Games@Large Distributed Gaming System

Arto Laikari[1], Philipp Fechteler[2], Peter Eisert[3], Audrius Jurgelionis[4], Francesco Bellotti[5], Alessandro De Gloria[6]

[1]VTT Technical Research Centre of Finland, Espoo, Finland; [2,3]Fraunhofer Institute for Telecommunications, Berlin, Germany; [4,5,6]University of Genoa, Genoa, Italy;

E-mail: [1]Arto.Laikari@vtt.fi, [2]philipp.fechteler@hhi.fraunhofer.de, [3]eisert@hhi.fraunhofer.de, [4]jurge@elios.unige.it, [5]franz@elios.unige.it, [6]adg@elios.unige.it

Abstract: **The requirements of computer games by means of CPU and graphics performance are continuously growing. At the same time many low cost and modest performance CE devices are gaining popularity. People are already used to mobile life style inside home and on the go and want to enjoy entertainment everywhere. This paper describes a novel gaming system, called Games@Large, which enables heavy PC game play on low cost consumer electronic devices (CE) without the need of game software modification. The key innovations of the Games@Large system are distribution of game execution, streaming of graphics, video, audio, and game control as well as network quality of service management.**

Keywords: remote gaming, graphics streaming, video streaming, networking, input device capturing, remote control

1 INTRODUCTION

Home computers have already brought Ethernet and WLAN as standard equipment to homes. Wireless or wired home networks are almost as familiar and common as electricity and water pipes in many countries. Future home is considered to be an always-on connected digital home with wide variety of appliances. Although people are not yet willing to invest to a "future home", they are already eager to acquire a lot of entertainment equipment. Computer gaming has been utilizing this infrastructure already for a long time. Modern games have become highly realistic and they are consumed by a wide population, not only youngsters. Due to the increasing demands on computer hardware for the realistic connected virtual worlds, high CPU processing power and graphics performance is required to play these games.

At the same time mobility and digital home entertainment appliances have generated the desire to play not only in front of a home PC, but everywhere inside the house and also on the go. As the result of TV digitalization Set-Top Boxes (STBs) have entered homes and as a new trend mini-laptops are gaining popularity. Several low cost CE end devices are already available at home and on the go. Although these devices are capable to execute software, modern 3D computer games are too heavy for them.

Additionally, several Games-on-Demand services are currently entering the marked. OnLive, announced for late 2009, that they will provide instant game play in resolutions up to 1080i via browser plugins. This performance is achieved by running a proprietary video codec on customized hardware [17]. Playcast and Gaikai have both announced similar Game-on-Demand services also for late 2009.

The Games@Large project is developing a novel system for gaming both for homes and for enterprise environments, like hotels, internet cafés and elderly homes [1, 2]. In the Games@Large system the key concepts are execution distribution, audio, video and graphic streaming and decoupling of input control commands. According to the Cloud Computing concept, games are executed in one or more servers and the game display and audio is captured and streamed to the end device, where the stream is rendered and the game is actually played. Game control is captured at the end device streamed back to the server and injected to the game. It is important to note that no special game adaptation is required, but almost any commercial game can be played via the Games@Large platform. In comparison to other currently emerging systems, the Games@Large framework explicitly addresses support of end devices with rigorously different characteristics, ranging from PCs with different operating systems over settop boxes to simple handheld devices with small displays and low CPU power, with adaptive streaming techniques.

This paper presents the novel Games@Large architecture and its key functionalities, like the graphics and video streaming, audio streaming and game control command transfer as well as the quality of service approach. Finally we present the testing arrangements of the Games@Large system and the experimental results.

2 GAMES@LARGE ARCHITECTURE

The system depicted in Figure 1, consists of three major element classes: servers, end devices and access points. A user that interacts with the Games@Large system mainly uses his/her end device to browse, select and play games. Moreover, the system supports background activities such as: device discovery in the network, authentication and authorization of the device and user, log the activity in the system for tracking and billing, configure and maintain the system.

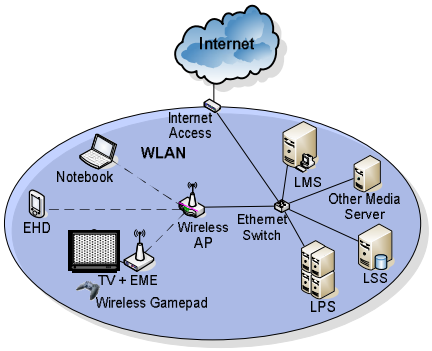

Figure 1: Games@Large architecture

Games run on the Local Processing Server (LPS), which utilize also the Local Storage Server (LSS). In the home version of the Games@Large system, these logical entities will be located in the same physical computer. End devices, like STBs or Enhanced Multimedia Extenders (EME), Enhanced Handheld Devices (EHD) and notebooks are connected to the server either with wireless or wired connection. The system exploits an adaptive streaming architecture and uses Quality of Service (QoS) functionalities to ensure good quality gaming to a variety of devices connected the wireless network. The six stage general Games@Large execution process is depicted in Figure 2.

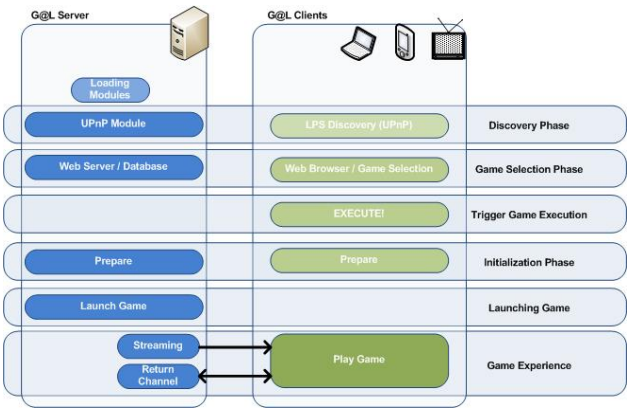

Figure 2: General flow of information

3 REMOTE RENDERING OF GAMES

The objective of the streaming architecture is to support as many kinds of end devices as possible with an efficient and high quality game experience, independent of software or hardware capabilities. Additionally, in order to keep the effort for integrating new end devices at a minimum, the involved protocols and algorithms have to be as much as possible standard compliant resp. commonly used techniques. To meet these demands a streaming architecture has been developed that is able to support two streaming strategies to display the game remotely: Graphics Streaming and Video Streaming (shown in Figure 3).

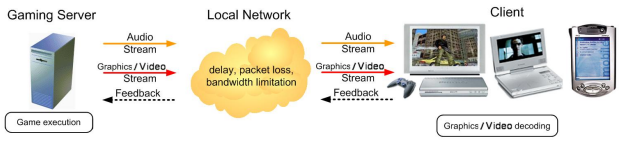

Figure 3: Graphics/Video streaming architecture

Graphics Streaming is used for end devices with accelerated graphics support, like computers or set-top-boxes, typically having screens of higher resolution [3]. Here the graphics commands from the game are captured, transmitted and rendered locally on the end device. In this way, high image quality is achieved, since the scenes are always directly rendered for the desired screen. To achieve interactivity with a minimum of delay, several optimizations have been developed: caching of server memory on client side, simulation of graphics state on server side and encoding/compression of graphics stream [15]. For cross-platform purposes an abstraction layer has been developed for the particular graphics commands of OpenGL resp. DirectX. These abstract commands are transmitted in order to be translated on the client appropriately to be handled by the graphics system present there. Figure 4 depicts the detailed block diagram of the components involved in the 3D streaming. First, the 3D commands issued by the game executable to the graphic layer API used by the selected game (e.g., DirectX v9) need to be captured. The same technique used for capturing the DirectX v9 can also be used for capturing other versions of DirectX (and also the 2D version of DirectX-DirectDraw). This is implemented by providing to the game running on the LPS a pseudo-rendering environment that intercepts the DirectX calls. The proxy Dynamic Link Library (DLL) is loaded by the game on its start-up and runs in the game context. This library forms the server part of the pipeline which passes the 3D commands from the game executable to the client's rendering module..

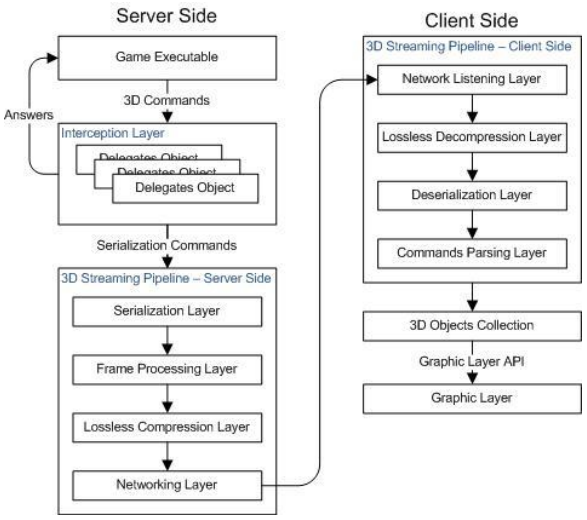

Figure 4: 3D streaming

Video Streaming, the alternative approach, is used mainly for end devices without a GPU, like handheld devices, typically having screens of lower resolution. Here the graphical output is rendered on the game server and the frame-buffer is captured and transmitted encoded as H.264 video stream [4]. In Video Streaming the bit rates are in general much more predictable in comparison to Graphics Streaming. However, H.264 encoding on server side as well as decoding on end devices is computational demanding. In order to support simultaneous execution of several instances of games in parallel and to achieve interactive frame rates with a minimum delay several optimizations have been developed. Besides the reduction of all buffering and caching on server and client to a minimum, information available in the games rendering context is exploited in several ways to reduce the computational complexity of H.264 encoding itself. First of all, the rendering commands are intercepted and modified before their execution to meet exactly the client's properties without any image degradation or processing delay (e.g. resolution by adapting

the viewport etc.). Additionally the H.264 encoding is accelerated in comparison to general purpose encoding by utilizing information of the rendering context. One integrated optimization exploits skyboxes, a commonly used game rendering technique [16]. Further optimizations are focused on direct calculation of motion vectors and macroblock partitions based on graphics information. In this way computational demanding trial and error search methods of generic H.264 encoding is significantly reduced.

Besides the visual appearance computer games also produce sounds and music. In order to deliver this audio data to the client in an efficient manner, an **audio streaming** sub-system has been developed. Since computer games typically produce their audio samples in a block-oriented manner, the current state-of-the-art audio encoder in this field has been integrated: the High Efficiency Advanced Audio Coding version 2 (HE AAC-v2) [5]. The AAC implementation is configurable to meet the capabilities of the connected end device (AAC-LC, HE-AAC or HE-AAC v2, mono or stereo, 8 or 16 bits per sample and at several sample rates, e.g., 22.05, 44.1 or 48 kHz).

To provide a synchronized and real-time playback of visual and audio data, the UDP based Real-time Transmission Protocol (RTP) [6,7] and Real Time Control Protocol (RTCP) have been implemented. The former protocol provides time stamps to each packet and the latter provides the information for synchronization of several RTP channels.

4 COMMUNICATION CHANNEL ARCHITECTURE

Different end devices typically provide several different input modalities. In the initial discovery phase performed by the UPnP device discovery protocol, the end device sends its properties and capabilities. During the gameplay, the input from the controllers is captured either using interrupts (keyboard, mouse) or through polling (joysticks, joy pads). The captured commands are then transmitted via persistent connection to the server. At the server side the input commands are mapped to the appropriate game control and injected in real-time to the game running at the server [8].

Figure 5 describes the functionality of the return channel. The return channel is constructed by two communicating modules; the server side module and the client side module. The return channel on the server side is responsible for receiving the commands from each connected client and injecting them to the appropriate game the user is playing.

The server side module that implements the return channel is part of the core of the system and more specifically the Local Processing Server. The return channel on the server side is responsible for receiving commands from each connected client and transforming them in such a form that they will be readable by the OS (Windows XP/Vista) and resp. by the running instance of the game.

The server side of the return channel receives the keyboard commands from the client. The communication between the server and the client follows a specific protocol in order to a) be successfully recognized by the server and b) preserve the loss of keyboard input commands. An important aspect of the return channel infrastructure is the encryption of keyboard commands.

An encryption procedure is used to secure the keyboard commands that the client transmits. This is done, because sensitive user data, such as credit card numbers or passwords might be inserted using the keyboard.

Each end device supports of many possible input devices for interacting with the server. Capturing the input coming from the controllers is achieved by recording the key codes from the input devices. Input devices such as mice or keyboards are interrupt-driven while with joysticks or joy pads the polling method is used for reading. If the command that is to be transferred is originating from a keyboard device, the client uses the server's public key to encrypt the data. The encrypted message is transmitted to the server using the already existing socket connection.

Similarly mouse commands that originate from the client are sent to the server using the already open socket connection without encryption.

An issue that arises for the mouse input device is the mapping of different resolutions at the server and client, since absolute mouse positions are sent. However we realized that the rightmost bottom position of the mouse equals the resolution of the game when running in 3D streaming, and it is equal to the screen resolution when running in video streaming. On the server side matching of the resolutions should be done on the resolution of the game running on the server as every command is injected into the game window. Therefore the mouse positions are normalized on the client and the server side.

Joypad/Other input from the client. This means that any Joypad/Other input is first translated to suitable keyboard and mouse commands on the client side (using XML mapping files) and is then transmitted to the server for execution at the game instance. The execution of these commands falls to the previously described cases [10].

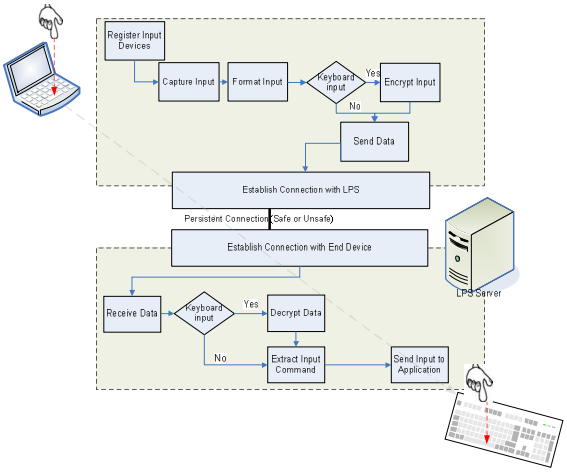

Figure 5: Communication channel

5 NETWORKING AND QUALITY OF SERVICE

To ensure smooth game play, decent Quality of Service (QoS) is required from the wireless home network. To fulfil this goal, we use both a MAC-layer mechanism to prioritize some traffic classes and an application-layer mechanism to easily map the application requirements to the MAC-layer QoS. In order to meet the requirement of using low cost components, the choice for the wireless home network has been to use IEEE 802.11 based technologies, because of their dominant position in the market [9].

Priority based QoS can be supported in IEEE WLANs with the Wi-Fi Multimedia (WMM) extensions specified by Wi-Fi Alliance. WMM is a subset of IEEE 802.11e standard and divides the network traffic into four access categories which receive different priorities for the channel access in competition situations. In this way applications with high QoS requirements can be supported with better service than others with less strict requirements. In addition to MAC layer QoS support, there is a need for QoS management solutions in a complete QoS solution. Our platform relies on UPnP QoS specification. The specification defines services for policy management and network resource allocation. In practice, it acts as a middleware between the applications and the network devices performing the QoS provisioning.

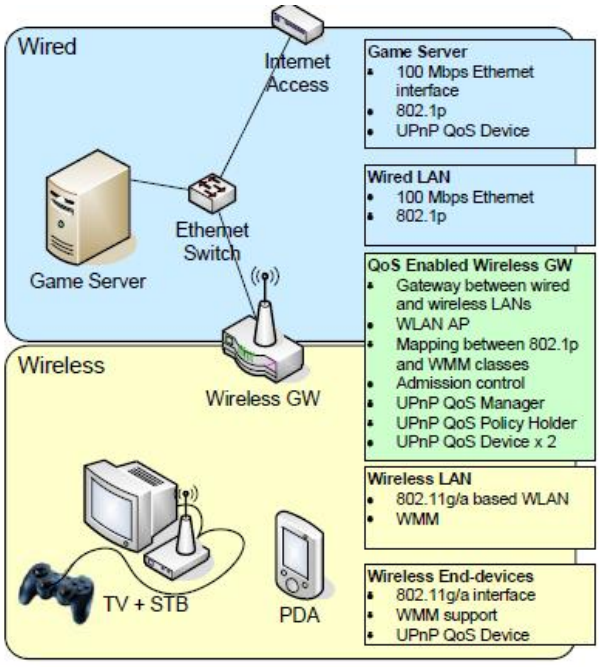

Figure 6: QoS enabled home network for gaming

6 EXPERIMENTAL RESULTS

In [10] we have demonstrated and discussed multiple game execution and Games@Large system performance with and without the QoS. Based on the results in [10] we have performed a more extensive study with 2 expert gamers and a larger number of different games for two different test scenarios of the Games@Large system operation. The two test cases consider different conditions in the wireless network, since this reflects the real potential of the Games@Large system. The test scenarios presented in this paper show the results of the initial tests for the Games@Large system. Later in the project more thorough tests with various end devices and multiple players for the system will be performed [10].

6.1 Testbed and test scenarios

We have redesigned the testbed from [10, 11] in which we can monitor the performance of network, devices and Games@Large system processes as well as introduce some background traffic in the wireless local area network. Figure 7 depicts the testbed setup.

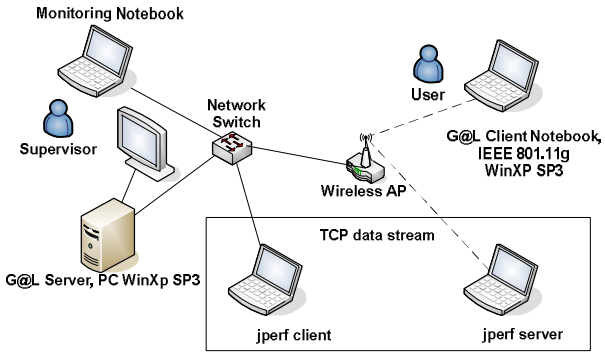

Figure 7: Games@Large testbed

The Games@Large server (Intel 2GHz 2 CPUs, 2048MB RAM, 256MB dedicated video memory) running Windows XP is connected to a 100 Mbps switch which is connected to the WLAN Access Point (AP) via a wired Ethernet connection. The client notebook (Intel 2GHz 2 CPUs, 1024MB RAM, 384MB shared video memory) is connected to the Wireless AP via the IEEE 802.11g wireless connection. For system performance monitoring we used an external Monitoring notebook.

All the PCs and NBs were SNMP/WMI enabled for performance monitoring purposes. For TCP traffic generation we have used two additional notebooks running jperf [12] server and client software, Figure 7.

We used a PRTG Network Monitor [13] on the Monitoring notebook to monitor network, device and Games@Large processes with minimal influence on system's performance.

An internal performance logging system was implemented in the Games@Large client and server software. The logging system can record the client's and server's processing delay and a game's frame rate at which it is rendered on the client notebook.

We have performed our experiments with four different video games: Sprill (casual game), Mahjong World (board game), Total Overdose Demo or TOD (first person shooter game), Turtix (casual game).

6.2 Test scenarios description

The first scenario is designed to measure the system's performance in a wireless network in nearly ideal conditions when there is no other traffic interfering with Games@Large game session streams.

In the second test scenario the wireless local network is loaded with a TCP traffic stream. In the case of TCP cross-traffic generation the jperf client sends as much data as there is free bandwidth.

The second test scenario considers the presence of cross-traffic in the Games@Large wireless network without the QoS. Such an approach is a good way to put the Games@Large system under stress conditions and analyse the acceptance by end users. The QoS is deactivated which implies that system performance for the second test scenario would be considerably improved in case the QoS is active [10].

The described test scenarios include a full system workflow which consists of the following steps: Games@Large server discovery from the client device, web user interface access and game list browsing, selection of the game and starting to play.

Prior to the tests both the test participants were given time to familiarize themselves with the 4 video games used for tests. They have been asked to play the games on the client notebook for a few hours. The games were installed and run natively on the client notebook.

In order to have reference data we have performed measurements and asked test participants to rate the gaming experience also when games were run natively on the client notebook.

Thus, the evaluation is based on the comparison between the game experience when a game is run on the client notebook natively and on the Games@Large framework.

The gaming experience was rated in the Mean Opinion Score (MOS) scale after each game session [14].

Each of the test sessions' for performance measurement and gaming experience rating was 5 minutes in length.

6.3 Results and discussion

Performance measurements and user ratings were averaged for each test session and are shown in the figures below.

Figure 8 represents the MOS score of the two test participants for each game for different test scenarios. In the presence of TCP traffic the gaming experience degradation is noticeable for all of the tested games; the Total Overdose Demo game was rated by one of the test participants as practically not playable in the presence of jperf's generated TCP cross-traffic, while the second one rated it much better, which suggests that this first person shooter game is considered as playable by non demanding gamers. A game is considered to be not playable by most of the users when its gaming experience is rated below 3 in MOS scale [14].

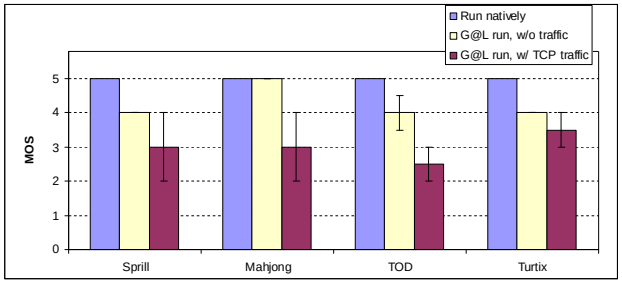

Figure 8: MOS rating of gaming experience

According to [10], figure 9 and figure 10 the frame rate of a game is proportional to the network throughput for the Games@Large system which decreases with the increasing round trip time (figure 10) and causes degradation in the game experience. The performance of the system improves a lot when the QoS is active [10]. Nevertheless, video games that use high data rates per frame require very high bandwidth and thus, problematic to be run with the Games@Large system via currently used wireless networks.

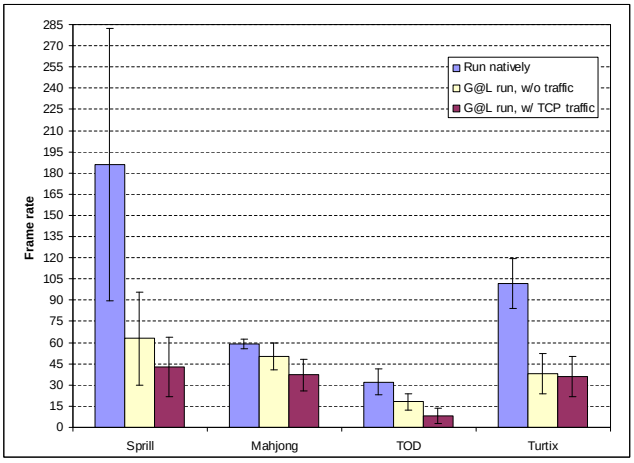

Figure 9: Average frame rates on the client notebook

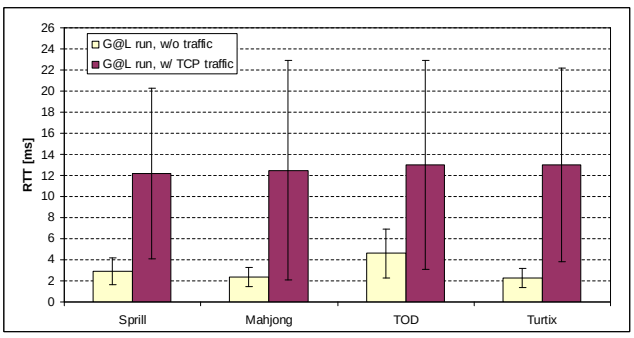

Figure 10: Average RTT between G@L server and client

Figure 11 and Figure 12 show the processing delay for client and server of the Games@Large system. It can be seen that for different network conditions (test scenarios) the variation in client and server delays is very small. Processing delay depends on the type of game and device specifications.

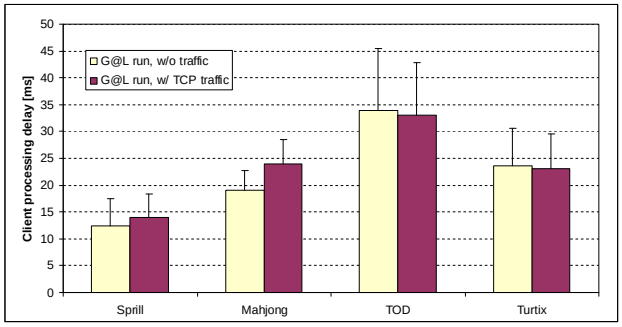

Figure 11: Average G@L client processing delay

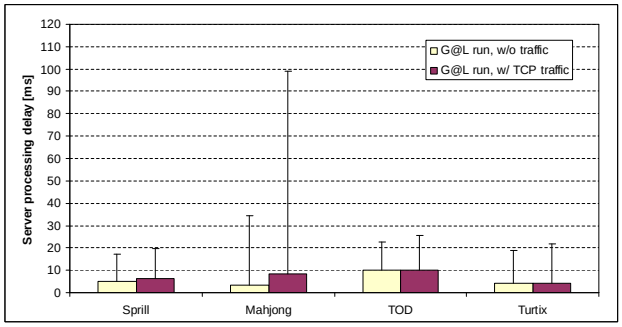

Figure 12: Average G@L server processing delay

The server must have enough CPU power and memory to run the game natively. Additionally, the amount of video memory that a game requires when running natively, must be available in the system memory when running in the Games@Large environment (that is because the graphic objects are emulated by the streaming module in the system memory). As for the CPU requirements, most games still do some graphics processing in software, so decoupling of the rendering from the game actually leads to a CPU gain on the server. As long as the processing server has sufficient CPU and memory resources to run multiple games at once it can run them.

The most important hardware requirement for the client device is the video adapter (for 3D streaming). It should have hardware acceleration capabilities to enable fast rendering of 3D scenes. As on the server, the graphic resources that the game stores in the video memory should be available in the system memory to enable manipulation prediction and caching. So memory requirements for the client should be 200-300 MB available to the client application for fairly heavy games.

7 CONCLUSIONS

Games@Large is implementing an innovative architecture, transparent to legacy game code, that allows distribution of a cross-platform gaming and entertainment on a variety of low-cost networked devices. Virtually extending the capabilities of such devices the Games@Large system is opening important opportunities for new services and experiences in a variety of fields and in particular for the entertainment in the home and other popular environments.

We have shown that the Games@Large system is capable of running and delivering a good gaming experience for video games of different genres including first person shooter games that are usually highly interactive and demanding for high end device performance.

Acknowledgment

This work has been carried out in the IST Games@Large project [1,2], which is an Integrated Project under contract no IST038453 and is partially funded by the European Commission.

References

[1] Games@Large project website, http://www.gamesatlarge.eu
[2] Y. Tzruya, A. Shani, F. Bellotti, A. Jurgelionis. Games@Large – a new platform for ubiquitos gaming, Proc. BBEurope, Geneva, Switzerland, 11-14 December 2006
[3] P. Eisert, P. Fechteler, Remote rendering of computer games, SIGMAP 2007, Barcelona, Spain, 28-31 July 2007.
[4] Advanced video coding for generic audiovisual services, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.
[5] MPEG-4 HE-AAC v2, ISO/IEC 14496-3:2005/Amd.2
[6] RFC 3550, “RTP: A Transport Protocol for Real-Time Applications”
[7] RFC 3640, “RTP Payload Format for Transport of MPEG-4 Elementary”
[8] Bouras, Poulopoulos, Sengounis, Tsogkas, Networking Aspects for Gaming Systems, ICIW 2008, Athens, Greece, 8-13 June 2008.
[9] J-P. Laulajainen, Implementing QoS Support in a Wireless Home Network, WCNC 2008, Las Vegas, USA, 31 March-3 April 2008.
[10] A. Jurgelionis et al, Platform for Distributed 3D Gaming, in Cyber Games and Interactive Entertainment, International Journal of Computer Games Technology, Article ID 231863, vol. 2009
[11] A. Jurgelionis, F. Bellotti, A. Possani, A. De Gloria, “Designing enjoyable entertainment products”, workshop on "Now Let's Do It in Practice: User Experience Evaluation Methods in Product Development" in CHI 2008, Florence, Italy, April 6th, 2008
[12] http://sourceforge.net/projects/iperf
[13] PRTG Network Monitor, http://www.paessler.com/
[14] Ch. Schaefer, Th. Enderes, H. Ritter, M. Zitterbart, “Subjective Quality Assessment for Multiplayer Real-Time Games”, in Proceedings of the 1st workshop on Network and system support for games, pp. 74 – 78, Braunschweig, Germany, 2002
[15] P. Eisert and P. Fechteler,"Low Delay Streaming of Computer Graphics," Proc. Intern. Conf. on Image Processing (ICIP), Oct 2008
[16] P. Fechteler and P. Eisert,"Depth Map Enhanced Macroblock Partitioning for H.264 Video Coding of Computer Graphics Content“ Proc. Intern. Conf. on Image Processing (ICIP), Nov. 2009
[17] BBC NEWS: “OnLive games service 'will work'”, 1st April 2009, http://news.bbc.co.uk/2/hi/technology/7976206.stm