MainServer and GameServers module and one for the client module. The InstantPlay platform does not require both server modules to be installed on the same machine, these modules could be installed as well separately, on cloud virtual machines for example. Need for Speed Rivals was the game used for testing purposes.

Figure 3 shows the main window that is displayed when the client application is launched.

When the user presses the "Play Game" button, a MPlayer process will be loaded, and it will play the video stream received from the assigned game server. On the server side, the game server application responsible for running the game application launches the requested game and streams the graphical output of it to the client.

From this moment forward, the user can interact with the game ran on the game server machine, as if it would locally, as seen in Figure 4. The user's commands are intercepted and sent to the assigned game server, where they are executed.

In order to determine the performances of our implementation, we developed a testing scenario for our cloud gaming platform. The testing scenario consisted of running Need for Speed Rivals on through our cloud gaming platform examining environment information. The main parameters for determining the performance of our platform are network traffic and system resources usage. With regard to pointing out the advantages brought by cloud gaming platforms, the client module was installed on a virtual machine with the following system configuration: Intel 2.20 Ghz processor, 1 GB RAM, 21MB video memory, Windows 7 OS. This virtual machine does not meet the system requirements for running Need for Speed Rivals: CPU: Intel 2.4 GHz Core 2 Duo or AMD 2.6 GHz Athlon X2; video card: AMD Radeon HD 3870 512Mb or better, NVIDIA GeForce 8800 GT 512 Mb or better, Intel HD 4000 integrated 512Mb or better; RAM: 4GB.

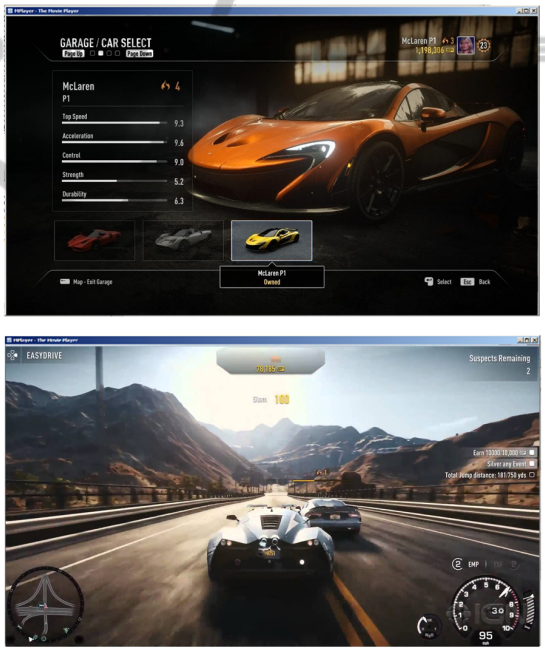

Figure 4: Need for Speed Rivals game run on InstantPlay platform.

The game can be ran on the machine described above, only using a cloud gaming system. For testing purposes, we recorded Internet traffic when running the game with InstantPlay for two minutes. Considering the fact that both the game server and the client were connected to the same local network, which offered a significantly higher Internet speed than in a real scenario, we used an auxiliary application, NetLimiter (NetLimiter, 2015), in order to limit the Internet speed at which data was transferred.

While running the game through InstantPlay platform, we also monitored the system resources usage for the client machine. As we can clearly see in Figure 5, approximately 47% of the CPU and 800MB of RAM are used. It is worth mentioning that a considerable part of the resources consumption was determined by the applications used for monitoring platform performance (WireShark (Wireshark, 2015), NetLimiter, NetBeans etc.). By analyzing Figure 5, we determine that approximately 25 MB of RAM were used by InstantPlay adjacent applications (ClientProject.exe and mplayer.exe processes).

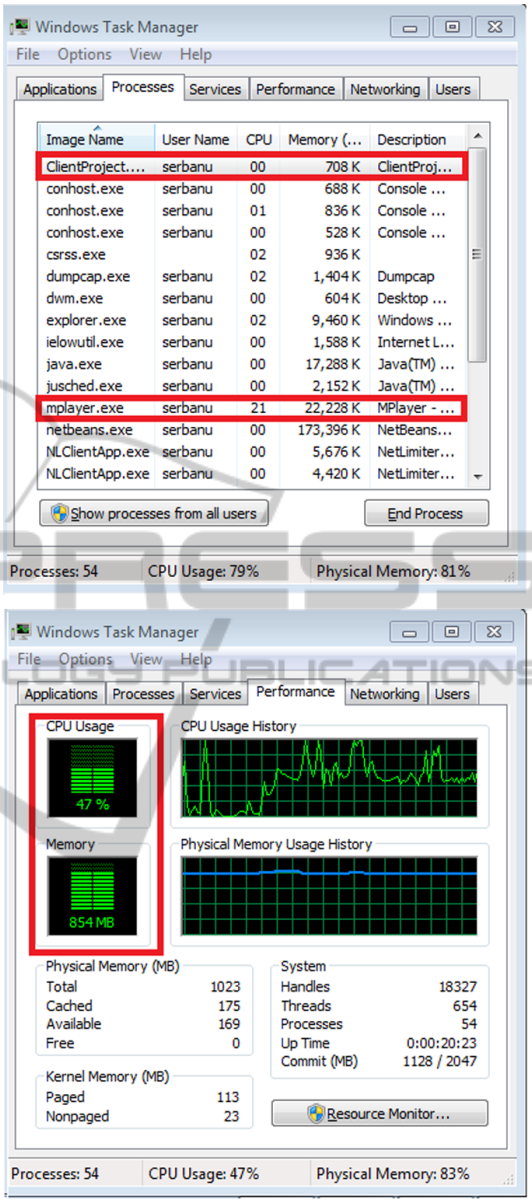

Figure 5: Used resources on the client machine during the game execution.

5 COMPARISON WITH ONLIVE

The next step in determining the performance of our implementation would be to test it in comparison with an existing, well known platform. For this test we chose OnLive, even though it is merely impossible to come to a realistic conclusion regarding the platform performance. It is highly challenging to simulate the amount of users that OnLive has at a point in time. In this regard, we captured and recorded internet traffic of the client machine for when running Need for Speed Rivals on both InstantPlay and OnLive, using a network traffic limiter for InstantPlay scenario. Afterwards, we used an application named SteelCentral Packet Analyzer (Riverbed, 2015) to generate an extensive report containing detailed statistics on the state of the network.

The network traffic summary is described in Figure 6. We chose a set of parameters that we can determine through tests. We can assert that the size of the data transferred for InstantPlay is significantly lower than for OnLive. This main reason behind this result is that OnLive provides higher quality video stream and, currently, InstantPlay does not offer sound support. Another aspect worth mentioning is that, as we can determine from the tables above, the number of packets transferred is greater for InstantPlay, meaning that InstantPlay uses packets of smaller size, compared with OnLive packets.

Using SteelCentral Packet Analyzer we could obtain information regarding the network status from different perspectives. Figure 7 contains detailed information regarding the bandwidth in bytes per second used by the TCP/IP and UDP protocols. Using this chart, we can assert that both platforms intensely use the UDP protocol, and for a good reason. As explained in the third section, the UDP protocol is the best choice for transmitting most of the data in cloud gaming platforms. The video stream and user commands need to reach destination as soon as possible. Even though some network packets may be lost in the

InstantPlay data

Statistic Name	Value
Total Number of Bits	613,861,336
Total Number of Bytes	76,732,667
Total Number of Packets	225,331
Number IP Bytes	76,620,473
Number TCP Bytes	271,852
Number UDP Bytes	76,348,321

OnLive data

Statistic Name	Value
Total Number of Bits	900,464,192
Total Number of Bytes	112,558,024
Total Number of Packets	125,932
Number IP Bytes	112,540,733
Number TCP Bytes	749,119
Number UDP Bytes	111,791,314

Figure 6: InstantPlay vs. OnLive: network traffic statistics.

Figure 7: TCP/UDP bits for InstantPlay (top) and OnLive (bottom).

process, this will not have a great impact on the user's experience.

6 CONCLUSIONS

Cloud gaming is a strong alternative to traditional gaming, bringing relevant advantages to the table, like device and operating system independence, reduction of hardware costs, install and update free games, uniform access to games different by design. Gaming as a Service can be seen as an intermediate between gamers and game developers, facilitating user access to gaming software products. The only mandatory resource for these types of services is an Internet connection, which has to be strong and stable in order to offer a pleasant experience.

In this paper we take a shot at explaining cloud gaming technologies, and also to try to raise awareness between developers and help the growth of this domain. Our goal is to provide an alternative to GamingAnywhere, which is one of the few, if not the only, provider of technical information regarding cloud gaming services.

The previously presented and analyzed platform InstantPlay is a cloud gaming platform offering basic functionality. Sound support and optimizations in regards to video capture are some areas that need further development. Currently, the graphical output of the game application is initially displayed on the standard output of the GameServer, then to be captured by a DirectShow filter. Implementing a video card driver, driver that sends the image bitmaps directly through the network instead of displaying them, could bring great improvement to this system.

REFERENCES

Beal, V. (2015). Thin Client Definition. http://www.webopedia.com/TERM/T/thin_client.html.

Brown, R. (2011). Gaikai Cloud-Gaming Service Goes Live. http://www.cnet.com/news/gaikai-cloud-gaming-service-goes-live/.

FFmpeg (2015). FFmpeg Multumedia Framework. http://www.ffmpeg.org.

Hachman, M. (2015). Cloud gaming pioneer OnLive sold to Sony, will cease operations April 30. http://www.pcworld.com/article/2905407/.

Huang, C.-Y., Hsu, C.-H., Chang, Y.-C., and Chen, K.-T. (2013). GamingAnywhere: An Open Cloud Gaming System. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13)*, pages 36–47. ACM.

Microsoft (2015). Microsoft Developer Network. http://msdn.microsoft.com.

MPlayer (2015). MPlayer Movie Player. http://www.mplayerhq.hu.

NetLimiter (2015). NetLimiter Internet Traffic Control and Monitoring Tool. http://www.netlimiter.com.

Newman, J. (2012). Sony Acquires Gaikai: 4 Possible Outcomes. http://www.pcworld.com/article/258664/sony_acquires_gaikai_4_possible_outcomes.html.

OnLive (2015). OnLive Gaming Platform. http://www.onlive.com.

Riverbed (2015). SteelCentral Packet Analyzer. http://www.riverbed.com/products/performance-management-control/network-performance-management/High-Speed-Packet-Analysis.html.

Russinovich, M., Solomon, D., and Ionescu, A. (2012). *Windows Internal, 6th edition, part 1*. Microsoft Press.

Salomon, D. (2004). *Data Compression. The Complete Reference, 3rd edition*. Springer.

Thornhill, S., Asensio, M., and Young, C. (2002). *Data Compression. The Complete Reference, 3rd edition*. The JISC Click and Go Video Project.

Wireshark (2015). Wireshark Network Protocol Analyzer. http://www.wireshark.org.