# *Multimedia Systems –*
# C-WG3#1
# Design Specification

## PURPOSE OF THIS DOCUMENT

This document contains everything concerning design and implementation of the course work.

## REVISION HISTORY

| Revision # | Date | Author(s) | Comments |
|---|---|---|---|
| 0.1 | 09.11.09 | Pasi Keski-Korsu | Template created |
| 0.2 | 09.11.09 | Pasi Keski-Korsu | Notes added |
| 0.3 | 30.11.09 | Pasi Keski-Korsu | Text and pictures added to chapter 4 |
| 0.4 | 30.11.09 | Pasi Keski-Korsu | Chapter 4 written |
| 0.5 | 30.11.09 | Antti Väyrynen | 1&3 added |
| 0.6 | 30.11.09 | Pasi Keski-Korsu | Table of contens and small fixes |
| 1.0 | 30.11.09 | Lauri Majamaa | Final version, converted to docx & pdf |
| 1.1 | 2.12.09 | Lauri Majamaa | Revision, rewrite of chapter 3&5 |

## DEFINITIONS AND ACRONYMS

| | |
|---|---|
| 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 |
| ACK | Acknowledgment |
| Qt | Cross-platform application development framework |
| LibVLC | Video decoding and playing framework |
| Git | Fast Version Control System |
| OpenGL | Open Graphics Library |

# 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. We decided to have Linux on the server and Windows on the client machine. The server will be running the game, video streaming scripts and software to receive game controls. 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 the time difference 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 MPEG4, which uses motion compensation and produces adequate quality with low bitrates. 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. For the version management we chose to use Git.

# 2 FRAMEWORK AND PROTOCOLS

In the current project many network protocols, technologies and ready open source frameworks are used to make everything work. Qt – a crossplatform application and user interface development framework was used both in the client and server for easier implementation and portability of the client between Windows and Linux. GLC is an sound and OpenGL – video capture tool for linux. It can capture video from any game using OpenGL to render graphics. OpenGL is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. With our modification GLC also allows writing the captured file to a pipe, what is read by FFmpeg. It encodes it for streaming and sends it forward when client connects to server.

Streaming uses RTP, which is a standardized packet format delivering audio and video over the Internet. RTP is used extensively in communication and entertainment systems that involve streaming media. It allows stripping of late-arrived frames, giving some latency compensation. RTP is run on top of UDP, which is a network protocol used to send datagrams to other hosts on an Internet Protocol network without requiring prior communications to set up special transmission channels or data paths. It uses a simple transmission model without implicit hand-shaking dialogues for guaranteeing reliability, ordering or data integrity. Client uses LibVLC library from the popular VLC crossplatform media player to decode and play the game stream. It can be embedded to an application to gain audio/video playing features. User commands are send using Qt API to record mouse movements and key presses. Server generates global keycommands from them with XTest and directs them to the game. XTest is an extension library for X.org X11 server. It is a minimal set of client and server extensions required to completely test the X11 server with no user intervention.

# 3 DESIGN LIMITATIONS

Our program runs on two different computers. Client computer does not have huge limitations on hardware or software, but server computer needs quite a lot processor capacity. Running the game itself needs a lot resources and added to that server must, among other things, compress the video. One particular task is not heavy to process but when there are lots of small tasks the total need of resources is quite high.

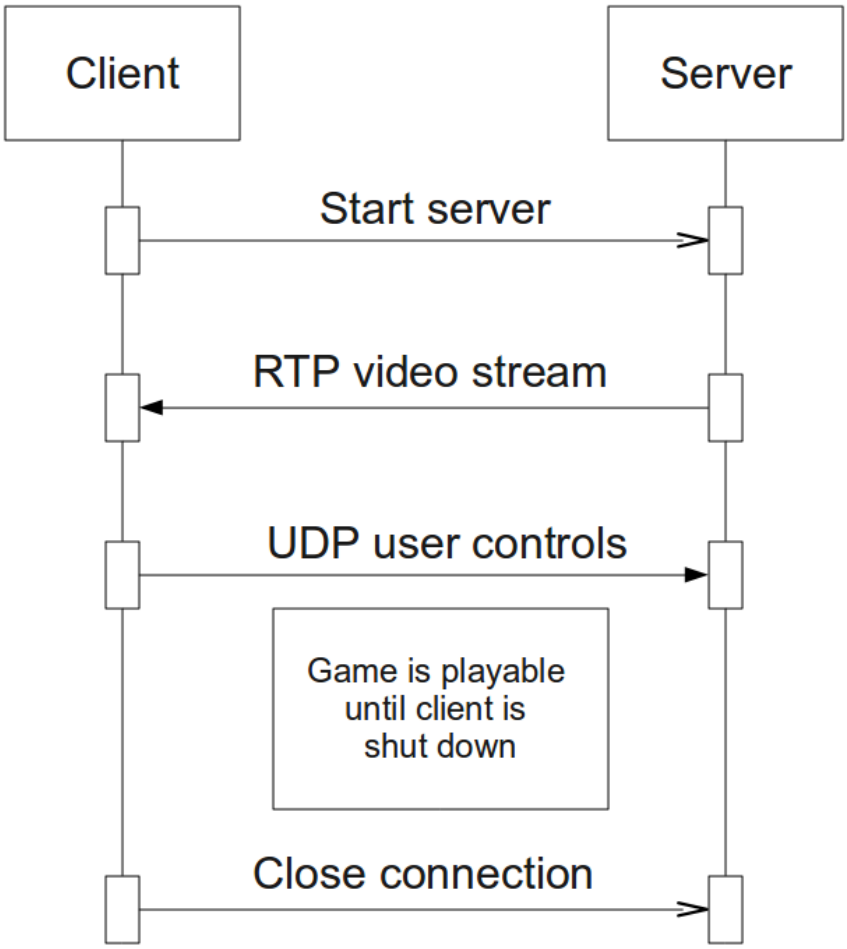
Second main limitation in our project is network. Any kind of network is usable to our program but the best option is to use straight 1Gbit Ethernet cable between computers. Ethernet cable is the most efficient way to connection between server and client, because no additional latency added by routers or other network equipment. Public Wlan or ordinary internet connection brings up more delay, but also limit the usable bitrate for streaming. With current codecs in mind, atleast 100 Mbit network connection between computers is needed to reach good picture quality and larger screen sizes.

# 4 PROJECT DESIGN

## 4.1 General

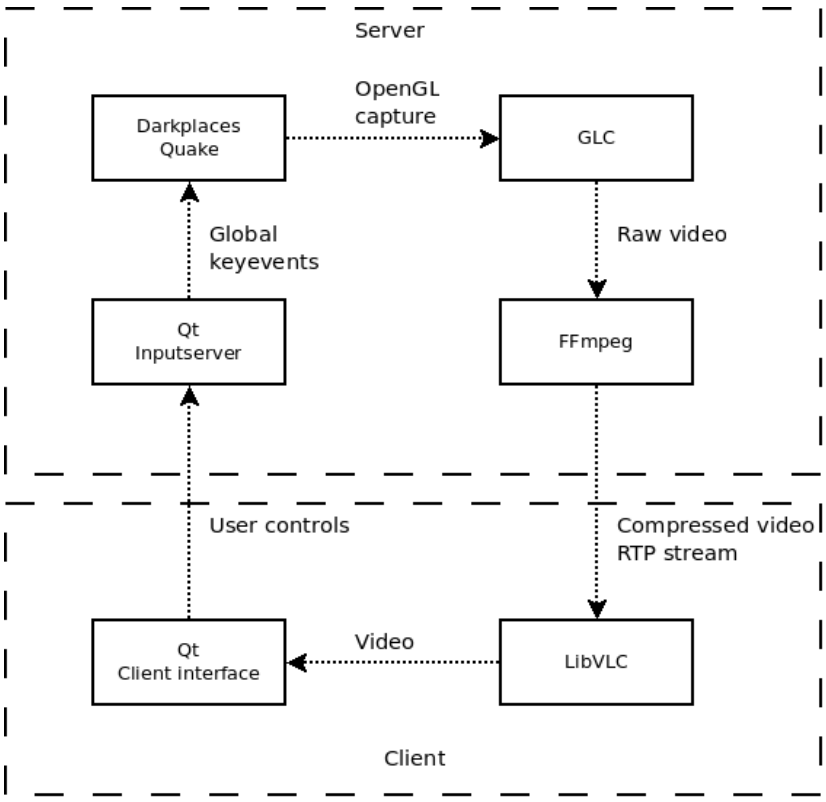
In this section, project design is presented with a help of UML graphs. Sequence diagram presents the data packages which are send over the network. Component diagram shows the third party and other software components in this project. Lastly, we introduce you the main algorithms and user interface in our program.

## 4.2 Sequence Diagram




In our sequence diagram, we have client computer on the left and server computer on the right. First, client sends a message to server to start the game. When server has received the message it starts to send video stream and is ready to receive user controls. User plays the game by getting video stream on his/her screen and sending controls over the network. When user wants to close the program client sends a message to server to close the connection. We use UDP packages to communication so client and server do not send ACK messages to each other.

## 4.3 Component Diagram




In our project we use third party software components quite a lot. We use FFmpeg, GLC, Qt and Darkplaces in our server computer. Darkplaces is the game which is played by the user from the client computer. GLC captures the video stream and FFmpeg packs the raw video and sends it to client. Client receives compressed video with LibVLC. Client interface and Inputserver is developed by using Qt framework.

## 4.4 Main Algorithms

Our main algorithms are mostly third party software components. We use open source program FFmpeg and MPEG4 video codec to capture, encode and stream the video. With FFmpeg video is captured and streamed. Video codec MPEG4 packs the video so we can use our network bandwidth more efficiently. RTP is a standardized protocol for sending audio and video packages over the Internet. Despite of specificied codecs, our design allows to quickly change used codecs and video picture size depending on the available network.

## 4.5 User Interface

User starts to use the program by executing the file Client.exe. First, program opens the menu screen (Picture 1) where user can press “Connect” to start playing the game or “Exit” to exit the program. After pressing the button “Connect” program connects to the server computer and starts playing the with the game user interface (Picture 2).

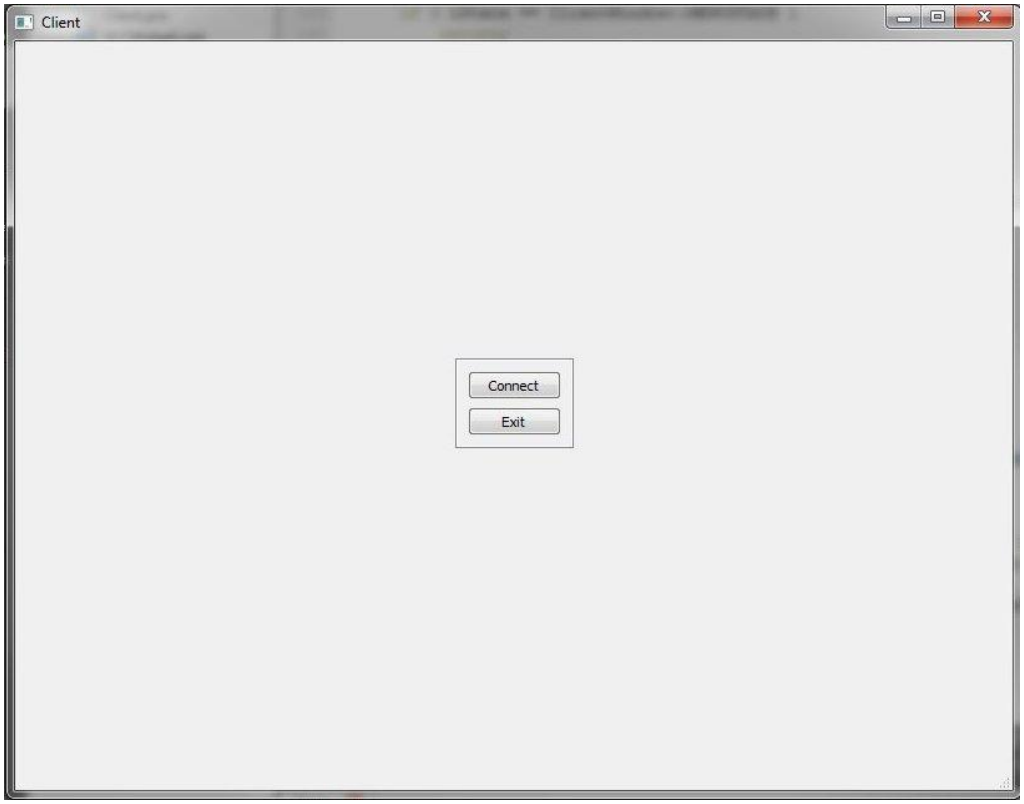


Image 1. The program menu


Image 2. The game stream

# 5 COMPILING & EXECUTING INSTRUCTIONS

## 5.1 Compiling & Executing Instructions for client

### 5.1.1 Compiling for Ubuntu 9.10:

sudo apt-get install libvlc-dev libqt4-dev qt4-qmake

qmake
make

Run:

./Client

### 5.1.2 Compiling under Windows

Compiling source files can be done using Qt Creator included in the Qt SDK by Nokia. Note: VLC plugins folder must be in the same folder with the binary file.

## 5.2 Compiling & Executing Instructions for server for Ubuntu 9.10 32bit

### 5.2.1 Installing depencies

sudo apt-get install build-essential libxtst-dev libqt4-dev qt4-qmake ffmpeg
sudo apt-get install build-essential cmake libx11-dev libxxf86vm-dev libgl1-mesa-dev libasound2-dev libpng12-dev

### 5.2.2 Compiling GLC:

cd server/glc/elfhacks
cmake . && make && sudo make install
cd ../packetstream
cmake . && make && sudo make install
cd ../glc
ln -sf ../glc-support ./support
cmake . && make && sudo make install

### 5.2.3 Compiling Inputserver

cd ../../inputserver
../scripts/mkfifo.sh
qmake
make

### 5.2.4 Starting the server

cd server/inputserver
./inputserver

Note: Original datafiles of Quake for DarkPlaces engine are not included because they are not published under open source licence. Therefore, an original game is needed to run DarkPlaces.

# 6 REFERENCES


[1] GLC video capture framework: http://nullkey.ath.cx/projects/glc/

[2] DarkPlaces Quake modification: http://icculus.org/twilight/darkplaces/

[3] Qt Cross Platform Application and UI framework: http://qt.nokia.com/products