Testing cross-platform streaming of video games over wired and wireless LANs

A. Jurgelionis, F. Bellotti, A. De Gloria
Department of Biophysical and Electronic Engineering
University of Genoa
Genoa, Italy
{jurge, franz, adg}@elios.unige.it

J.-P. Laulajainen
Converging Networks Laboratory
VTT Technical Research Centre of Finland
Oulu, Finland
jukka-pekka.laulajainen@vtt.fi

P. Fechteler, P. Eisert
Computer Vision & Graphics, Image Processing Department
Heinrich-Hertz-Institute,
Fraunhofer-Institute for Telecommunications
Berlin, Germany
{philipp.fechteler, eisert}@hhi.fraunhofer.de

H. David
R&D Department
Exent Technologies Ltd.
Petach Tikva, Israel
hdavid@exent.com

***Abstract*—In this paper we present a new cross-platform approach for video game delivery in wired and wireless local networks. The developed 3D streaming and video streaming approaches enable users to access video games on set top boxes and handheld devices that natively are not capable to run PC games. During the development of the distributed gaming system we have faced a number of challenges and problems posed by the hardware and network limitations. In order to solve these problems we have developed a multilevel testing methodology which is based on user assessment and technical measures for the system under development. In this paper we focus on the technical measures and instrumentation that we use for the system's performance measurement and testing. The benefits of our testing methodology are demonstrated through examples from the development and testing work.**

Keywords-video and 3D streaming; interactive entertainment; testing and evaluation; Gaming on Demand; distributed and cloud computing

I. INTRODUCTION

Video games are typically executed on Windows platforms with DirectX API and require high performance CPUs and graphics hardware. For pervasive gaming in various environments like at home, hotels, or internet cafes, it is beneficial to run games also on mobile devices and modest performance consumer electronics (CE) devices in order to avoid the necessity of placing a noisy workstation in the living room or costly computers/consoles in each room of a hotel.

One of the latest advancements in gaming technology that enables such a pervasive gaming are cloud-based game systems, also called Gaming on Demand (GoD). They are networked media platforms that offer web-based services allowing game play on smaller end-devices like low-cost PCs or set top boxes without requiring the games to be installed locally. Most of these systems are based on video streaming. A server executes the game, the graphical output is captured, and then transmitted as video to the client. For an interactive experience, such a system requires a low end-to-end delay, which will be achieved through high compression efficiency and low encoding complexity [1].

This paper presents a new distributed system for video game streaming called Games@Large (G@L) [1, 2]. Novel system architecture and protocols, based on cloud computing paradigm, are used to transfer the game graphics in real-time across the network to multiple concurrently connected low-cost end devices. G@L is implementing an innovative architecture, transparent to legacy game code, which supports unmodified commercial video game titles.

Currently used wireless networks and CE devices face difficulties in terms of achieving low end-to-end delay of high-quality graphical output on the remote end device for highly interactive applications. This requires some additional efforts from the testing and development in order to identify and overcome these issues. We have developed a multilevel testing methodology, based on user tests and performance measurements [3], which we have applied for G@L testing. In this paper we describe the main challenges and testing tools used with regard to the performance of the newly developed 3D and video streaming protocols and networking in the G@L system.

II. RELATED WORK

There are a number of commercial Gaming on Demand systems, overviewed in [1], that have been presented to the market. More recently, there have been some new announcements about the upcoming systems such as Playcast Media Systems and Gaikai's Streaming Worlds(tm)

technology. However, there is very little detailed technical information publicly available about the commercial systems.

There have been many publications on streaming graphical content in the academic field that are discussing different techniques to achieve minimal delay, using high-performance H.264 video encoding, necessary for interactive applications [4], or to reduce the computational complexity of the MPEG-4 encoding process [5, 6]. Video streaming performance and QoS issues in wireless networks have been discussed in numerous papers such as [7, 8]. The effects of networking, frame rate and resolution on online games have been studied in [9, 10, 11, 12].

This paper is inspired by the mentioned scientific publications, and by the fact that there is very little detailed technical information publicly available about the emerging commercial systems.

III. G@L System Implementation and Challenges

Streaming PC content in real-time is a well known paradigm, e.g. 2D streaming of an X server in UNIX-based systems. Extensions for streaming 3D graphics also exist, for example the OpenGL Stream Codec (GLS) that allows the local rendering of OpenGL commands. These systems are usually based on error-free TCP/IP protocols, with best effort transmission without any delay constraints. However, they do not support streaming of modern video games that use DirectX and are not optimized for streaming media with high interactivity requirements.

The G@L framework enables modern video game streaming from a local server to remote end devices in local area networks (LANs). The system and streaming protocols are developed and adapted for highly interactive video games.

In the following sections we present the 3D streaming and video streaming protocols, as well as the networking architecture used for video game streaming.

A. 3D Streaming

The majority of games run on Windows using DirectX as a lower graphic layer that provides a graphic abstraction layer above the graphics hardware. In Games@Large 3D streaming those commands are intercepted and their execution transferred to the client device.

The architecture of the 3D streaming implements a pipeline that consists of several layers on both server and client sides as depicted in Figure 1. The layers on the server side are executed in the game context. The first layer intercepts the 3D commands that are issued by the game using a proxy implementation of the graphic library that is being used by the game for rendering graphics. The next two layers combine a logical compression layer that compresses special data like vertex buffers and textures. Vertex buffers are compressed using a technique to predict the coordinates and colors of the next frame based on their current values and the current values of their neighbor vertices. The knowledge about the content of the data is being used to compress the data efficiently and by that, reducing the bandwidth used by the 3D streaming protocol.

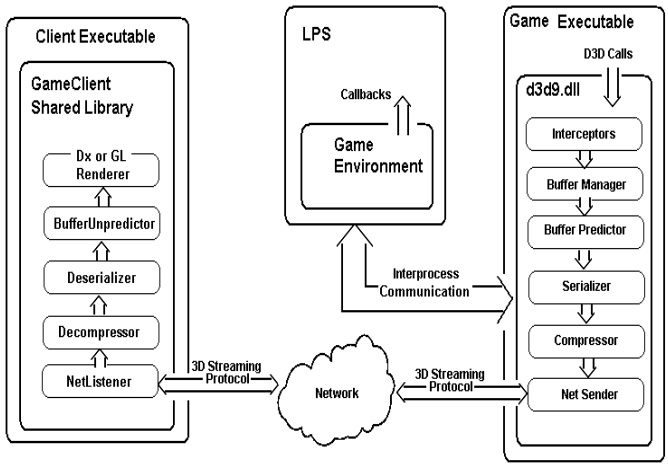

Figure 1. 3D streaming block diagram

The next layer serializes various structures describing commands and the graphics state to a buffer which is compressed by a well known lossless compression algorithm (e.g. zlib) in the compressor layer. The compressed buffer is sent via the network layer to the client device.

On the client side, there is a similar pipeline that reverses the operations of the server and sends the 3D commands to a rendering layer that uses the graphics library on the end device (DirectX or OpenGL) to render the commands on the local display.

Using the Games@Large 3D streaming enables the end device to use a high-resolution display without reducing the quality of the graphics and without the complexity of encoding big frames. Also, the 3D streaming protocol enables to use a different graphic library on the client side such as OpenGL or a version of DirectX that is different from the one being used by the game on the server.

B. Video Streaming

The alternative approach is Video Streaming which is used if 3D streaming is not applicable. These cases are: a) the client lacks the needed graphics hardware, b) the games 3D commands are not sufficiently manageable and c) the clients screen is too small so that video streaming results in better efficiency.

Here, the rendering takes place on the server and the resulting frame buffers are captured and video encoded using the current state of the art video standard H.264/AVC. Since video encoding is performed in parallel to the game execution and is itself quite demanding in terms of computational complexity, several optimizations have been developed which are mainly based on exploitation of the available render context information of DirectX / OpenGL. In H.264/AVC, each input frame is subdivided into 16x16 pixel macroblocks which are the basic entities of encoding. To support more detail-richness, each macroblock is further partitioned into smaller partitions (16x16, 16x8 … 8x4, 4x4). For each partition in P frames a motion vector is calculated which denotes its location in the previous frame. Both, macroblock partitioning and motion vector calculation, of generic H.264/AVC encoders are

computational very demanding, because they are implemented as trial-and-error searches. An acceleration based on render context information has been developed which allows the direct calculation of motion vectors, similar to [5]. In the same way we exploit a commonly used game rendering technique called skybox, besides the direct macroblock partitioning based on the depth map [13].

In order to minimize the efforts needed to integrate new client end devices the streamed video data is fully compliant to H.264/AVC. On the other side, it is beneficial to consider as much the end device characteristics as possible. This is realized by storing profiles of end devices in a database on the server. According to these profiles the encoding is parameterized, so that the resulting video stream exactly matches the client's capabilities. E.g. CABAC, which results in approximately 10% lower bitrates will be used only for end devices which possess enough computational power to handle the increased complexity. Another example is the end device's screen size: The viewport related commands of the games render engine are intercepted and altered so that the rendered scene always matches exactly the device's screen size. In this way, there is no additional scaling needed which in turn would introduce scaling artefacts, additional delays and computational load. To guaranty robust playback even on end devices with very weak processors, like Personal Digital Assistants (PDAs) etc, the H.264/AVC profile and level can be specified to which the resulting video stream will be compliant.

Since the streaming of highly interactive content, like 3D computer games, is not the common use case for generally available media players, like the VLC player, a dedicated client software has been implemented. The current implementation is already optimized for low delay playback through minimized buffering. Further improvements are planned, e.g. to continuously and systematically intersperse I-macroblocks into the video stream so that a continuous update of the frame takes place. This will reduce the number of peaks in the bit rate caused by full I frames at the cost of a slightly increased overall bit rate.

C. *Quality of Service enabled LAN*

The G@L system requires low-latency and high throughput communication between the server and client devices. Network delay should be minimized to maximize the interactivity and controllability and the network bandwidth should be high enough for good quality gaming.

Currently used LAN technologies such as IEEE 802.3u/ab Ethernet or IEEE 802.11g/n Wireless Local Area Network (WLAN) technologies are able to satisfy the requirements of G@L in good conditions, but problems may occur if the network is shared by several users. In case of congestion, the network impairments like increased delay, jitter, and packet losses distribute evenly on all competing traffic. While this can be accepted by traditional Internet applications such as file downloading or web surfing, it is not acceptable for G@L gaming. Congestion caused latency, limited transmission capacity, and the effects caused by them cannot be fully concealed from the player and the user perceived quality of the game drops. But the quality can be enhanced using Quality of Service (QoS) technologies to give higher priority to game traffic in the network bottlenecks.

In the initial G@L system [2] it is assumed that all the related components are situated in the same LAN,so QoS for the Internet access does not have to be considered. In home environments the possible bottleneck of the network can be further limited to the wireless LAN since the capacity of wireless LAN technologies is generally lower than wired LAN technologies thus congestion is more likely to occur there. In large deployments (e.g. hotels) QoS mechanisms such as IEEE 802.1p should be used also in the wired part of the LAN to give higher priority to game traffic.

Two levels have to be considered for providing QoS in a network. There has to be a way to implement QoS on the medium access (MAC) layer and there has to be a solution for managing the QoS and mapping the application layer requirements to the MAC layer QoS.

The MAC operation of IEEE 802.11 [14] technologies is based on CSCMA/CA which does not support any kind of QoS. This weakness is addressed in IEEE 802.11e [15] standard that defines new MAC layer operations to provide different treatment for different traffic classes. Some parts of the standard are included in commercial Wi-Fi Multimedia (WMM) specification [16] which is widely supported by the WLAN devices on the market. WMM is a powerful means to support real-time applications such as gaming in wireless LANs [17] so we rely on it in the initial G@L system.

QoS configuration of a network can be either statically configured or managed in a dynamic manner. A static configuration may be applicable to enterprise networks where there are professional administrators who can setup the QoS configuration to support games. In homes, the networks are operated by people not trained with network technologies so the configuration process should be automated. For this we use UPnP QoS [18] middleware which provides the applications a possibility for provisioning priority based QoS. When a game session is started, the game application communicates with UPnP QoS system, which takes care of the QoS configuration to the needed network devices.

IV. INSTRUMENTATION FOR TESTING

The design and development of the G@L system is performed using an iterative approach where the technical development and testing are run in parallel. To facilitate deep analysis on the system's performance, the initial G@L platform is equipped with detailed logging of system behaviour.

The game server logs information on all game sessions it serves. The information includes statistics on CPU and memory usage, details on 3D and video streaming, as well as network performance. The game client logs similarly statistics on CPU and memory usage, details on received frames and their processing and rendering times, and the amount of frames presented to the user per second.

The measured parameters affect the delivered quality of a game on the end device. Therefore it is important to understand

the dependencies between them in order to achieve a good system's performance.

The parameters that are monitored during the operation of the system in 3D streaming mode are described below. During a game session, every layer in the pipeline logs the required information such as timestamp, size of data used, frame rate etc.

D3D commands rate; In G@L implementation the server does not send all the 3D commands to the client side but caches some of them, and eliminates the commands that query the state of a 3D object from being transferred to the client by tracking its status on the server side. That's why the number of Direct3D commands generated by the game is different from the number of commands executed on the client. The number of 3D commands, which are actually transmitted, is decreased for most of the games.

Data transfer rate is the amount of raw data that are needed to be transferred from the server to the client and also the amount of compressed data that is actually been transferred. It is important to achieve a higher compression ratio in order to reduce the used bandwidth.

Frames per second (FPS) and frame delay; FPS is the number of frames per second that have been rendered by the client. High FPS results with smooth movements in the 3D scene. In the Games@Large implementation, the server and the client are using acknowledgment mechanism to control the frame rate, so the server produces and sends frames in a rate that the client can process.

Delay is defined as the time between a player's action (e.g. move, shoot) and the time the actual resulting game output on the screen. The total delay is compound of the processing time on the server side, the network native delay and the processing on the client side in both uplink and downlink. Therefore, a user's action on the current frame will be effected on the end-device output only on the successive frame. The gaming experience is directly influenced by the delay.

Measuring the delay manually adds inaccuracy of about a millisecond because it requires synchronization of the client and the server computer clocks. During testing it has been observed that it leads to approximately the same results (within a magnitude of a few milliseconds) as the theoretical analysis approach. Thus the average frame delay is computed as the average of frame delay values for a game session. Considering the synchronization between the server and client and since the user action is issued somewhere between the current frame and the next frame, the frame delay, *d*, is bounded with

$$\frac{1000}{FPS+1} < d \leq \frac{1000}{FPS} \qquad (1)$$

Logical compression/decompression time per frame; The logical compression layer works in conjunction with the lossless compression layer in a sense that the former emits buffers that the latter compresses more efficiently. Due to this, when analyzing the performance of each compression layer, the others' performance must not be ignored. The desired outcome of the logical compression layer is lower bandwidth and/or higher FPS (which essentially leads to lower frame delay and better user experience). The measurements have shown that the logical compression layer work is negligible in terms of server resources and that there is nearly no computational overhead, so it is worth using it.

Lossless compression/decompression time per frame is the time that was spent for compressing and decompressing the streamed data. The portion of time which the sever spends compressing data using lossless compression (with or without logical compression, which run at different average FPS). The results have shown that the logical compression does not add complexity to the lossless compression and the difference between the two modes is negligible.

Client – Rendering Time per Frame is the average portion of time per frame that the client spends on rendering (making actual graphic calls).

Frame rate and frame delay are used as the major measurements of the system in case of 3D streaming. The frame rate is influenced by the bandwidth of the network, the delay of the network and the computation time on the client (decompression, and rendering time) and the server (lossless compression and serialization time).

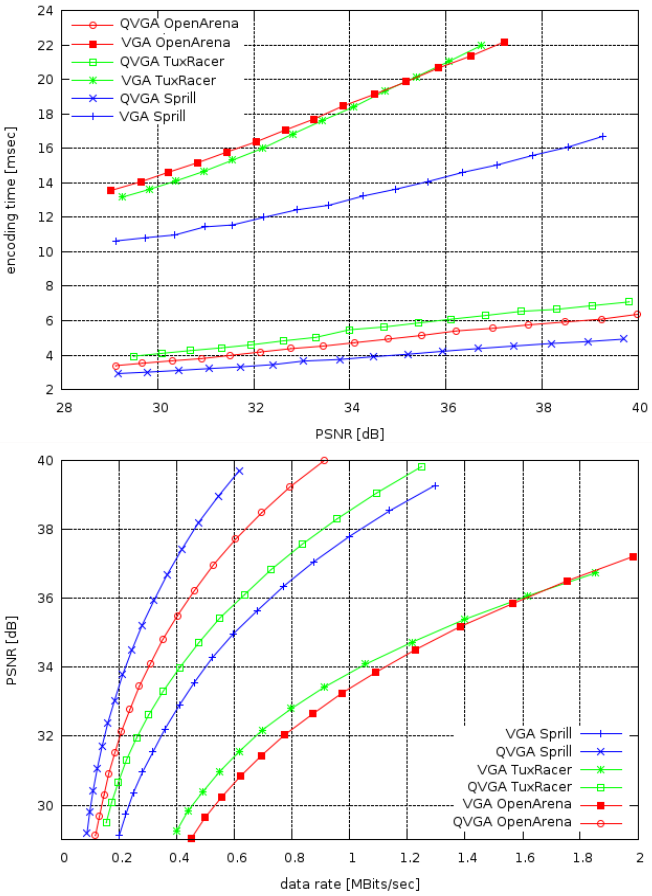

Figure 2. Video streaming performance measurements

Video streaming performance is analyzed by logging several measurements of the H.264/AVC encoder on a per frame bases: encoding time, resulting PSNR and SSIM image

quality as well as the fraction of I- and P-macroblocks. In Figure 2 we show the resulting graphs for three games, Sprill, Tux Racer and OpenArena, streamed with two different resolutions: QVGA (320x240) and VGA (640x480). The measurements were performed on a quad-core 3 GHz computer and represent typical measurements for the evaluation of video streaming performance. The graphs in the two diagrams show how big the resulting bit-rate will be for given resolution at a certain quality and how long the per-frame-encoding will take. Games with less movement like Sprill result in much better encoding performance than action games, like OpenArena or TuxRacer.

In addition external measurement tools such as Wireshark and QoSMeT [19] are used for high precision network measurements and analysis. The measured parameters include the following: throughput, one-way delay, one-way delay jitter, packet loss, amount of MAC-layer retransmissions, and amount of transport layer retransmissions. All the parameters are logged separately for both downlink and uplink directions to enable detailed analysis.

The system performance can be tested under different controlled network conditions by using network emulators (e.g. Netem) and traffic generators (e.g. iperf). With a network emulator we can introduce additional delay, delay jitter, packet losses, and bandwidth limitations. With traffic generators we can make additional network flows of different types to test G@L system's behaviour under competition.

V. CASE STUDY: G@L PERFOMANCE VS. CROSS-TRAFFIC IN WLAN

This section presents an example case of the iterative process between the protocol design and testing operations.

A number of tests was performed to study the behaviour of the 3D streaming protocol when competing against other traffic using the same wireless LAN. Several games of different genres were tested, but due to space limitations, the results are presented here only for Sprill game. A single game session was played in each test and the main parameters were measured to observe any problems when sharing the WLAN with additional traffic. No QoS was used in these tests.

Our test setup (Figure 3) included four laptops and a IEEE 802.11g WLAN access point. Laptop A was used as a game client while laptop B was running the G@L server software. Laptops C and D were used to generate competing traffic to the WLAN using the open source iPerf tool. The background traffic was generated from D to C and TCP protocol was used. All the performance measures as described in Section 3 were logged for analysis.

In addition, there have been invited four experienced gamers who have tested and rated the gaming experience for every game. The gaming experience was rated in the Mean Opinion Score (MOS) scale after each game session [20]. A game is considered to be not playable by most of the users when its gaming experience is rated below 3 in MOS scale.

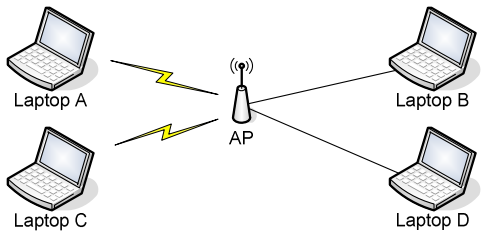

Figure 3. Testbed configuration

At the first time when the test was performed, it was found out that the 3D streaming protocol was not competing too well with other traffic using the network. Figure 4 shows the cumulative distribution function (CDF) of the realized frame rate presented to the user. As it can be seen, the game was presenting frames with a speed up to 200 fps when there was no competition and only 10% of time a frame rate lower than 25 fps was used. In the case of competition, the game could not work faster than 40 fps and the frame rate was lower than 25 fps 97% of time. Figure 5 shows similar problems in form of high downlink transmission delay when competing with other traffic. 30% of the time the transmission delay for downlink packets was above 100 ms. This alerted us to some problems in the protocol design and after more investigation on the protocol operation it was found out that application level acknowledgement function used to synchronize the client and the server during the game play made the game stream very cautious in using the network. In practice, a normal TCP stream was able to take the majority of network capacity leaving only a bit of capacity to the game stream.

When the problems in competitiveness of the graphics streaming protocol were found, the issue was addressed in the development by releasing a new version of the system which was not using application layer acknowledgements but instead was limited to the maximum frame rate of 25 fps. The same set of tests was run with the new version and a big change in the results could be seen. Now the competing traffic did not have a notable effect to the speed of frames presented to the user (Figure 4, dotted graphs). We still could see a significant effect in the transmission delay when introducing other traffic to the network, but the effect was now much smaller (Figure 6). Without competition, the delay was all the time lower than 7 ms and in case of competition it increased up to 15 ms. Anyway, the game was now competing equally with other TCP applications and did not suffer from being too cautious.

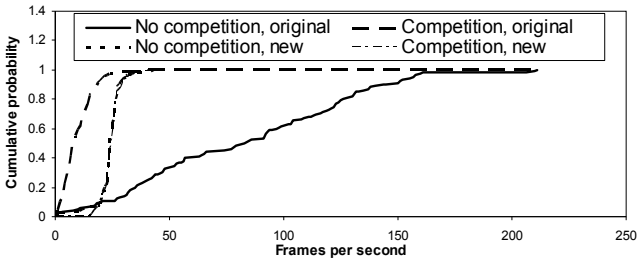

Figure 4. CDF for realized FPS, both designs

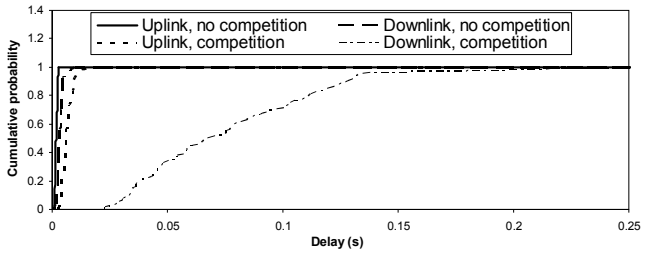

Figure 5. CDF for transmission delays, original design

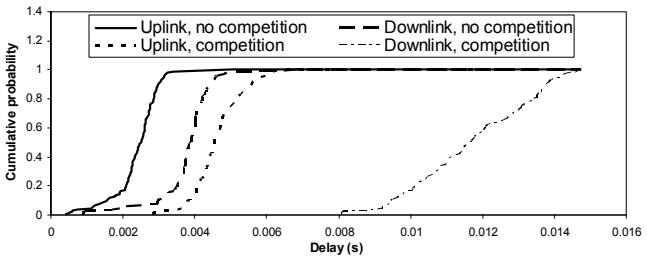

Figure 6. CDF for transmission delays, updated design

On the downside, the game could not use extremely high frame rates at all which may decrease the gaming experience in some cases as it can be seen from Figure 7 that represents the MOS score of the 4 test participants. The gaming experience in the presence of the competing traffic has been rated slightly better for the updated design but when there was no competition in the LAN the updated design scored lower ratings. This issue will be addressed in future development of 3D streaming protocol.

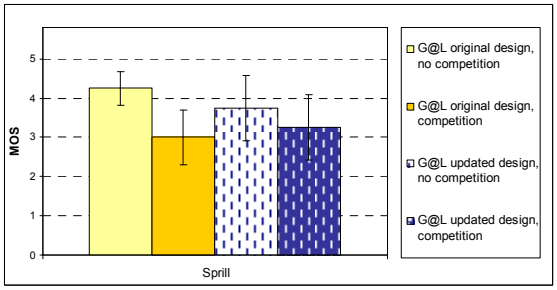

Figure 7. MOS rating of gaming experience

VI. Conclusions and Future Work

The benefits of our test and development methodology were demonstrated in a real case where a design weakness was discovered during the testing process and addressed in the protocol development. The issue of the trade-off between the graphics streaming protocol competitiveness and the resulting gaming experience will be further addressed in future development of the 3D streaming protocol. In addition to the discussed improvements for the future work, a similar test approach will be applied also for system tests with the video streaming.

The Game@Large system implements a subset of the features that are peculiar to the cloud computing and similar Gaming on Demand systems. Thus, the work presented in this paper is pioneering a more in-depth insight into game streaming systems and their development, which are currently one of the hot topics in cloud computing.

Acknowledgements

The work presented in this paper has been developed with the support of the European Integrated Project Games@Large which is partially funded by the European Commission.

References

[1] A. Jurgelionis, P. Fechteler, P. Eisert, et al., "Platform for Distributed 3D Gaming," International Journal of Computer Games Technology, vol. 2009, Article ID 231863, 15 pages, 2009.

[2] Y. Tzruya, A. Shani, F. Bellotti, A. Jurgelionis. Games@Large - a new platform for ubiquitous gaming and multimedia, Proc. BBEurope, Geneva, Switzerland, 11-14 December 2006

[3] A. Jurgelionis, F. Bellotti, P. Eisert, J.P. Laulajainen, Testing and evaluation of new platforms for delivery of distributed gaming and multimedia, Web3D 2007 Symposium, Perugia, Italy, 15-18 April 2007

[4] D. De Winter, P. Simoens, L. Deboosere, et al., "A hybrid thin-client protocol for multimedia streaming and interactive gaming applications", NOSSDAV '06, Newport, RI, USA, May 2006

[5] L. Cheng, A. Bhushan, R. Pajarola, and M. El Zarki, "Realtime 3d graphics streaming using mpeg-4", IEEE/ACM BroadWise '04, pp. 1–16, San Jose, Calif, USA, July 2004.

[6] Y. Noimark and D. Cohen-Or, "Streaming scenes to MPEG-4 video-enabled devices," IEEE Computer Graphics and Applications, vol. 23, no. 1, pp. 58–64, 2003.

[7] N. Cranley, M. Davis, "Performance evaluation of video streaming with background traffic over IEEE 802.11 WLAN networks", ACM WMuNeP '05, Montreal, Canada, Oct 2005.

[8] A. Ksentini, M. Naimi, A. Gueroui, " Toward an improvement of H.264 video transmission over IEEE 802.11e through a cross-layer architecture", IEEE Communications Magazine, vol 44, no. 1, pp. 107-114, Jan 2006.

[9] P. Svoboda, M, Rupp, Online gaming models for wireless networks, IASTED IMSA 2005, February 2005, Grindelwald, Switzerland

[10] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, M. Claypool, The effects of loss and latency on user performance in Unreal Tournament 2003, NetGames 2004, Portland, Oregon, USA

[11] M. Dick, O. Wellnitz , L. Wolf, Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games, NetGames 2005, Hwathorne, NY, USA

[12] K. Claypool, M, Claypool: On frame rate and player performance in first person shooter games, Multimedia Syst. 13(1): 3-17 (2007)

[13] P. Fechteler and P. Eisert, " Depth Map Enhanced Macroblock Partitioning for H.264 Video Coding of Computer Graphics Content", IEEE ICIP2009, Cairo, Egypt, November 2009

[14] IEEE, Std 802.11-1999 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[15] Std 802.11e-2005 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, 2005.

[16] Wi-Fi Alliance Technical Committee, QoS Task Group, WMM (including WMM power save) specification V1.1, 2004

[17] J-P. Laulajainen, *Implementing QoS Support in a Wireless Home Network*, IEEE WCNC 2008, Las Vegas, NV, USA, April 2008.

[18] UPnP Forum, UPnP QoS: Architecture V2.0, Oct 2006.

[19] J. Prokkola, M. Hanski, M. Jurvansuu, M. Immonen, "Measuring WCDMA and HSDPA Delay Characteristics with QoSMeT", in Proc. IEEE International Conference on Communications, June 2007

[20] Ch. Schaefer, Th. Enderes, H. Ritter, M. Zitterbart, Subjective Quality Assessment for Multiplayer Real-Time Games, NetGames 2002, Braunschweig, Germany , 2002