Networking Aspects for Gaming Systems

Christos Bouras, Vassilis Poulopoulos, Ioannis Sengounis and Vassilis Tsogkas
Research Academic Computer Technology Institute,
N. Kazantzaki, Panepistimioupoli Patras, Greece
bouras@cti.gr, poulop@cti.gr,
jns@sch.gr, tsogkas@cti.gr

Abstract

As the evolution of computer technology introduces new advances in networks among others, online gaming becomes a new trend. Following the trends of our era, Games At Large IST Project introduces an innovative platform for running interactive, rich content multimedia applications over a Wireless Local Area Network. Games at Large project's vision is to provide a new system architecture for Interactive Multimedia which will enhance existing CE devices such as, Set Top Boxes (STB), Small Screen and other devices, which are lacking both the CPU power and the graphical performance to provide a rich user experience. This paper presents the controllers' sub-system of the innovative mechanism that will be implemented in the context of Games at Large project. More specifically, it presents the general architecture of the complete system and focuses on the "capturing" and "execution of commands" modules at the different clients and the remote execution of the commands on server side. The client software captures the input from the different input devices, sends the commands over a Wireless Local Area Network and the server is responsible for receiving and executing the commands to the corresponding application.

Index Terms — *remote control, online gaming, remote command execution, input device capturing*

1. Introduction

Computer games constitute nowadays one of the most dynamic and fastest changing technological area, both in terms of market evolution and technology development. In this area, as the computer games are evolving and online activities and gaming become parts of our life, the need of interaction within a client – server architecture becomes very intense. The successful paradigms of online gaming such as WoW [2], Half Life [3] and Second Life [4] are only just the beginning of a new era for the online games. The idea that lies behind online gaming is that a game that can be played by multiple users should not have only a local context. A client with the basic game software is installed on the client machine while multiple servers are assigned with the task of interconnecting all the possible users to what is called the "world" or the scenario of the game. Games at Large project goes one step further than the classical procedure of online gaming and the main intention is to enhance the idea of application on demand [1] in order not only to support games on demand, but also to enable devices that lack the physical power to load a game, to run games [7].

The main idea is that one or more powerful servers will actually execute the game for the client and the client will be presented only with the screens of the game and not the game loader or the execution of complex graphics. On the other hand the basic aspect of a game is the interaction with the end user (gamer). This means that apart from only presenting the game to the user (through this client – server architecture) the system must be able to capture the input from the input devices of the end user and transfer them to the server in order to represent the interaction that is done on a physical level when playing a game.

In this document we present a mechanism for transferring input commands from any device, acting as the client, to execution commands at the corresponding program - game of the server. The purpose of this mechanism is to be able to control a program that runs on the centralized server from a remote operating system. This mechanism is created within the scope of the Games at Large project. Meeting the demand of highly interactive multimedia systems with low cost end devices (CE), requires a radical change in the system's architecture. Games@Large project intends to design a platform for running interactive rich content multimedia applications. Games@Large vision is to provide a novel system architecture for Interactive Multimedia which will enhance existing CE devices such as, Set

Top Boxes (STB) and other devices which are lacking both the CPU power and the graphical performance, to provide a rich user gaming experience.

The future home is an always-on connected digital home. By 2010, there will be over 420 million broadband households worldwide [5],[6]. With the standard set for super-high speed, always-on connection, the way people view entertainment has fundamentally changed and created new standards for consumption. Consumers no longer expect their internet access to be only from a desktop PC - now they want it through the TV in their living room or in the palm of their hand, inside the house and on the go. The presented scenario [8] bundles video gaming capabilities into consumer electronics devices, such as Set-Top Boxes (STBs), Digital Video Recorders (DVRs), home entertainment systems, TVs, handhelds and other devices that are not considered, today, as real gaming devices since they lack the necessary CPU power and graphical performance. In this paper we present a new system for pervasive gaming and multimedia, which is being developed under the EU FP6 project, Games@Large (GaL) [1]. The paper is dedicated to the design testing concept elaboration, in order to base the approach for the development of evaluation and testing methodologies. Testing and verification process is part of the iterative, spiral-life workflow model (user-centered design and incremental improvement based on feedback from user and expert evaluation of prototypes).

In this paper we present the general architecture of the sub-system that controls the input of the devices and their server side execution. More specifically, we examine how, input is able to be captured by any input device on the different end devices and on different operating systems, how commands are sent over the network and finally, how commands are executed at the target software of the server.

The rest of the paper is structured as follows: the next section describes the vision and goal of the Games at Large project. Section 3 describes the general architecture of the system and the architecture on each device (the end device and the server). Section 4 describes the command channel infrastructure which is the main scope of this manuscript while section 5 describes the general Server Side infrastructure and section 6 the client side infrastructure on the different end devices and on different operating systems. The paper finalizes with general remarks and future work to be done on both server side and client side.

2. Games at Large Project

Games at Large (Games@Large) being an Integrated Project (IP) intends to research, develop and implement a new architecture to provide users with a richer variety of entertainment experience in their entire houses, hotel rooms, cruise ships and Internet Cafés, incorporating unprecedented ubiquitous game-play. The project evolved from the home environment to other local Focus Areas (FA) regarding the benefits such FA may gain based on the unique technology approach of Games@Large. The Integrated Project includes activities of TV Multimedia and Gaming using Enhanced Media Extender, Local Processing and Storage Server(s), Handheld Devices and Local Wireless Network. Games@Large intends to enhance the existing Digital Living Network Alliance (DLNA) and the UPnP Forum standards by introducing the unique set of features required for running games over a local network, like all other media and content types (video, audio).

Market interest is now revolving around capitalizing on the rapid increase of always-on broadband connectivity, which becomes ubiquitous. Broadband connection drives to a new, digital, “Future Home” as part of a communications revolution, which will affect every aspect of consumers’ lives, not the 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 and new types of content and services.

Games offer a leisure time activity for every member of the household – from avid gamers to kids, as well as allowing whole families to play together. Games offer also leisure time activity for guests in hotels and visitors in Internet Cafes. Games@Large offers ubiquitous accessibility for all members of the household on all desired entertainment devices. The project focuses on new innovative ideas such as multiple-game execution on the Games Gateway and delivery of graphics-rendering meta-data over the home network via low latency, low bandwidth Pre-Rendering Protocol to achieve low-cost implementation of ubiquitous game play throughout the house, while taking advantage of existing hardware, and providing multiple members of the family with the ability to play simultaneously.

Games@Large intends to enable the Games to diversify from dedicated appliances and a single corner of the house, to any place at home such as, the TV in the living room, the handheld device or any other device with the relevant screen, controls and connectivity. The project will also provide the required infrastructure for running games on the hotel guest

room TV or on small screens for people sitting in Internet Cafés, cruse ships, trains or airplanes.

The technological challenges of the Games@Large project are:

- Distributed computing and storage
- Video/Image/Graphics delivery with very low latency through a wired/wireless home network
- Adaptation of PC screen-images to TV screen and handheld devices
- Integration of wireless users' game control devices
- Translation of user ergonomics to different devices and form factors
- Research of new class of Media Extenders for games
- Enhancement of STBs to support video games
- Development of new methods for QoS linking Consumer prospective with system measurements
- Enhancement of relevant industry standards for time critical multimedia content while maximizing Users Experience

The Games@Large mission is to develop a new method for ubiquitous video games through unique technology to transfer graphical data while reducing latency and ensuring QoS in a cost-effective manner. Main focus will be given on studying and supporting the use of video games within four different focus areas: User's home, Hotels, Internet Café, and Elderly Houses. A multi-layer approach will cut horizontally across the Games@Large focus areas, aiming to assess the conditions under which a Games@Large platform may frame within and improve the state of the art of each business domain, through performing the following, logically consecutive activities: collecting user requirements, researching and developing common Technologies, implementing and integrating those technologies within the required Servers and prototype CE Devices, running technology verification and Training and evaluating all results.

3. System Architecture

The next figure depicts the general system architecture. As it is obvious, the system consists of two different "levels". The first level includes all the possible servers that will be used for the system, while the second level includes the connection of the different end devices of the system. The server side constitutes of multiple different servers that are assigned with the task of serving the games and require a very quick and stable communication between them while the second level is the interconnection of any possible end device with the server in order to communicate and interact in order to load and play a game. While on the architecture that is described all the servers can communicate to one another, the end devices can "see" only one server which is the main serving and processing server for the games.

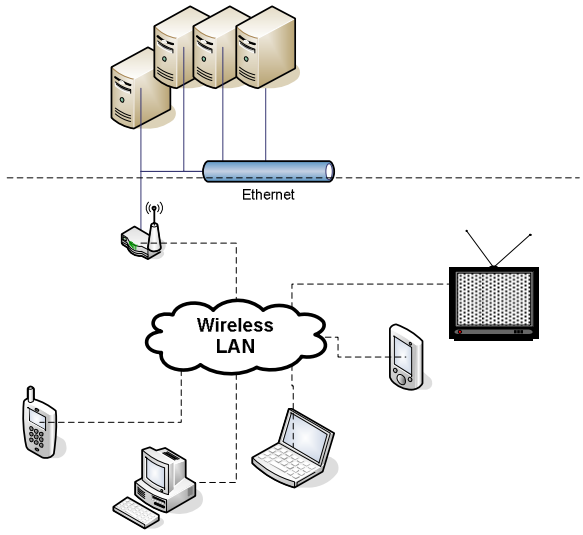

Figure. 1 General System Architecture

The server side of the system is assigned with the task of executing the games and sending the corresponding scenario of the system to the possible clients. The clients are sending feedback to the server which is the commands that are executed for the game. Thus, the server should be able to have at least two communication channels with each client: one for sending the game "pictures" and one for receiving the commands of the game. In this paper we will focalize on the command communication channel which is a direct connection from each client to the server.

4. Command Channel Infrastructure

The idea that lies beneath the communication command channel architecture is depicted on the following flow diagram. Each end device consists of many possible input devices that enable the user to interact with the device and thus enable the user to interact with the game that is played on the end device.

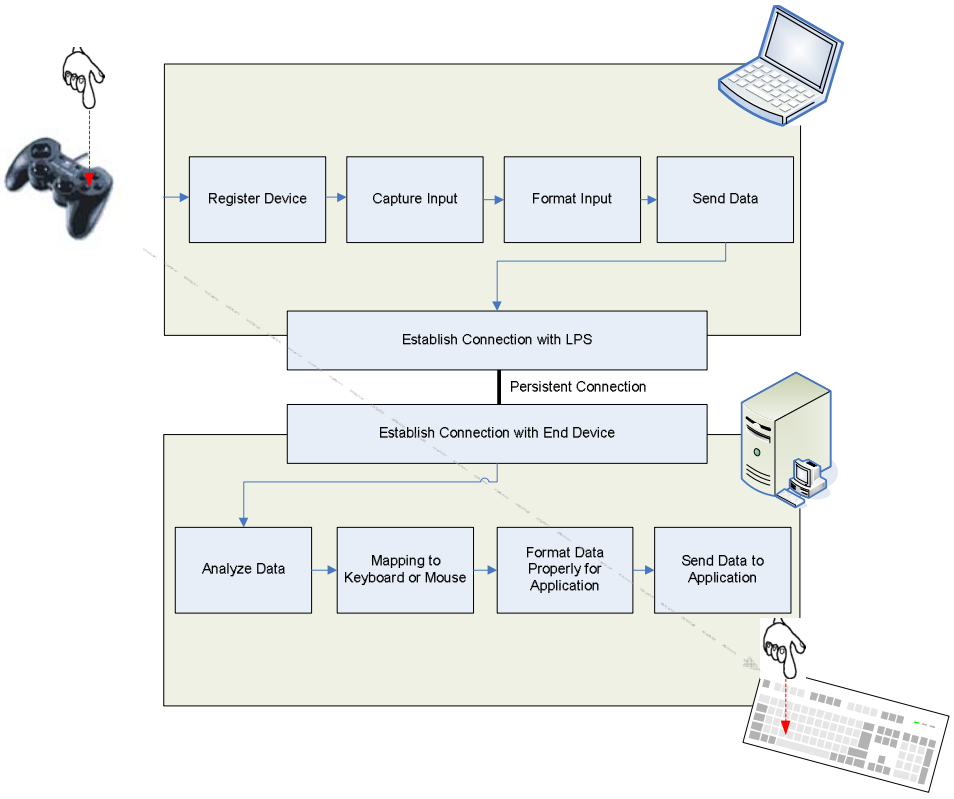

Figure. 1: Communication Channel Architecture.

When the client program starts, it initiates the device discovery procedure, which may be offered either by a separate architectural module, e.g. the device discovery module which uses UPnP, or by a system call causing the discovery for input devices attached to the system. It is essential afterwards, that the results of the device discovery are registered in our program so that we are aware of the existing input devices marking out several other non-existing.

The next step of the procedure is to capture the input coming from the input controllers. This is achieved by recording the key codes coming 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. The previous means that whenever an input event is caused by a keyboard or a mouse, an interrupt message is sent to the message queue of our program; then it is translated and finally recorded. However, the polling case of joysticks or joy-pads means that these devices have to be polled by a program's thread in order to sense motion or button presses. The polling period has to be small enough to capture any input, but not too small to monopolize the system's CPU. A period of 10ms seems to be a wise trade off.

After an input key code has been captured, the transmission of it takes place. This is achieved using an already open socket connection with the server side. Data is transmitted through the socket in the form of a string with a certain communication protocol.

The socket connection can either be of TCP or UDP protocol. Since UDP emphasizes on real time, low latency transmission, it is preferable for this type of communication. Even if some key codes are lost in the process of transmitting them over the network, there is no real loss since there is a flow of key codes that can overcome this possible threat. However, in real life, error prune networks, such as WiFi's, the TCP protocol is preferred avoiding the possible game experience fall caused by lost controller's packets transmission.

Since the key codes have arrived at the server side, they are executed at the running game instance. At this point, there needs to be a distinction between the different types of transmitted key codes. There are basically four types of possible input device's data transmission. Commands may be coming from: (a) keyboard, (b) mouse, (c) joystick / joy-pad device or (d) any other HID input device.

In the first case, the server has to recognize the virtual key code, or the "pressed" / "released" event of a keyboard button, then do a possible mapping to some

other key code, based on the game and user profile, and finally deliver it to the active application window for execution.

In the case of mouse input, the server has to recognize the virtual key code or the "pressed" / "released" event of a mouse button, recognize any mouse wheel event or any mouse movement (absolute or relative), then do a possible mapping to some other key code, based on the game and user profile, and finally deliver the key code to the active application window for execution.

In the case of joystick/joy-pad input, the server recognizes the state of the joystick/joy-pad device, maps the state to the appropriate keystrokes using the xml mapping file of the particular game-joystick/joy-pad combination. In this way, we are able to emulate the joystick/joy-pad input using pure keystrokes– mouse movements that represent the actual behavior of the input device. Finally the key code is delivered to the active application window for execution.

For any other HID input device the system treats input similarly to the joystick/joy-pad. The only prerequisite is the existence of a mapping file in order to convert the commands to keyboard and mouse instructions

5. Server Infrastructure

As long as we are creating an environment with one server and multiple clients, it is essential to analyze how each end device will be able to capture all the commands from the input devices. This is because the unique server of the system should receive data that are sent over the network and execute the commands on the specific procedure that runs each game.

The "gateway" of the servers is the LPS (Local Processing Server). The main goal of Local Processing Server is to run multiple games simultaneously on the server, whereas each game runs in its own game environment and is streamed to an end-device. The game environment is an isolated and encapsulated "sandbox," providing the environment for game execution. The procedure, that makes the simultaneous running of multiple games possible, decouples the game execution from the game output, directed to display card/PC monitor, and all user-facing I/O, directed to the keyboard/mouse/HID.

This "sandbox" environment for the server is created dynamically according to: a) the current occupancy of the resources and the hardware requirements that the game sets on the server, b) the software requirements on the client side and c) the current network condition. In order to be able to run a game on the server, the system monitors in a periodical manner the hardware resources of the server and the network conditions (jitter, latency and bandwidth). Additionally, according to the end device specifications (hardware and software), the server decides on the manner that the game will be executed on the client side.

6. Client Side Infrastructure

The possible different clients of the G@L environment are: (a) a Laptop with Windows XP / Vista environment, (b) a Set-Top Box with either Linux or Windows CE and (c) an enhanced handheld device with either Windows CE or a Linux version for small screen devices.

Each client implementation should consist of the following components: (a) a device discovery module, (b) the game browser and game launcher modules, (c) authentication modules, (d) input capturing and command transferring modules and finally, (e) decoders in order to run the streamed game that is sent from the server.

The device discovery module is used to seek for an appropriate LPS to connect to and introduce itself to the LPS with the End Device capabilities. The G@L Game Browser, which queries the Games Service on the LPS for listing the available games to the user, enables the user to browse the list of available games and select one to launch. Personalization of the UI should be available to the user/provider for enabling different views for users (i.e. Browser skins). When the user selects a game and requests to launch it, the G@L Game Browser issues a Start Game request to the G@L Client Game Launcher. The Game Browser will show to the user only the list of games that can run on the End Device by filtering the list according to the capabilities of the End Device compared with each game requirement.

Authentication communicates with the G@L Game Browser to authenticate the user against the LPS authentication module that authenticates the user against the Management Server. The Client Game Launcher controls all modules on the client side. The Game Launcher communicates with the LPS discovered by the device discovery module. The capture controller captures the Human Input Device (HID) controls and transfers them to the Controller Emulator on the LPS via the network layer, using the Controller protocol.

6.1 Capturing commands in Windows OS

As already mentioned, the end devices of the system can be multiple and thus they may use various

operating systems, whereas the server is based on Windows operating system. When the client utilizes Windows operating system, the implementation of the modules, and more specifically, the command capturing module, is implemented as a generic driver. This driver is able to recognize any input device and transform the command from them to keyboard and mouse commands, according to mapping files that are utilized for this scope. The aforementioned is a windows application, included in the game launcher of the client, which is called reverse channel module. The main assignments of the reverse channel module is a) to ensure that the connection to the server is established successfully, b) to capture commands from any input device, and c) to send the commands over the channel that is present between the client and the server.

6.2 Capturing commands in Linux OS

When the end device utilizes Linux operating system, there is no need for a low level driver (as in the Windows case) to be implemented as a client-side program in order to capture the input devices' input. On the contrary, the command capturing is feasible through the evdev generic input driver and the evbug capturing implementation, both of which are available to any modern Linux kernel. In this way, the generated devices are manipulated as "files", and thus it is possible to create multiple "hooks" for each device in order to capture any input originating from them. Finally, the keycodes from any input device are translated to keyboard and mouse commands and are then transmitted to the server for execution in the game instance.

7. General Remarks and Future Work

In this paper we have described the command execution channel of the Games at Large project, an IP project with the vision to research, develop and implement a new architecture to provide users with a richer variety of entertainment experience in their entire houses, hotel rooms, cruise ships and Internet Cafés, incorporating unprecedented ubiquitous game-play. Running already the second year of the project, we have managed to interconnect the clients with the servers and we are able to remotely execute commands to the server that are sent from a variety of different clients.

As the system is implemented, more and more features are included on the release versions, such as modules that enable encryption of the commands and modules that utilize network specific characteristics in order to adapt on the possible network environment.

Additionally, efforts are made towards the direction of creating software for every possible operating system in order to enable more end-devices to be connected to the Games at Large Environment.

REFERENCES

[1] Games at Large project's official website, http://www.gamesatlarge.eu
[2] World of Warcraft official website, www.worldofwarcraft.com
[3] Half Life official website, http://orange.half-life2.com/
[4] Second Life official website, http://www.secondlife.com
[5] By 2010, One-Third of the Predicted 422m Broadband Households will be Able to Receive IPTV. http://www.findarticles.com/p/articles/mi_m0EIN/is_2006_Sept_26/ai_n16837715
[6] Worldwide online access. http://www.emarketer.com/Report.aspx?bband_world_jun06&src=report_summary_reportsell
[7] Y. Tzruya, A. Shani, F. Bellotti, A. Jurgelionis, Games@Large - a new platform for ubiquitous gaming, BroadBand Europe 2006, Geneva, Switzerland, November 2006
[8] P. Casas, D. Guerra, I. Irigaray, User Perceived Quality of Service in Multimedia Networks: a Software Implementation, Joint Research Group of the Electrical Engineering and Mathematics and Statistics Departments, 2006
[9] L. Mathy, C. Edwards, D. Hutchison, Principles of QoS in group communications, Journal Article, Telecommunication Systems, Springer Netherlands, 2004, pp. 59-84
[10] Software Quality Assurance & Usability Testing, http://members.tripod.com/bazman/usability.html?button6=Article+on+Usability+Testing