Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 231863, 15 pages
doi:10.1155/2009/231863

Research Article

Platform for Distributed 3D Gaming

A. Jurgelionis,[1] P. Fechteler,[2] P. Eisert,[2] F. Bellotti,[1] H. David,[3] J. P. Laulajainen,[4] R. Carmichael,[5] V. Poulopoulos,[6,7] A. Laikari,[8] P. Perälä,[4] A. De Gloria,[1] and C. Bouras[6,7]

[1] *Department of Biophysical and Electronic Engineering, University of Genoa, Via Opera Pia 11a, 16145 Genoa, Italy*
[2] *Computer Vision & Graphics, Image Processing Department, Heinrich-Hertz-Institute Berlin, Fraunhofer-Institute for Telecommunications, 10587 Berlin, Germany*
[3] *R&D Department, Exent Technologies Ltd., 25 Bazel Street, P.O. Box 2645, Petach Tikva 49125, Israel*
[4] *Converging Networks Laboratory, VTT Technical Research Centre of Finland, 90571 Oulu, Finland*
[5] *Department of Psychology, Goldsmiths, University of London, New Cross, London SE14 6N, UK*
[6] *Research Unit 6, Research Academic Computer Technology Institute, N. Kazantzaki, Panepistimioupoli, 26504 Rion, Greece*
[7] *Computer Engineering and Informatics Department, University of Patras, 26500 Patras, Greece*
[8] *Software Architectures and Platforms Department, VTT Technical Research Centre of Finland, 02044 VTT, Espoo, Finland*

Correspondence should be addressed to A. Jurgelionis, jurge@elios.unige.it

Received 1 February 2009; Accepted 18 March 2009

Recommended by Suiping Zhou

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 CE devices avoiding the necessity of placing a noisy workstation in the living room or costly computers/consoles in each room of a hotel. This paper presents a new cross-platform approach for distributed 3D gaming in wired/wireless local networks. We introduce the novel system architecture and protocols used to transfer the game graphics data across the network to end devices. Simultaneous execution of video games on a central server and a novel streaming approach of the 3D graphics output to multiple end devices enable the access of games on low cost set top boxes and handheld devices that natively lack the power of executing a game with high-quality graphical output.

1. Introduction

Computer games constitute nowadays one of the most dynamic and fastest changing technological areas, both in terms of market evolution and technology development. Market interest is now revolving around capitalizing on the rapid increase of always-on broadband connectivity which is becoming ubiquitous. Broadband connection drives a new, digital "Future Home" as part of a communications revolution that will affect every aspect of consumers' lives, not least of which is the change it brings in terms of options for enjoying entertainment. Taking into account that movies and music provided by outside sources were at home long before the internet and broadband, the challenge is to invent new content consumption patterns of existing and new types of content and services [1].

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

Running an interactive content-rich multimedia applications (such as video games) requires the high performance hardware of a PC or a dedicated gaming device. Other devices such as set top boxes (STBs) or handheld devices lack the necessary hardware and adding such capabilities to these devices will cause their prices to become prohibitive [1].

A system which enables rendering of PC games on next-generation STB and personal digital assistant (PDA) devices without causing a significant increase in their price is a solution for future networked interactive media. This approach enables a pervasive accessibility of interactive media from devices that are running on different platforms (architecture and operating system), thus facilitating users to enjoy video games in various environments (home, hotel, internet café, elderly home) without the need to attach to a single device or operating system, for example, a Windows PC.

This paper describes the Games@Large (G@L) pervasive entertainment architecture which is built on the concept of distributed remote gaming [2] or Virtual Networked Gaming (VNG). It enables pervasive game access on devices (set top boxes and handheld devices) that typically do not possess a full set of technical requirements to run video games [1]. In general, the system executes games on a server PC, located at a central site or at home, captures the graphic commands, streams them to the end device, and renders the commands on the end device allowing the full game experience. For end devices that do not offer hardware accelerated graphics rendering, the game output is locally rendered at the server and streamed as video to the client. Since computer games are highly interactive, extremely low delay has to be achieved for both techniques. This interactivity also requires the game controllers' commands to be captured on the end device, streamed to the server, and injected into the game process [3]. The described functions are implemented by an application which is running on the client and a "return cannel" which is constructed between the clients and the server. The application on the client is responsible for recording any input command arriving from every existing input device while the return channel is utilized in order to send the commands from the clients to the server for execution. On the server side the commands are injected into the proper game window.

In order to ground our research and system developments, we have performed a thorough analysis of state of the art in gaming platforms available in today's market, presented in Section 2. The rest of the paper is organized as follows: Section 3 describes the Games@Large framework; Section 4 its components and operation fundamentals; Section 5 presents some experimental results on tests of the initial system and its components demonstrating multiple game execution on a PC and Quality of Service (QoS) optimized transmission of the games' graphics to the end devices via a wireless network; Section 6 presents the conclusions.

2. Gaming Platforms Analysis: State of the Art in Consoles, PC and Set Top Boxes

We have conducted an overview of state of the art in common gaming platforms such as consoles, PCs and set top boxes. One recent development in gaming market activities which has implications for new consumption patterns is technology based on distributed-cross-platform computing (or cloud computing); we introduce and overview this relatively new concept of Virtual Networked Gaming Platforms (VNGP) in Section 2.4.

2.1. Consoles. The home console system enables cheap hardware and guarantees product quality. Unlike the past, console functionality is being continuously upgraded post-release (e.g., web-browser and Wii channels on the Wii; high-definition video-on-demand downloading for Xbox 360; and PlayStation Home for PS3).

Xbox 360 (Microsoft). Microsoft were the first to release their next generation console, the Xbox 360, followed by the Xbox 360 Elite designed to store and display high definition video with a 120 GB hard drive. The Xbox 360 has perhaps the strongest list of titles of the three next gen consoles, including Halo 3 in 2007, though the style and content of each console's titles differ from the others and personal preferences play a role in which catalogue, and therefore which platform, appeals most to a certain gamer/user. In online functionality Microsoft is the most well established with its Xbox Live/Live Anywhere/Games for Windows-LIVE gaming services, Live Marketplace (used to distribute television and movies), and online Xbox Live Pipeline.

PlayStation 3/PS3 (Sony). As the most powerful games console ever made, it is the most future-proof in terms of where games development can go and its built-in Blu-Ray player. Expert reviews on the console have improved since its initial reception and commentators have remarked that the first PS3 games only use about 30–40% of the platform's capacity, so the gaming experience it offers should improve as developers use more of its capacity. PS3 functionality includes streaming movies or PS3 games to PSP over LAN. In Europe the PS3 is backwards compatible with most of the massive PS2 games catalogue (with all in US and Japan). Online Functionality/Support: The PS3 has a web browser based on NetFront; Home is a free community-based gaming service.

Wii (Nintendo). Nintendo's Wii features gesture recognition controllers allowing intuitive control and more physical play which must take much credit for the Wii's successful appeal to many consumers who had not been gamers before. The Wii also has a large back-catalogue of GameCube titles and developing games is cheaper and easier than for other platforms, suggesting a rapid proliferation of titles. Online functionality: Opera web browser software; a growing number of Wii Channels; the Message Board supports messaging with Wii users around the world via WiiConnect24, handles the Wii email messaging, and logs all play history, facilitating parental supervision; some titles now support multiplayer online play.

2.2. PCs. The PC is an open system which can be exploited by virtually any game manufacturer. It is also the broadest of gaming platforms—catering to casual games and casual gamers but also through to the top end of digital gaming in specialised gaming PCs. The PC has by far the highest install base of all gaming platforms (discounting simple mobiles) with rising broadband connections and very well-developed online games services, such as Games for Windows-LIVE, PlayLinc, and many casual games sites (e.g., Verizon, DishGames, RealArcade, Buzztime). Though relatively expensive, it is bought and used for many things besides gaming but is often not equally accessible to all members of the household. This multifunctional nature of the PC is being eroded by nongaming functionality being added to consoles. Game Explorer is a new one-stop application within Vista designed to make game installation far simpler and also allows parents to enforce parental controls.

Input Devices. The PC and the games based on it use keyboard and mouse as the input device, which allows more complex games to be played but does not travel well into the living room where the large-screen TV, 10-foot viewing experience, comfy chairs, and social gaming are enjoyed by console gamers. There are some existing living room-friendly PC-game controllers though they have not been widely taken up. Microsoft's wireless game-pad is compatible with both the PC as well as the Xbox 360. A gamepad is, however, not suited for playing some game genres associated with the PC (notably MMOGs and real-time strategy/RTS) but viable alternative control devices do exist which could allow PC games of all genres to successfully migrate to the TV (e.g., Microsoft's qwerty keyboard add-on for the Xbox 360/PC wireless gamepad, trackball controllers such as the BodieLobus Paradox gamepad, or the EZ Commander Trackball PC Remote). They could also help the development of new games and peripherals and support web features on TV (such as Intel/Yahoo's planned Widget Channel).

2.3. Set Top Boxes. The Set Top Box is emerging as a platform for casual games and some service providers are offering games-on-demand services (e.g., the long-established Sky Gamestar). The fast growth of digital terrestrial television (DTT) in Europe also suggests the STB install base will rise steadily, potentially greatly increasing its role. With a potentially large mainstream audience, support from advertising revenues could be significant for STB gaming. Wi-Fi-enabled set top boxes (e.g., Archos TV+) are starting to emerge which combine a Wi-Fi Media Player with a high-capacity personal video recorder (PVR) for enjoying movies, music, photos, podcasts, web video, the full internet and more on widescreen TV.

Several companies are committed to enabling gaming services for the STB platform, including Zodiac Interactive, PixelPlay, Buzztime, TV Head, and PlayJam in the US, and Visiware, G-Cluster, and Visionik (part of NDS) in Europe. These companies provide their content and technology solutions to a few TV service providers currently deploying gaming services, including BSkyB, Orange, EchoStar, and Cablevision. Several Telco TV and DBS TV service providers in the US are actively exploring 3D STB gaming and their demos make many of today's cable STB games look antiquated (see Section 2.4 for details of NDS Xtreamplay technology).

User uptake of gaming platforms and choice of console depend on games catalogues and online services as well as hardware specifications and functionality. PC games are effectively tied to the desktop/laptop and console gaming is seen by many as expensive or for dedicated gamers only. The Wii has broadened the console user base but there remains a massive potential for mainstream gaming on TV given the right technology solution, content/services offerings and pricing. The open PC platform is supported by much programming expertise and is powerful and ubiquitous but PC games need to make the transition to the more comfortable and social TV-spaces with a wide range of low-cost, accessible, digitally distributed games-on-demand.

2.4. State of the Art in Virtual Networked Media Platforms. Of great relevance to Games@Large are developments in technology aimed at putting PC gaming onto TV screens. Service providers and web-based services are moving into the PC-gaming value chain and several commercial solutions for streaming games over the network exist already. These allow 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 low end-to-end delay, high compression efficiency, and low encoding complexity. Therefore, many solutions have adapted standard video streaming and optimized for the particular graphical content. For example, t5 labs announced a solution for instant gaming on set top boxes via centralized PC based servers, which are hosted by cable TV or IPTV operators. In order to reduce the encoding complexity at the server which has to execute the game and the video encoder, particular motion prediction is conducted exploiting information about the graphical content. Reductions of 50–80 % in encoding complexity are reported. In contrast, StreamMyGame from Tenomichi Limited also offers a server solution which enables the user to stream his/her own PC games to another PC in the home, record the game play or broadcast the games for spectators. The streaming is based on MPEG-4 and typical bit-rates of 4 Mbit/s at XGA resolution are reported. Besides PCs, multiple different end devices are supported such as PlayStation 3, set top boxes and networked media devices. Similar to the other two approaches, G-Cluster's server client system also offers MPEG-based compression of game content and its streaming to end devices for remote gaming applications. Currently, this system has been employed by operators mainly for casual games. A system that offers high-definition (HD) resolution is the Xtremeplay technology of NDS. They enable the high resolution streaming of computer games to set top boxes, but adaptations of the game code to the Xtreamplay framework are required. High resolution streaming of game content is also provided by the Californian company Dyyno. However, their application is not interactive gaming over networks but the distribution of game output to remote displays. Another somewhat different approach is AWOMO from Virgin Games. In contrast to the other approaches, they do not stream the game output but the game code. The game is downloaded and installed locally on a PC for execution. However, the technology offers a progressive game download, such that the user can start playing the game after only a small part of the data has been received. The remaining data is continuously fetched during game play. A similar approach is also used by the InstantAction system from GarageGames. Users can play 3D games in their web browser. InstantAction uses a small plug-in and an initial download of the game which are required to allow play.

Another streaming solution is offered by Orb (http://www.orbnetworks.com). Downloading Orb's free remote-access software, MyCasting 2.0, onto a PC (Windows only) transforms it into a 'broadcast device', the content of which can now be accessed from any web-enabled device (PC, mobile phone, etc.) with a streaming media player. MyCasting 2.0 now works with gaming consoles, enabling Xbox 360/Wii/PS3-owners to stream PC content onto the TV. Orb's software has enabled 17 million households (according to ABI Research) to bridge the PC-to-TV divide, at no cost, using what is essentially existing technology. However, streaming of video games is not supported.

Advances in wireless home entertainment networks and connectivity—which stream content between devices within the home—also present potentially important solutions for playing PC games on TV screens. For example, Airgo Networks' faster-than-wired True MIMO Media technology will allow streaming of rich high-definition television (HDTV) content to SimpleWare Home (STMicroelectronics) enabled devices within the home (at speeds faster than 10/100 Ethernet). Intel has collaborated with Verizon to launch a games-on-demand service that allows consumers to play PC games on their TV sets using Intel Viiv PCs. Also planned is a version of the online multiplayer service, PlayLinc, which will tie in with the service.

Although 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. In [4], for example, a thin client has been presented, that uses high-performance H.264 video encoding for streaming the graphical content of an application to a weaker end device. In this work, the buffering scheme at the client has been optimized in order to achieve minimal delay necessary for interactive applications, but encoding is based on standard video encoding. In contrast, [5] exploits information from the graphics scene in order to directly compute the motion vectors and thus significantly reduces the computational complexity of the MPEG-4 encoding process. The work in [6] also uses MPEG-4 as codec but goes one step further by using more information from the graphics state. For example, different quantizer settings are used dependent on the z-buffer content. Thus, objects that are further away in the scene are encoded with lower quality than foreground objects closer to the camera. Both approaches, however, require an application that passes the necessary graphics information to the codec and does not work with existing game programs. If encoding complexity should be reduced even more, simple encoding techniques can be used. In [7], a nonstandard compliant codec is presented that allows the streaming of graphics content with very little encoding effort. Coding efficiency is, however, also much lower than when using highly sophisticated codecs like H.264.

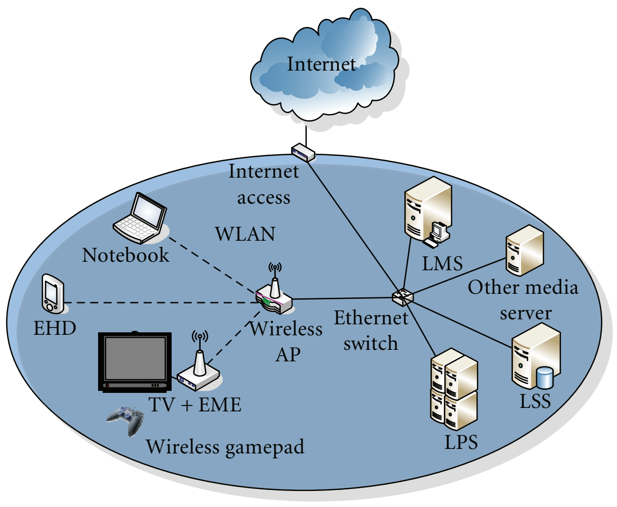


Figure 1: Games@Large framework.

The reviewed systems offer a variety of possibilities though all of them have limitations for interactive media such as video games, and especially existing game titles. For some of these formats games would need to be specially made or expensive hardware purchased, other formats provide moderate visual quality, unlike Games@Large's aim of being able to run all or most standard PC games including newly developed ones with high visual quality (in Sections 4 and 5 we will present some criteria for titles to be supported by Games@Large). Games@Large aims to offer benefits for wider stakeholders too (service providers, games developers/publishers, CE manufacturers, and advertisers) enabling business models which ensure that end users benefit not only from the technology solution but a wide choice of products and services at low cost.

3. Games@Large Framework

The Games@Large framework depicted in Figure 1 enables interactive media streaming from a PC-based machine to other CE, computer and mobile devices in homes and enterprise environments such as hotels, internet cafés and elderly homes.

The framework includes the following main components that are briefly introduced below and described in detail in Section 4.

Server Side. The Local Storage Server (LSS) is responsible for storage of games. The Local Processing Server (LPS) a Windows PC runs games from LSS and streams to clients. It is responsible for launching the game process after client-side invocation, managing its performance, allocating computing resources, filing system and I/O activities, and capturing the game graphic commands or already rendered frame buffer for video encoding, as well as managing execution of multiple games. The LPS is further responsible for receiving the game controller commands from the end device and injecting them into the game process. The LPS is also responsible for streaming game audio to the client.

Graphic Streaming Protocol Stack. The Graphics Streaming Protocol is intended to become a standard protocol used for streaming 3D commands to an end device allowing lower performance devices such as STBs to present high performance 3D applications such as games without the need to actually execute the games on this device.

The video streaming scenario is intended for devices lacking hardware accelerated rendering capabilities. H.264 [8] is exploited for low-delay video encoding. Synchronisation and transmission is realised via UDP-based RTP/RTCP in a standard compliant way.

HE-AACv2 [9] is used for audio streaming. Again, synchronisation and transmission is realised via UDP-based RTP/RTCP in a standard compliant way.

Client Side devices. Notebook (NB); Enhanced Multimedia Extender (EME), which is a WinCE or Linux set top box; Enhanced Handheld Device (EHD)—a Linux-based handheld. The client module is responsible for receiving the 3D commands and rendering them on the end device using local rendering capabilities (OpenGL or DirectX). For the video streaming approach, H.264 decoding must be supported instead. The client is also responsible for capturing the controller (e.g., keyboard or gamepad) commands and transmitting them to the processing server [3].

4. Games@Large Framework Components

4.1. 3D Graphics Streaming. Today, interfaces between operating system level libraries, such as DirectX and OpenGL, and the underlying 3D graphics cards, occur in the operating system driver and kernel level and are transmitted over the computer bus. Simultaneous rendering of multiple games and encoding their output can overload a high-performance server. For that purpose DirectX, and/or OpenGL graphics commands, has to be captured at the server (LPS/PC) and streamed to the client (e.g., STB or a laptop) for remote rendering. This is similar to the 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 usually work in an error-free TCP/IP scenario, with best effort transmission without any delay constraints.

The 3D streaming and remote rendering developed for Games@Large are achieved by multiple encoding and transmission layers shown in Figure 2. First of which is the interception and the very last one is the rendering on the client machine. All layers in between these two are independent of any specific graphics API. The latter implies that the postinterception 3D data streamed till the client rendering process is not specific to either DirectX or OpenGL, but rather utilises higher-level concepts common to all 3D graphics.

Since efficient direct translation from DirectX API commands to OpenGL commands is difficult, due to the significant differences between these APIs, a set of common generic concepts may be of assistance. In general, a 3D scene consists of multiple objects that are rendered separately. Before rendering an object, several parameters (states) must be set and these include lighting, textures, materials, the set of 3D vertices that make a scene, and further various standard 3D transforms (e.g., translate, scale, rotate).

Figure 2 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.

In our implementation, we have implemented delegates objects for each of the 3D objects created by the game. Each such delegates object uses the 3D streaming pipeline for processing the command and its arguments. For many commands, a delegate's object can answer the game executable immediately without interaction with the client—this is done in many cases in order to avoid synchronized commands. For example, when the game needs to change a texture (or vertex buffer) on the graphic card, it first locks it, and then it changes the buffer and then unlocks the texture. Originally, those commands must be synchronized. But in our implementation, the delegate object for texture does not interact with the client when the game tries to lock the texture on the graphic card but postpone the call for the unlock call. When the game issues an unlock call, the delegate object checks what parts of the texture were changed and sends a single command to the client with the changes. The client implementation, which is aware of this logic, will first lock the corresponding texture on the client's graphic card, change the texture and unlock it. This is one example of commands virtualization that allows avoiding synchronous commands, and reducing the number of commands—typically such a set of commands is called hundreds of times per frame.

The Serialization Layer serializes various structures describing the graphics state to a buffer. Serializer's additional function is to fill the buffers until certain criteria is met (theoretically it can pass the buffer to compressor after each command which, of course, would not be efficient for networking). The compression layer's purpose is to use an

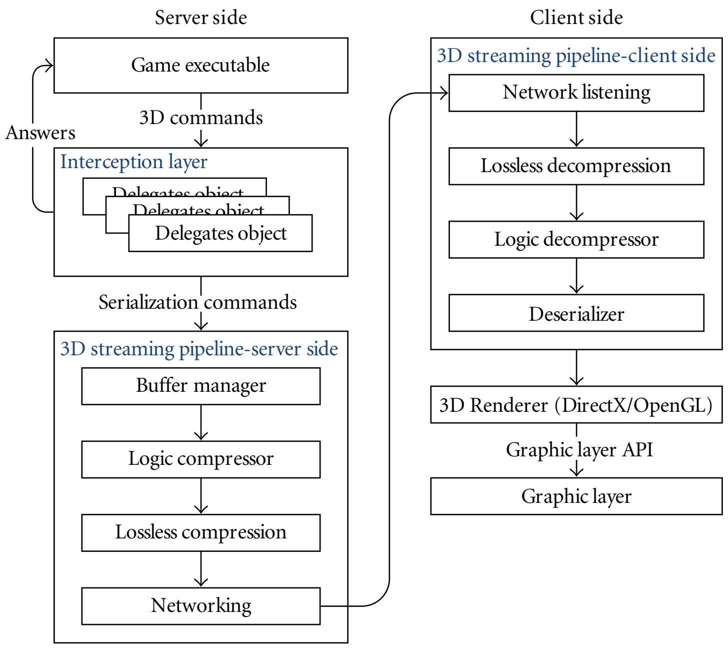


Figure 2: 3D Streaming—detailed block diagram.

efficient third-party compression library (e.g., zlib or LZO compression) to compress the 3D stream before sending it to the network.

The Network Layer is responsible for maintaining the connection with the client and for sending the buffers. After each sent buffer, an ACK (acknowledgement) is sent back by the client. The purpose of this ACK is to further synchronize server and client and to try to not overflow network buffers. The nature of the data requires that no buffer will be lost in transmission (which, in the current implementation, implies the use of TCP). A possibility to use or develop a transport protocol (e.g., UDP based) which could replace TCP is investigated.

On Microsoft Windows clients the renderer is using DirectX to render the commands, while in Linux clients the renderer is using OpenGL commands. There is a certain overhead in OpenGL rendering because some data (especially colour and vertex data) must be reorganised or rearranged in the processing stack before it can be given to OpenGL for rendering. This may result in increased demand of Central Processing Unit (CPU) processing and memory transfer between system memory and the Graphics Processing Unit (GPU) [3].

Although the graphic streaming approach is the preferable solution since it offers lower latency and enables execution of multiple games on one server, it cannot be used for some small handheld devices like PDAs or smart phones. These end-devices typically lack the hardware capability for accelerated rendering and cannot create the images locally for displaying them. Therefore, the alternative solution using video streaming techniques is described in the next section.

4.2. Video Encoding. The alternative approach to 3D Graphics Streaming in the Games@Large framework is Video Streaming. It 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 video stream. For video encoding, the H.264 video coding standard is used [8], which is the current state of the art in this field and provides the best compression efficiency. But in comparison to previous video coding standards, the computational complexity is significantly higher. However, by selecting appropriate encoding modes, the encoding complexity for the synthetic frames can be significantly reduced while preserving high image quality.

In order to keep the effort moderate for integrating new client end devices into the Games@Large framework, the video streaming subsystem has been developed in a fully standard-compliant way. Nevertheless, the server side encoding and streaming is adapted to the characteristics of the present end device. This means that end device properties such as display resolution or supported decoding profiles are selected appropriately on the server (e.g., optional H.264 encoding with CABAC [10], which typically increases the compression efficiency at the cost of increased computational load of decoding at the client). Similarly, the proportion of IDR frames in the resulting video stream, which are used to resolve the dependence on previous frames, can be set under consideration of the network properties.

The delay between image generation on the server side and presentation on the client side is crucial and has to be as small as possible in order to achieve interactive gaming. To reduce this delay a H.264 decoder for end devices has been developed which is implemented with a minimum of buffering. As soon as a video frame has been received it will be decoded and displayed. This is quite different to TV

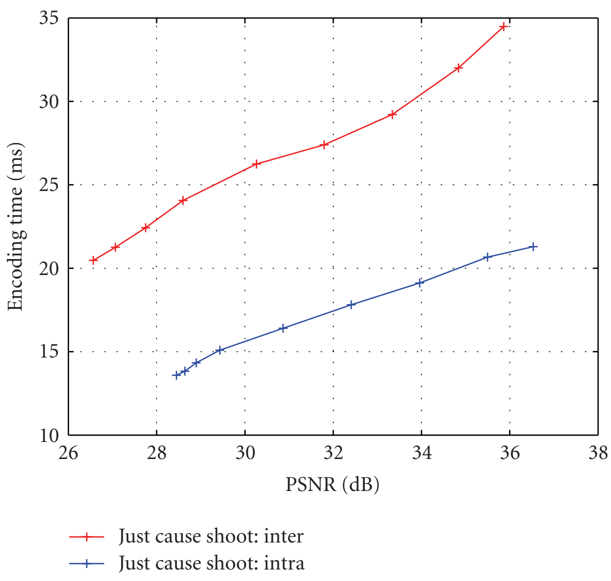


Figure 3: Comparison encoding timings for different quantizer settings.

streaming where large buffering is used to remove the effects of network jitters.

H.264 video encoding is computationally quite demanding. In Figure 3 the encoding times for a game scene are depicted for streams encoded with different quantizer settings which results in different qualities. It is clearly visible that for increased image quality the encoding time increases. Since the video encoding is executed in parallel to the actual game both are competing for the processor time. Aside from that, the desire to execute and stream several games simultaneously from a single game server increases the need for reduction in computational complexity in the video streaming system.

One method for reducing the complexity at the server is the removal of the scaling of the games output to the required resolution of the client device. For that purpose, the render commands of the game are intercepted and modified, so that the rendered images always fit the end device's resolution. Besides the reduction in complexity, an advantage of this technique is that the quality of the images achieved is much better, because the images are already rendered at the desired resolution without any scaling artefacts. An example is depicted in Figure 4.

Current research is focused on reducing the computational complexity of the H.264 encoder itself by incorporating enhancements based on the available rendering context information. The main idea is adapted from [11]. The motion prediction in video encoding, which is realized in common encoders as a computationally very demanding trial and error search, can be calculated directly by using the current z-buffer as well as projection parameters available in the games rendering context of OpenGL/DirectX. The encoding complexity can be reduced further by predicting the macroblock partitioning on the basis of discontinuities in the z-buffer. This is also usually realized in common encoders as a computationally demanding trial and error search. The key difference to [11] is that in [11] the authors assume to have full access to the rendering applications source code. In the Games@Large framework the output is generated from unmodified commercial games, which use quite sophisticated rendering techniques. The challenge here is to capture the appropriate information of the rendering context in order to correctly perform the motion prediction.

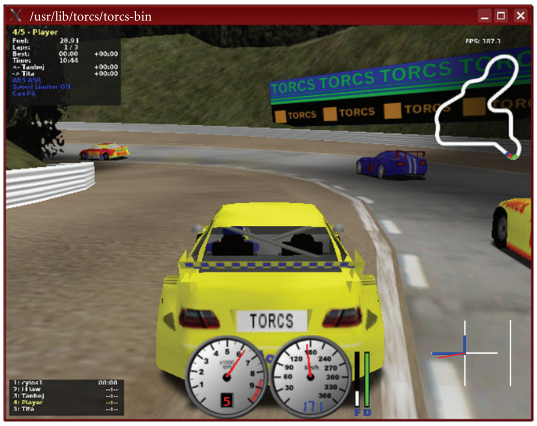
In order to transmit the encoded video stream in real-time, the RTP Packetization (Real Time Protocol [12]) is utilized. The structure of the H.264 payload for RTP is specified in [13]. Further details about real-time streaming and synchronization are discussed in Section 4.4.

4.3. Audio Encoding. Besides the visual appearance computer games also produce sounds. 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) [9]. Our HE AAC-v2 implementation is configurable so that it can encode mono or stereo, 8 or 16 bits per sample and at several sample rates, for example, 22.05, 44.1, or 48 kHz. In order to stream the encoded audio data in real-time the RTP packetization (Real Time Protocol [12]) is utilized. The structure of HE AAC-v2 payload for RTP is specified in [14]. Further details about real-time streaming and synchronization are discussed in Section 4.4.

4.4. Synchronized Real Time Streaming. Since the performance of the system is highly dependent on the delay between content generation on the server side and its play back on the client, the video streaming as well as the audio streaming are based on the UDP-based RTP (Real Time Protocol [12]). Every RTP network packet contains a time stamp as well as a well defined structure of payload data.

In order to prevent errors of different timings among the video and audio channels and to overcome different kinds of network jitters, the RTP channels are explicitly synchronized. For this purpose the RTCP (Real Time Control Protocol [12]) has been integrated. The content-generating server periodically sends a so-called Sender Report RTCP Packet (SR) for each RTP channel. This SR contains a mapping from the timestamps used in the associated RTP channel to the global NTP (Network Time Protocol [15]). With this synchronization of each RTP channel to NTP time, all the RTP channels are synchronized implicitly with each other.

4.5. Client Feedback to the Game Server. 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 one that the user is playing. The return


(a)

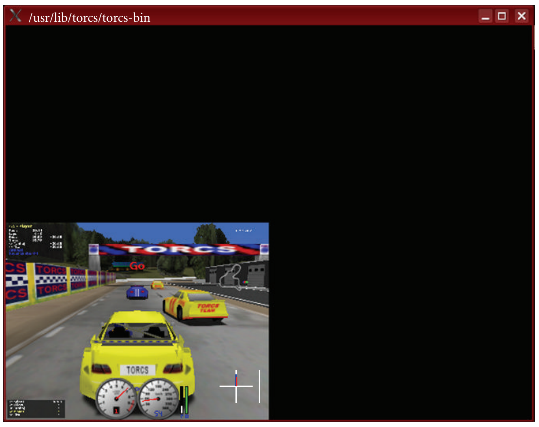

(b)

FIGURE 4: Rendering in resolution adapted to particular end device.

channel is constructed by two communicating modules; the server side module and the client side module.

4.5.1. Server Side. 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 more specifically by the running instance of the game. The method utilizes a proxy of the DirectInput dynamic library and injects the commands directly to the DirectInput functions used by each game.

A crucial part of the server and client side return channel is the socket communication. The HawkNL [16] library is used for the communication between the server and the clients. This assures that the implementation of the socket is based on a system that is tested by a large community of users and that no major bugs exist on that part of the code. For faster communication between client and server we disable the Nagle Algorithm [17] of the TCP/IP communication protocol. Having done so, the delivery times of the packets are almost instantaneous as we omit any buffering delays.

4.5.2. Keyboard. The server side of the return channel receives the keyboard commands that originate from the client dedicated socket connection. 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 which is described in the following section.

For the case of a game that uses a DirectInput keyboard, we implement a proxy dll method. For this method, we create a modified dinput8.dll of our own, modifying only the function that is used for passing data to the virtual DirectInput keyboard device that is created when the game launches in order to read data from the original keyboard.

Encryption. The encryption procedure is needed only for the keyboard commands that the client transmits, since sensitive user data, such as credit card numbers or passwords, are only inserted using the keyboard. RSA encryption was selected as it fulfils the demands of our specific environment.

Start-Up Phase. When both the client and the server start, some local initializations take place. The client then launches a connection request to the server which is advertised to the network neighbourhood through the UPnP module. The server accepts the new client generating a unique RSA public-private key combination.

Transfer of Encrypted Keyboard Input. The idea that lies beneath the communication command channel architecture is depicted in Figure 5.

Each end device consists of many possible input devices for interacting with the server. When the client program starts, it initiates the device discovery procedure, which may be offered either by a separate architectural module, for example, the device discovery module which uses UPnP. The next step of the procedure is to capture the input coming from the controllers. This is achieved by recording the key codes coming from the input devices. 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 after it has been suitably formatted adhering to a certain communication protocol. The encrypted message is transmitted to the server using the already existing socket connection.

Once the encrypted message has arrived at the server side, the server decrypts it using its private key, obtaining the initial keyboard commands that the client has captured.

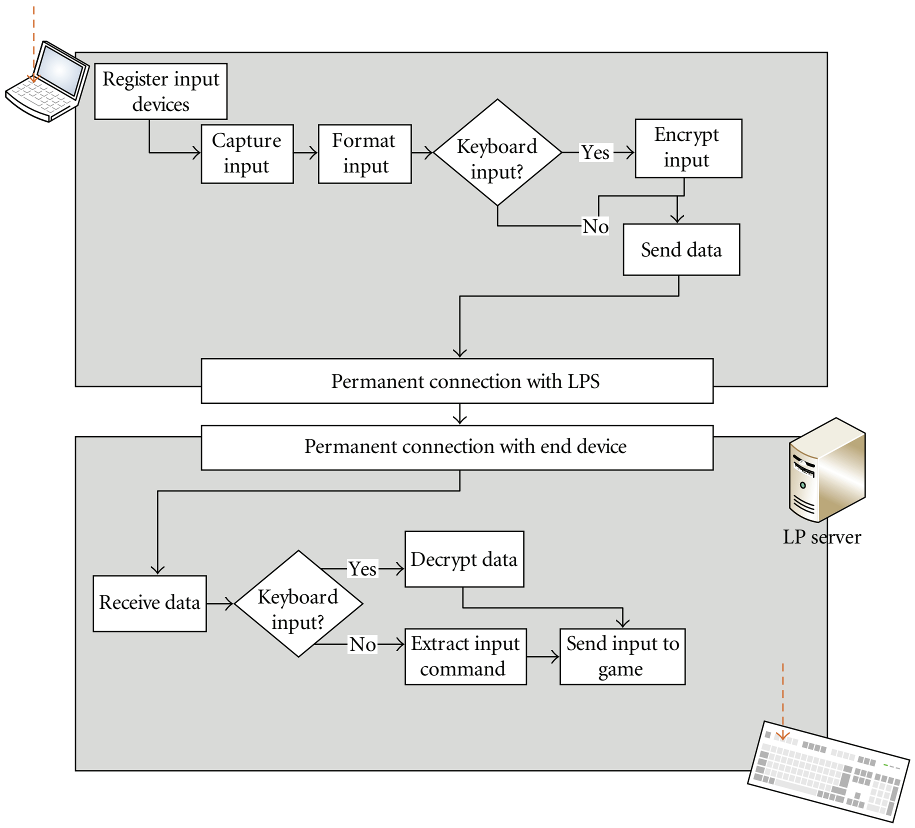


Figure 5: Encrypted command channel.

If the received message is not from a keyboard, the server bypasses the decryption stage, delivering the commands at the running game instance. The algorithm procedure of this step is described in the following sections.

4.5.3. Mouse. The server side of the return channel receives the mouse commands that originate from the client using the already open socket connection. The communication between the server and the client follows a specific protocol in order to be successfully recognized by the server and is exactly the same as the keyboard apart from the encryption part and the resolution part that follows.

An issue that arises when using the mouse input device is how the commands are executed correctly if the client has a different resolution to the server. This is because what is sent from the client to the server is the absolute mouse position. We realized that when a game is running on the client, 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, we observed that the matching of the resolutions should not be done with the resolution of the screen but again with the resolution of the game running on the server because every command is injected into the game window. The mouse positions have to be normalized on the client and the server side.

4.5.4. Joypad/Other. The server side of the return channel receives mouse and keyboard commands that originate from the client's Joypad/Other input via the already open socket connection. This means that any Joypad/Other input is firstly translated into suitable keyboard and mouse commands on the client side (using XML mapping files) and it is then transmitted to the server for execution at the game instance. The execution of these commands falls to the previously described cases.

4.6. Quality of Service Optimized Transmission. The Games@Large gaming architecture is based on streaming a game's 3D or video output to the client running on a separate device. This kind of distributed operation sets high requirements for the network in terms of bit rate and latency. A game stream with sufficient quality is targeted to require a bit rate of several megabits per second and the latencies have to be minimized to maximize the gaming quality. The same network which is used for gaming is also assumed to be available to other applications such as web surfing or file downloading. If the network did not have any kind of QoS support, these competing applications would have a negative effect on the gaming experience. Thus, the network has to implement QoS to satisfy the requirements of gaming regardless of other applications using the same network.

As presented in Figure 1, the network connection to the game client can be wireless. This is a further challenge for providing QoS for the gaming application. Our platform is based on IEEE 802.11 standard family [18] wireless LAN (WLAN) technologies. Currently, the most used WLAN technology is IEEE 802.11g which could provide the bandwidth needed for four simultaneous game sessions in good conditions. The near future IEEE 802.11n will enhance the maximum bit rate, but still shares the same basic medium access (MAC) method which does not support QoS. Priority-based QoS can be supported in IEEE WLANs with the Wi-Fi Multimedia (WMM) extensions [19] specified by Wi-Fi Alliance. WMM is a subset of IEEE 802.11e standard [20] and divides the network traffic into four access categories which receive different priority 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. Our platform is based on IEEE 802.11 (either g or n) and WMM. As presented later in the results section, WMM can be used to enhance the gaming experience substantially compared to the case of basic WLAN MAC.

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 [21]. 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.

The experimental results presented later in this paper prove that our standard-based solution enhances the game experience and gives superior performance compared to reference system without QoS support.

4.7. UPnP Device Discovery. To ensure easy system setup and operation as well as flexibility in dynamic home networks, various system components need to find each other automatically and be able to exchange information about their capabilities.

In the Games@Large system, we have selected to use the UPnP Forum [22] defined technologies for this functionality.

UPnP technology defines architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. The technologies leveraged in the UPnP architecture include common internet protocols such as IP, TCP, UDP, HTTP and XML [23].

The required functionality of device discovery is to allow a Games@Large client to find Games@Large servers in the network it is connected to. Device discovery is also available in servers to find other servers in the larger Games@Large network. For example, in a large system a Local Management Server (LMS) needs to find all LSSs in the network; in the home version, the logical servers are usually located in a single PC, but in an enterprise version, such as a hotel environment, there might be several physical server machines.

The device discovery component is also able to find information about services provided by the found devices. In the discovery phase the devices are also exchanging capability information, for example, an end device will inform the server of its capabilities, like screen resolution, connected input devices and so on. Servers can also advertise their capabilities to other servers and end devices.

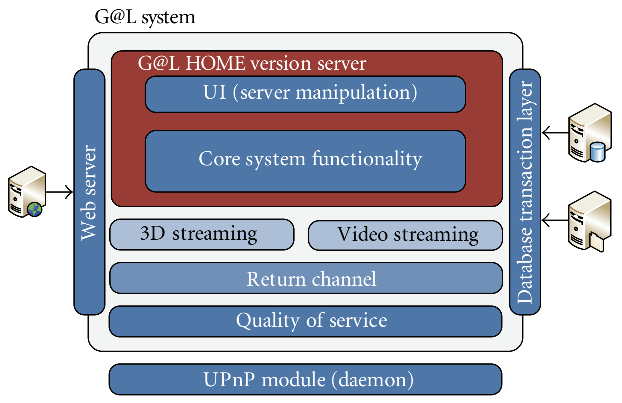


Figure 6: General server architecture.

4.8. System Integration. The local servers of Game@Large consist of three separate servers: LPS (Local Processing Server), LMS (Local Management Server), and LSS (Local Storage Server). In the (intended for the use in home environment) version, the main server of the system, is the Local Processing Server and at this stage it has (virtually) the core functionality which includes LPS, LMS, and LSS.

4.8.1. Local Processing Server. The "virtual" Local Processing Server is the core of the Games@Large System HOME version. It handles every communication with the clients while being responsible for every internal communication in parallel. The following Figure 6 represents the general server architecture.

At this stage of the implementation everything is manipulated within the server application. This web server is an Apache [24] server with support of PHP [25] and sqLITE [26] (as a PHP module) which is the database used in the HOME version of the system.

The LPS incorporates the implementations of 3D and Video Streaming, the Return Channel and the Quality of Service modules. In parallel it has a Web Server for serving the Web UI (user interface) to the clients and a Database Transaction Layer for the communication with the Database and the File System (game installations).

The basic procedure of the Processing Server is depicted in Figure 7.

When a client wants to connect to the system, it tries to locate the LPS that is running the G@L HOME system. The UPnP daemon that runs on the LPS "helps" each end device to locate the server's IP. The application that runs on each client launches a web browser with the given IP address and the LPS's Web Server starts interacting with the clients. The client is served with the corresponding web UI (different UI for each end device). The server is informed which UI has to be sent by a parameter that is passed together with the IP of the server in the web browser.

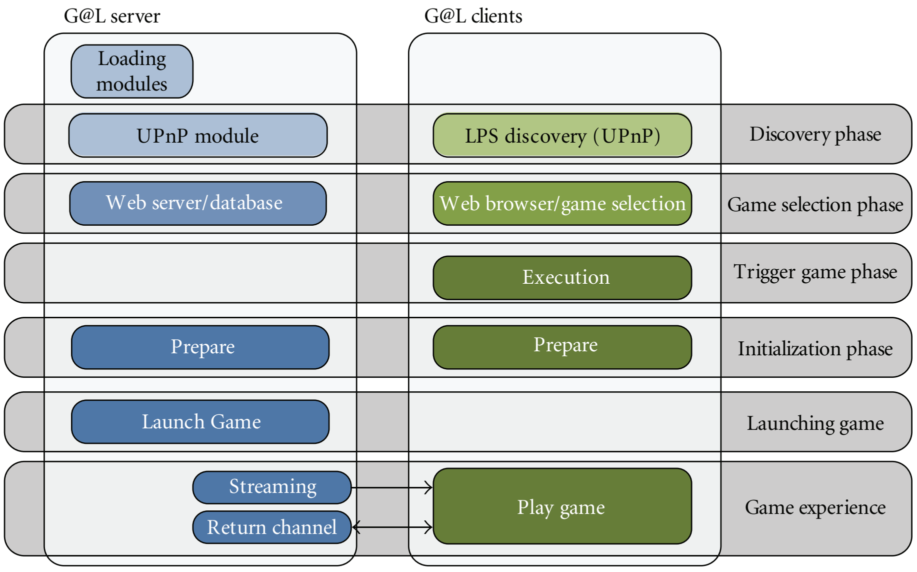


Figure 7: General flow of information.

After the log-in procedure of the end user, the game selection phase is launched. When the user selects a game to play the main client application is launched and the main communication procedures between the client and the server begin. The client is informing the LPS about its request to play a specific game. The LPS is processing the client's command and more specifically it starts the decision procedure.

During the decision procedure the server, with the help of the UPnP and QoS modules, observes the current system status and network utilization. If the game's Software, Hardware, and Network demands are met, then the game initialization procedure begins. The client is also informed that the launching of the game is imminent and thus it will be able to begin its initialization procedure. After the successful finishing of the initialization procedure, the game is launched with the 3D commands or video of the game streamed to the client. Additionally, the client is streaming the user's input commands to the server. The commands coming from the client are furthermore processed on the server side and they are delegated to the window of the game.

5. Experimental Results

In order to demonstrate multiple game execution and system performance analysis we designed a testbed [27] in which we could monitor the performance of network, devices and Games@Large system processes while running simultaneous game sessions. Figure 8 shows our testbed setup.

We performed our experiments with two client notebooks of which one was running Sprill (Casual game) and the second one Red Faction Demo (first person shooter game). The Games@Large server (Intel 2 GHz 2 CPUs, 2048 MB RAM, 256 MB dedicated video memory) running Windows XP was connected to a 100 Mbps switch which

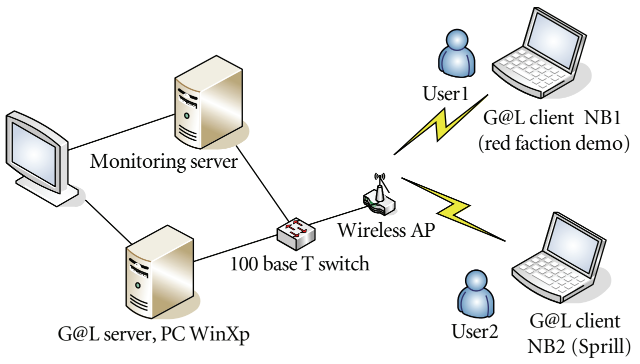


Figure 8: Games@Large testbed.

in turn connected to the WLAN Access Point (AP) via a wired Ethernet connection. The two client notebooks (NB1: Intel 2 GHz 2 CPUs, 1024 MB RAM, 384 MB shared video memory and NB2: AMD Athlon 2.1 GHz CPU, 1024 MB RAM, 256 MB dedicated video memory) were connected to the Wireless AP via the IEEE 802.11g wireless connection. For system performance monitoring we used an external Monitoring PC. All the PCs and NBs were SNMP/WMI enabled for performance monitoring purposes.

We used the PRTG Network Monitor [28] on the Monitoring PC to monitor network, device, and Games@Large processes with minimal influence on system's performance. Additionally we used FRAPS [29] to measure the games' frame rate.

The test scenario included a full system workflow which consisted of the following steps, shown also in Figure 7: G@L server discovery from the client device, web user interface access and game list browsing, selection of the game, and starting to play, described in detail in Section 4.8.1. Both the test participants were familiar with the 2 games used for tests.

Table 1: Frame rate per second for tested games run natively and on G@L system.

Mode	Game	Mean FPS	Std Dev
Run Natively (Server PC)	Red Faction	59.23	5.16
	Sprill	349.14	87.65
Run on G@L	Red Faction (NB1)	18.26	12.12
	Sprill (NB2)	125.27	108.59

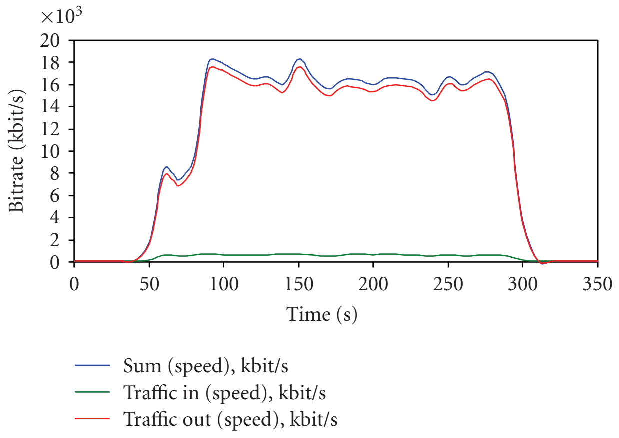


Figure 9: Bitrate versus Time for G@L Server.

We did not perform extensive user studies though we were recording user observations and perceptions about the gaming experience. Both participants commented that at the beginning there were some pauses in the game while it was loading, but after a while they disappeared. The gaming experience in the Mean Opinion Score (MOS) scale [30] was rated between 4 and 5. There were some differences with the original game play but participants were not frustrated and could enjoy the game play.

During the test sessions we were logging the frame rate of the games on the client devices, network, and G@L processes performance on the client and server. For analysis purposes, we ran both games natively on the server PC and measured their performance and frame rate in frames per second (FPS). Measurement results for native and G@L system game executions are presented in Tables 1 and 2.

Network usage during tests was measured on the server and both clients. Figure 9 shows the network usage bitrates for the G@L server simultaneously serving two clients. The mean sum bitrate for the clients is 6956.04 kbit/s for Red Faction game on NB1 and 8627 kbit/s for Sprill game on NB2, respectively.

The average bandwidth usage on the client (as well as on the server per single game) devices is correlated with the FPS. From mean bit rate of NB1 and NB2, and Table 1 we can see that when the frame rate is high, the network utilization is higher (Red Faction versus Sprill). The same correlation can be observed between the frame rate, CPU and memory utilization on the client and server. According to some proof of concept tests Windows based clients are running the games at higher frame rates than the Linux-based ones and those with the weak hardware capabilities.

The bandwidth that the game running on the LPS requires is directly proportional to the frame rate of the game. Clients that are capable of running games at high frame rates will spend a large portion of their time reading 3D data from the socket. After a frame has been read, an ACK is sent to the server so it can generate a fresh updated frame. When the frame-rate is above sufficient (20–25 FPS is enough for a good gaming experience), it can be artificially limited by the LPS to save the network resources. In such a way, it is possible for multiple (four good quality concurrent sessions per 1 AP/LPS) devices to be connected to the same LPS without overloading the network.

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 G@L 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, see Table 2 (in spite the streaming and the compression). As long as the processing server has sufficient CPU and memory resources to run multiple games at once it can run them. Since the games' graphics are not rendered on the LPS (when 3D streaming), there is no competition between games for the GPU and neither for the full-screen mode.

The most important hardware requirement for the client device is the video adapter. 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 cashing. So memory requirements for the client should be 200–300 MB available to the client application for fairly heavy games.

Besides the frame rate and technical requirements, such as hardware and network bandwidth, for a game to run playable on the end device in the Games@Large system it has to be compatible with the end device screen size and controller capabilities (e.g., some games cannot be played on small displays, other games cannot be controlled with the gamepad).

The above mentioned tests were performed over a Wi-Fi network without the QoS and with no other traffic (except the SNMP/WMI packets for system monitoring, but these crate a very small network load) present on the network than the one produced by the two game sessions of the client notebooks. Therefore latencies and other negative traffic effects did not assert during the tests, for example, measured mean round trip time (we sent an SNMP Ping of 30 Bytes from the server, every second 30 times) for both clients was <2 ms.

The solution for QoS optimized transmission described in Section 4.6 was evaluated by performing a series of tests in a laboratory environment. The experimental setup described

Table 2: G@L Server and Client process performance: Memory and CPU usage.

Mode	Game	Mean working set (Mbyte)/Std Dev	Mean CPU usage (%)/Std Dev
Run Natively (Server PC)	Red Faction	61.03/4.19	49.03%/1.75%
	Sprill	103.39/13.14	47.10%/7.68%
Run on G@L	Red Faction (NB1 process)	45.19/7.11	21.58%/7.03%
	Red Faction (Server process)	66.46/12.29	22.00%/8.11%
	Sprill (NB2 process)	112.15/30.36	67.69%/18.30%
	Sprill (Server process)	139.42/41.15	31.65%/10.67%

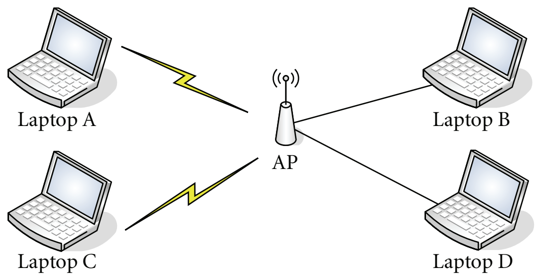


Figure 10: Test setup.

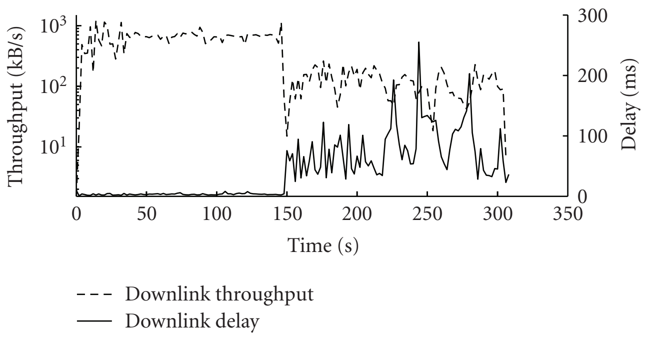


Figure 11: Downlink throughput and delay when both the game and the background traffic share the same priority.

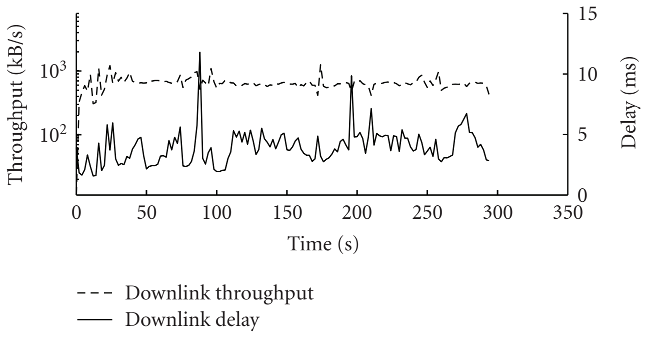


Figure 12: Downlink throughput and delay when the game has higher priority than the background traffic.

in Figure 10 includes four laptops and a WMM enabled WLAN access point (AP). Laptop A was used as a game client while laptop B was running the game server software. The game server was connected with a wired Ethernet connection to the WLAN and the client connection was wireless. In addition to game-related laptops, there were two additional laptops, C and D, which were used to generate background traffic to the network when testing the QoS capabilities of the solution. Similar to the game laptops, Laptop C was connected using a wireless connection and Laptop D with a wired connection. The laptops used were standard PC laptops equipped with IEEE 802.11g and WMM enabled wireless interfaces or 100 Mbps Ethernet interfaces. The AP was a normal WLAN AP with an addition of priority queuing in the AP kernel buffers in case of WMM queue overflow.

In each of the test cases, playing the game (Sprill, using 3D streaming) was begun in a wireless network without any additional traffic. From the middle until the end of the test, competing traffic stream was introduced from Laptop D to Laptop C. This stream was generated by an open source traffic generator iPerf. A single TCP session was used, thus simulating a file download between D and C using FTP or HTTP. The tests were performed with two QoS configurations. In the first one, all the traffic was sent using best effort priority, and in the second, the game traffic was using voice priority and the background best effort priority. Downlink and uplink throughput, delay, jitter, and packet losses for the gaming traffic were recorded using the QoSMeT tool [31] and the game client's realized frame rate was measured with Fraps [29].

The downlink performance is visualized in Figure 11 in terms of throughput and delay for the case with equal priority, and in Figure 12 for the case where gaming has higher priority. The effect of introducing the background traffic can be seen very clearly in Figure 11 while it is not visible in Figure 12.

The complete results are presented in Tables 3 and 4 for both cases respectively. In the case without prioritization, the game really suffers from the competing traffic in the WLAN. The downlink delay increases up to around 20 times as high as in uncongested conditions. This causes the realized frame rate at the client to decrease almost 90 percent which, together with the increased delay, practically destroys the game experience. When the game traffic is classified with a priority higher than the background traffic, the effect of competition is negligible. The downlink delay remains an acceptable level and the realized frame rate of the game at the client decreases only less than 10 percent.

Table 3: Average values when both the game and the background traffic share the same priority.

	Downlink through-put (kBps)	Uplink through-put (kBps)	Downlink delay (ms)	Uplink delay (ms)	Downlink jitter (ms)	Uplink jitter (ms)	Downlink packet loss (%)	Uplink packet loss (%)	Realized frame rate (fps)
Without competing traffic	651.3	26.6	3.4	1.7	1.0	0.8	0.020	0.009	82.8
With competing traffic	119.3	4.6	66.0	6.4	5.8	2.5	0.630	0.059	9.1
Ratio	0.18	0.17	19.60	3.67	5.73	3.24	31.16	6.65	0.11

Table 4: Average values when the game has higher priority than the background traffic.

	Downlink through-put (kBps)	Uplink through-put (kBps)	Downlink delay (ms)	Uplink delay (ms)	Downlink jitter (ms)	Uplink jitter (ms)	Downlink packet loss (%)	Uplink packet loss (%)	Realized frame rate (fps)
Without competing traffic	665.8	28.1	3.5	1.8	1.0	0.7	0	0.002	80.9
With competing traffic	624.4	25.7	4.2	2.2	1.2	0.8	0.003	0	73.5
Ratio	0.94	0.91	1.19	1.23	1.13	1.19	—	0	0.91

6. Conclusions

In this paper, we have presented a new distributed gaming platform for cross-platform video game delivery. An innovative architecture, transparent to legacy game code, allows distribution of a cross-platform gaming and entertainment on a variety of low-cost networked devices that are not able to run such games. This framework enables easy access to the game catalogue via the web based interface adapted for different end devices. A generalized protocol supports end devices with both OpenGL and DirectX API's. We have shown that it is feasible to use a single PC for multiple game executions and stream them with a high visual quality to concurrently connected clients via a wireless network using the QoS solution. The developed technology enables putting PC gaming onto TV screens which is a rapidly emerging trend in gaming market. Apart from that it also enables a pervasive video game access on handheld devices.

Future work is to support wider range of titles, we will need to implement the interception layer for all the graphic libraries used by the games which can supported by Game@Large. A possibility is investigated to use or develop a transport protocol (e.g., RTP), which could replace TCP for 3D streaming for the improvement of its performance over a wireless network. For video streaming current research is focused on reducing the computational complexity of the H.264 encoder itself by incorporating enhancements based on the available rendering context information using the motion prediction and by predicting the macroblock partitioning. In parallel, we will run extensive laboratory tests and field trials in the home environment in order to gather knowledge about users' perceptions and investigate the subjective expectations of gamers.

Acknowledgments

The work presented in this paper has been developed with the support of the European Integrated Project Games@Large (Contract IST-038453) which is partially funded by the European Commission.

References

[1] Y. Tzruya, A. Shani, F. Bellotti, and A. Jurgelionis, "Games@Large—a new platform for ubiquitous gaming and multimedia," in *Proceedings of the Broadband Europe Conference (BBEurope '06)*, Geneva, Switzerland, December 2006.

[2] S. Cacciaguerra and G. D'Angelo, "The playing session: enhanced playability for mobile gamers in massive metaverses," *International Journal of Computer Games Technology*, vol. 2008, Article ID 642314, 9 pages, 2008.

[3] I. Nave, H. David, A. Shani, A. Laikari, P. Eisert, and P. Fechteler, "Games@Large graphics streaming architecture," in *Proceedings of the 12th Annual IEEE International Symposium on Consumer Electronics (ISCE '08)*, pp. 1–4, Algarve, Portugal, April 2008.

[4] D. De Winter, P. Simoens, L. Deboosere, et al., "A hybrid thin-client protocol for multimedia streaming and interactive gaming applications," in *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '06)*, Newport, RI, USA, May 2006.

[5] L. Cheng, A. Bhushan, R. Pajarola, and M. El Zarki, “Real-time 3d graphics streaming using mpeg-4,” in *Proceedings of the IEEE/ACM Workshop on Broadband Wireless Services and Applications (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] S. Stegmaier, M. Magallón, and T. Ertl, “A generic solution for hardware accelerated remote visualization,” in *Proceedings of the Symposium on Data Visualisation (VISSYM ’02)*, pp. 87–94, Barcelona, Spain, May 2002.

[8] MPEG-4 AVC, “Advanced video coding for generic audiovisual services,” ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.

[9] MPEG-4 HE-AAC, “ISO/IEC 14496-3:2005/Amd.2”.

[10] P. Eisert and P. Fechteler, “Low delay streaming of computer graphics,” in *Proceedings of the International Conference on Image Processing (ICIP ’08)*, pp. 2704–2707, San Diego, Calif, USA, October 2008.

[11] D. Marpe, H. Schwarz, and T. Wiegand, “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, 2003.

[12] RFC 3550, “RTP: A Transport Protocol for Real-Time Applications”.

[13] RFC 3984, “RTP Payload Format for H.264 Video”.

[14] RFC 3640, “RTP Payload Format for Transport of MPEG-4 Elementary Streams”.

[15] D. L. Mills, “Network time protocol version 4 reference and implementation guide,” Tech. Rep. 06-6-1, Department of Electrical and Computer Engineering, University of Delaware, Newark, Del, USA, June 2006.

[16] Hawk Software, Hawk Network Library, http://www.hawksoft.com/hawknl.

[17] J. Nagle, “Congestion control in IP/TCP internetworks,” RFC 896, January 1984.

[18] IEEE Standard 802.11-1999, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” 1999.

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

[20] IEEE 802.11e-2005, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements,” 2005.

[21] UPnP QoS Architecture V2.0, http://www.upnp.org/specs/qos/UPnP-qos-Architecture-v2-20061016.pdf.

[22] UPnP Forum, http://www.upnp.org.

[23] UPnP device architecture, http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf.

[24] Apache Software Foundation, Apache HTTP Server, http://httpd.apache.org.

[25] PHP: HyperText Preprocessor, http://www.php.net.

[26] SQLite, http://www.sqlite.org.

[27] A. Jurgelionis, F. Bellotti, A. Possani, and A. De Gloria, “Designing enjoyable entertainment products,” in *Proceedings of the Conference on Human Factors in Computing Systems (CHI ’08)*, pp. 1–5, Florence, Italy, April 2008.

[28] PRTG Network Monitor, http://www.paessler.com.

[29] Fraps, “Real-time video capture benchmarking,” http://www.fraps.com.

[30] Ch. Schaefer, Th. Enderes, H. Ritter, and 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, April 2002.

[31] J. Prokkola, M. Hanski, M. Jurvansuu, and M. Immonen, “Measuring WCDMA and HSDPA delay characteristics with QoSMeT,” in *Proceedings of the IEEE International Conference on Communications (ICC ’07)*, pp. 492–498, Glasgow, UK, June 2007.