Multimedia Systems –
C-WG3#1
Requirement Specification

PURPOSE OF THIS DOCUMENT

This document contains all the user requirements and the overall definition of the product

REVISION HISTORY

Revision #	Date	Author(s)	Comments
0.1	12.11.09	Pasi Keski-Korsu	Template created
0.2	12.11.09	Pasi Keski-Korsu, Antti Väyrynen	Diagrams added, chapters 2 and 3 done
0.3	17.11.09	Pasi Keski-Korsu	Chapter 4 done
0.4	26.11.09	Lauri Majamaa	Revision, partial rewrite
0.5	27.11.09	Pasi Keski-Korsu	Small fixes to ch. 4
0.6	27.11.09	Lauri Majamaa	Small fixes
1.0	28.11.09	Lauri Majamaa	Final version

ABBREVIATIONS

UDP	User Datagram Protocol
RTP	Real-time Transport Protocol
GLC	OpenGL & ALSA video capture tool for Linux
FFmpeg	Cross-platform solution to record, convert and stream audio and video
Qt	Cross-platform application development framework
LibVLC	Video decoding and playing framework

TABLE OF CONTENTS

1 INTRODUCTION

The project is to create a working example of cloud gaming. In principle, cloud gaming means that the game runs on a server and the player interacts only with the client machine. Game video is captured and then streamed to client. User controls are then send back to the server by the client program. In order to have an enjoyable game experience delay between video and controls has to be minimal.

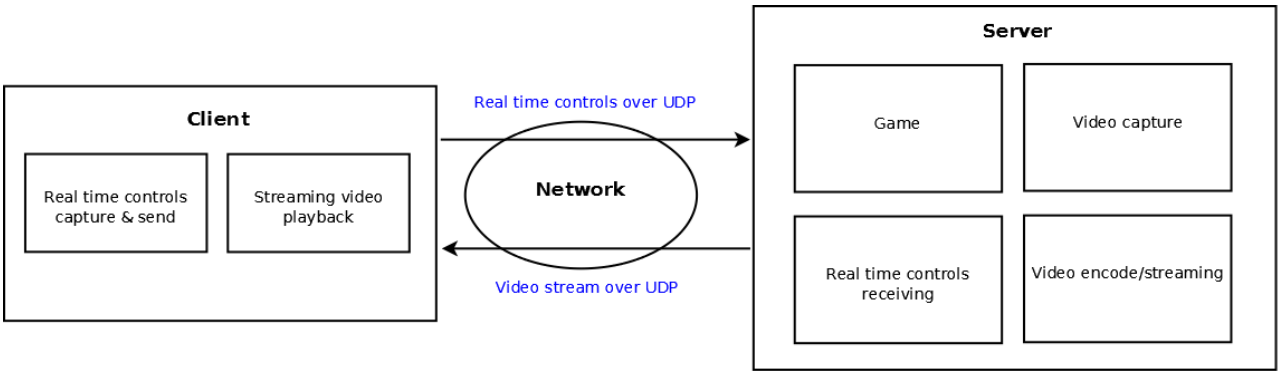
For capturing video from the game itself we chose to use an open source capture framework GLC. GLC capture works only with games that use OpenGL graphics to draw frames and is normally used only to record video, not for streaming it. Raw video from GLC's output is then compressed using FFmpeg and streamed with its FFserver. Stream uses real-time stream transport protocol RTP, which functions on top of UDP. RTP has several advantages over alternatives because of its real-time nature, and lower buffer can be used. Video is compressed as H263, a light weight codec designated to video conferences. Both GLC and FFmpeg were chosen because of their simplicity and sufficient documentation that allowed modifying them to our distinct purposes. We chose not to use Yukon framework for capture, like was originally planned because of its bad streaming performance.

Client software is implemented on top of Qt, a cross-platform application development framework. Qt provides easy-to-use graphical user interface (GUI) and is able to show video with LibVLC libraries and send game controls to server. In server end, software that receives the controls is also written on top of Qt. The cross-platform framework allows us to port the client both to Windows and Linux without problems. LibVLC is a framework for Videolan Client, a popular cross-platform media player, but it can also be used standalone with our client software. It provides tools for receiving and decoding the video stream, and as well displaying it in the user interface.

In this project we are using only UDP network protocol because of its time-sensitive nature. The protocol is simple and has no handshaking or means to determine packet loss between client and server. Therefore it should provide faster performance than traditional streaming on top of HTTP which uses TCP. Real time controls are also a good excuse to use UDP.

2 SYSTEM ARCHITECTURE

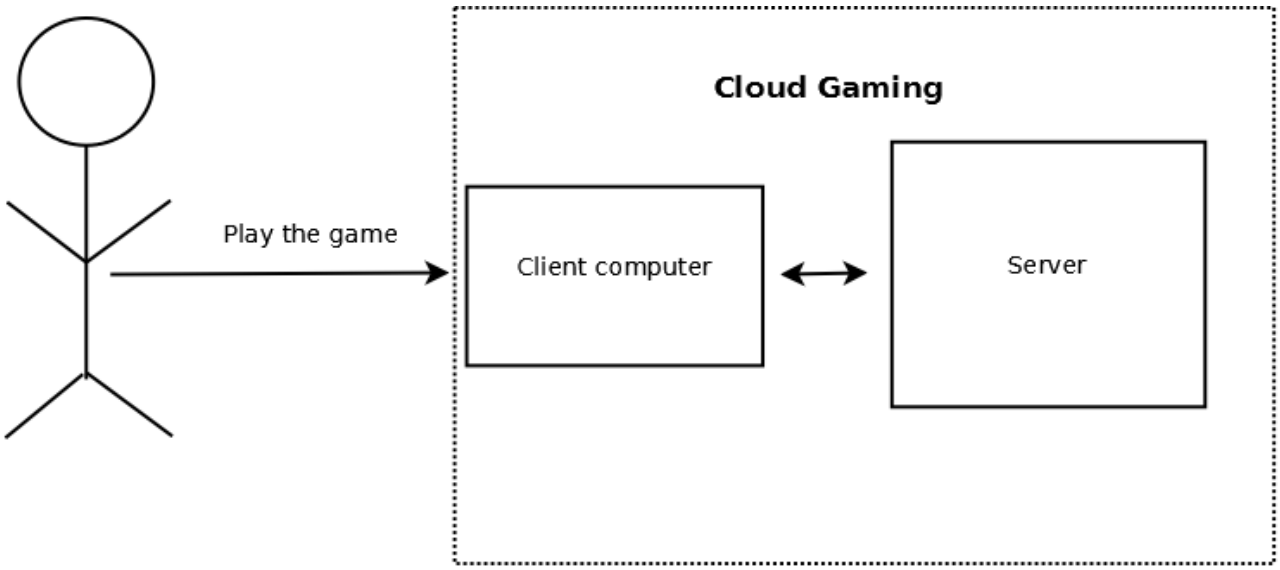
Diagram of system architecture:



In our system, we have the client program and server framework. Server framework controls several programs, like the game itself. Video is captured, encoded and streamed over network connection. Client program receives the video stream, shows it on screen and sends user's controls to server over the connection using UDP data packets. There are also some features to address the possibility of connection loss, such as automatic reconnect.

3 USE CASES

3.1 Use Case diagram



Our use-case diagram has only one actor: user. Its only use-case is to play the game, which runs on our cloud gaming framework.

3.2 Use Case description

3.2.1 Use Case 1: Play the game

Actors:	User
Preconditions:	User wants to play the game, which is running on a server
Description:	1. User connects to a server by using client program 2. When connection is established the game is ready to be played 3. User plays the game
Post-conditions:	Satisfying gaming experience is enjoyed by the user.

4 REQUIREMENTS

A list of requirements for the system is presented here in a brief textual form. Technical requirements present the requirements for the hardware/software platform, whereas the functional requirements define the requirements for the developed software. The requirements are derived from the Use Cases defined on the previous page.

Priorities:
1=must implement
2=should implement (time allowing)
3=be nice to have

4.1 Technical Requirements

Technical Requirements

#No	Description	Priority
1.1	Server is able to run the game	1
1.2	Client can run video stream	1
1.3	Network latency or lost packets do not affect game play significantly	2
1.4	Server compresses video efficiently without latency	2

4.2 Functional Requirements

Functional Requirements

#No	Description	Referred Use Cases	Priority
1.1	Client sends keyboard controls to server	1	1
1.2	Client sends mouse controls to server	1	2
1.3	Client receives video stream	1	1
1.4	Server receives keyboard controls	1	1
1.5	Server receives mouse controls	1	2
1.6	Server sends video stream	1	1
1.7	There is minimum latency	1	2
1.8	Client program works on many different operating systems	1	2

4.3 Project work scope

4.3.1 The operating environment of the system

The operating environment has at least two computers, server and client. They are connected via some kind of fast network connection. In our demo case we will have two laptops connected to each other by an Ethernet cable.

4.3.2 Platform

Server software is run on Ubuntu Linux operating system. For the client are two choices, either Windows 7 or Ubuntu.

4.3.3 Limitations, constraints

Both platforms, server and client, do have to have enough power to run the game or the client program. Video compression takes quite a lot of processor capacity so the server computer should have high performance and efficient compression has to be used.

4.3.4 Reliability requirements

The fundamentals of cloud gaming means that data is send through network connection. That brings many complications in programming. Main problem is to handle the situation when connection is lost. The problem can be solved just simply by reconnecting client to server. Also, there can be other problems in connection, like high latency. If video stream or controls packages are lost, situation should be handled. On platform and software, problem is simply to avoid all bugs. Server or client should not crash in any time because of bad coding or malicious user input.