EGGS ON THE DOOR

CENG 491

Initial Design Report

Duygu ÇELİK -Engin ERTAŞ – Serap İNCE

2010

Table of Contents

1 Introduction 4

1.1 Problem Definition 4

1.2 Purpose 4

1.3 Scope 4

1.4 Overview 4

1.5 Definitions and Abbreviations 5

1.6 References 5

2 System Overview 5

3 Design Considerations 6

3.1 Design Assumptions, Dependencies and Constraints 6

3.2 Design Goals and Guidelines 7

4. Data Models and Design 7

4.1 Network Settings & GamePlay Settings & Preferences 7

4.2 Friend List 7

4.3 Chat Diaries & Game Results 8

4.4 Extra Data Types 8

4.4.1 IP 8

4.4.2 Friend 8

5 System Architecture 8

5.1 Architectural Design 8

5.2 Description of Components 10

5.2.1 Network Module 10

5.2.2 GamePlay Module 21

5.3 Design Rationale 23

6 User Interface Design 29

6.1 Overview of User Interface 29

6.1.1 General System Functions 29

6.2. Screen Images 33

6.3. Screen Objects and Actions 34

7 Libraries and Tools 34

8 Time Planning (Gannt Chart) 35

8.1 Term 1 Gannt Chart 35

8.2 Term 2 Gannt Chart 35

9 Conclusion 35

1 Introduction

1.1 Problem Definition

Recently, millions of people have been using the social network sites on the Internet. Facebook and Mynet are popular, since they not only provide people with chatting feature with millions, but also give the opportunity to play multi-player games simultaneously. However, one of the most crucial lacks of these networks is the fact that they do not give users the capability of communicating in an audiovisual manner. On the other hand, there are some software, such as, Windows Live Messenger or Skype providing visual communication; still, they do not support those facilities to more than two users on the same conversation window. Moreover, more than two people cannot play online games using this software at the same game board.

1.2 Purpose

In this project, we propose to develop a video conferencing software in which people can play games that require real time audiovisual interaction between players, such as “taboo”.

The first aim of the project is to provide people with a framework in which two or more people can make audiovisual conferencing in the same window.

Secondly, we aim to develop a flexible gaming platform / console on which various multi-player games written for this software can be executed.

1.3 Scope

Our project will be a .NET based Audiovisual Video Conferencing Gaming Network. Our project will assist the people’s needs to make video conferencing with multiple people and also play games in the same platform. The users can be either clients or host in the system. Any user who wants to create a game table can be the host, and invite people from the friend-list to play the game.In such a case, the host should provide her/his IP address and a password to the clients which is the requirement to join the game. During the game, people will be able to make audio-visual conferencing, which serves a feature differing the software from the present gaming platforms. Our system is aimed to handle 6 people at the same time and respond in a specified time, but due to the bandwidth limits or any unsupported system requirements, the performance may decrease. Our aim is to provide a platform that different games can be implemented so that it will be easy to improve the project in the future.

1.4 Overview

Readers get a clear understanding of the project and basic design choices by examining this initial design report. With the help of several UML diagrams, the modular architecture of the system is clearly defined

in the document. Which public methods that modules have are listed in class diagrams, indicating the possible ways of inter-module interaction.

1.5 Definitions and Abbreviations

CLI: Command Line Interfaced
GUI: Graphical User Interface
MPEG: Moving Picture Experts Group
H.261 : the video coding standard of the ITU
IDR : Initial Design Report
SRS: Software Requirement Specifications

1.6 References

Westwater R., Furht B., "*Real-Time Video Compression: Techniques and Algorithms.*" America: Cruvel Academic Publishers Group, 2006, np.

Ouaret M., Dufaux F., Ebrahimi T., "*On Comparing Image and Video Compression Algorithms.*" Institut de Traitement des Signaux Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland.

Balaško H., "*Comparison of Compression Algorithms for High Definition and Super High Definition Video Signals."* Audio Video Consulting Ltd., Karlovačka 36b, 10020 Zagreb, Croatia

Raghuveer A., Kang N., David H. C. *"Techniques for Efficient Streaming of Layered Video in Heterogeneous Client Environments."* Dept. of Computer Science & Engg. School of Computer Science & Engg. Dept. of Computer Science & Engg , University of Minnesota Chung-Ang University University of Minnesota Minneapolis, MN 55455 South Korea Minneapolis.

2 System Overview

Our target software has three main functionalities. The first functionality is supplying communication between any group of people whose number of members does not exceed 6. In our system, this communication is supplied with the help of direct IP connection. In order to make this facility available, we are going to use some libraries of C# programming language and make use of fundamental network algorithms.

The second functionality is providing the software users with the capability of involving in audiovisual conversation. In order to reach that aim, we consider using some streaming algorithms that are explained further in following sections. Also, since our system's audiovisual ability is available to 6 people, we need to decrease the streaming size. Therefore, in order to make sampling rate considerable, we will use some

compression algorithms. Moreover, to enhance the compressed video streams, we plan to use filter algorithms mentioned in subsequent sections of our document.

The last functionality of our software is supporting many multi-player Java games which can communicate with the software with the help of Game Play Interface. Therefore, after the software gets in use, Java games could be classified as “supported” and “unsupported” with respect to their abilities to connect the main software.

In order to make a not complex design to reach these aims, we are supposed to follow a modular approach. Namely, dividing the system into some sub-modules with regards to different functionalities not only ease the project development steps, but also helps us to make our product reliable, efficient and consistent.

Our project is divided into two main modules (*Network* and *GamePlay*) which are clearly explained in *“Architectural Design”* part.

3 Design Considerations

3.1 Design Assumptions, Dependencies and Constraints

Our project aims to entertain people in an audiovisual interactive environment. As a result, one of our goals is to design a system that could not bother people by long response times, latencies and software problems. Hence, system constraints are defined according to this platinum base.

Firstly, we aim to design a system which is capable of responding six people in a suitable time. We choose not to exceed this player limit at our first version of software, since the greater the number of input streams, the heavier the load on the network.

Secondly, each user has different bandwidth limitations. However, the total web cam streams are too big to be received and to be sent, as 6 people are considered. Therefore, we need to reduce the amount of data to keep the system consistent. To get over the limitation problems, we use compression algorithms in order to handle 6 people on the system at the same time.

Thirdly, as we are multi-casting and working with the streams with different qualities, we should find a solution which is suitable in that kind of heterogeneous inputs. To handle this problem, we used the layered multi-casting streaming technique which serves best for our needs. Layered multi-casting streaming divides the video into different layers by filtering, quantization and down-sampling. Then, a basic low-quality image is sent to the receivers. On the computer of clients, with the help of filtering, the images are decompressed. Next, according to the existing hardware and software system of the receiver, the video is filtered and tried to be shown similar to the old quality. Using that kind of streaming

technique decreases the data load on the network. To sum up, our goal is to keep the system stable without losing too much data during the compression and filtering processes, and keep the video in a reasonable quality level.

3.2 Design Goals and Guidelines

Our first goal of design is to implement a system that communicates a few users consistently without synchronization problems with suitable response time and image quality. After we manage to establish the connection between users, we are going to embed the game play module in the system. The goal of this module is to contain an interface which will be a powerful tool to connect different kinds of games to the system. As an external game, we will implement the taboo game (which is the default game) in our project. The reason why we have chosen this game is that it requires visual interaction between the players during the game. Finally, if we have time, we will try to code different games for our system.

4. Data Models and Design

Since we do not have a database in our software, we do not have any complex relations between data and objects. In this section, we are to explain the trivial data types of our software.

4.1 Network Settings & GamePlay Settings & Preferences

In our system, a user has the capability of changing some system features. This ability makes our software much more user friendly.
These settings are going to be included in a single “settings.set” file. That file is supposed to contain each setting on each newline.
In case of error in the setting file, default setting options are going to be chosen by our system.

4.2 Friend List

In our software, each user is capable of constructing his/her friend list. According to our design, this friend list is going to be included in “friend.xml” file. This file is going to include “Friend” objects in the system.

We do not prefer using database here, since we think that a single user on each computer could not have more than 10000 friends. Only an index file is adequate for such a case. This index file is named as “index.dat”. Each line of file includes information only for one “Friend” object.

The system does not allow a friend list to have more than 10000 friend objects. Moreover, in case of file errors, the system uses the blank friend list.

4.3 Chat Diaries & Game Results

A user may save her/his conversation details in external files. This ability is included in Graphical User Interface of Network module. The details of each conversation are written to an external file whose name is determined by the user. The external files of conservations are contained in “diary” sub-directory.

A user may save her/his game results also. Also this ability is included in GUI. The game result reports are designed to be written by external games. However, a user must give permission to make the reports written. The details of each game are included in an external file whose name is determined by the user. The external files of conservations are contained in “results” sub-directory.

4.4 Extra Data Types

4.4.1 IP

As we know, a standart IP has a type like “a.b.c.d”. Here, a, b, c and d are unsigned integers ranging between 1 and 255. Likely, our IP data type has such a structure.

IP:
unsigned int: a
unsigned int: b
unsigned int: c
unsigned int: d

4.4.2 Friend

“Friend” data type consists of three members with primitive data types:

IP: user_ip
char[20]: username
char[30]: notes

5 System Architecture

5.1 Architectural Design

Our Audiovisual Gaming Software is divided into two main modules: *Network* and *GamePlay* modules. While *Network* module mainly focuses on the network communications between server and clients, video streaming, compression and filtering, *GamePlay* module supports various games with the help of designed interface.

Network is the key module of the system which is separated into four sub-modules: *Streaming*, *Filter, Compression* and *NetworkSettings*.

GamePlay module is responsible for connecting with game executables designed for this software. There are going to be many games supported by our product. Therefore, in order to link these games to the software, we need a powerful interface in this module. There are two sub-modules of *GamePlay*: *GamePlayInterface* and *GamePlaySettings*.

One of the most crucial parts of our software is *compiled Java games* supported by our software. By the help of those, people may play enjoyable games during audiovisual conversation. Any game communicating with *GamePlayInterface* module with correct interface messages are supported by our software. As default game in our software, we include Taboo game to be played.

The overall scope of the project is given on *Figure1*.

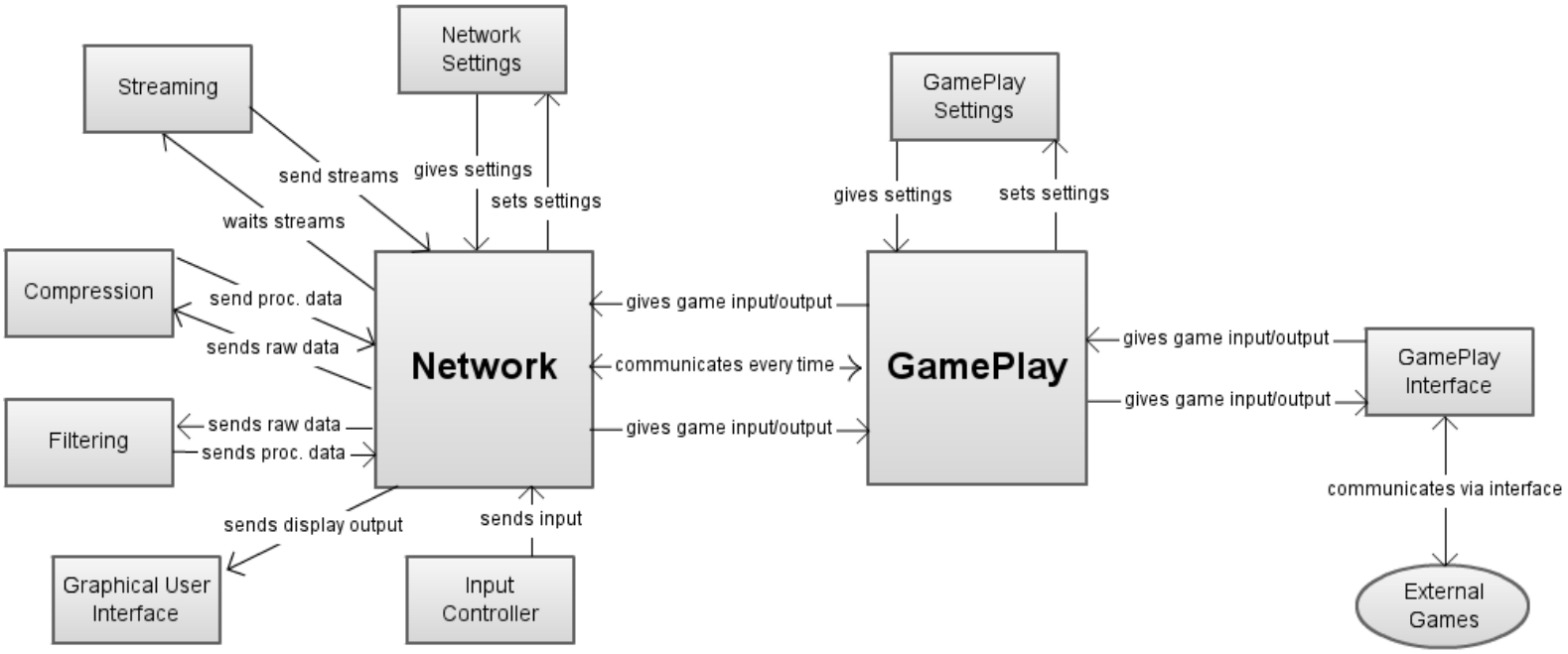


Figure 1: Architectural Context Diagram

5.2 Description of Components

5.2.1 Network Module

5.2.1.1 Network Module General Description

Figure 2: Network Module System Functions

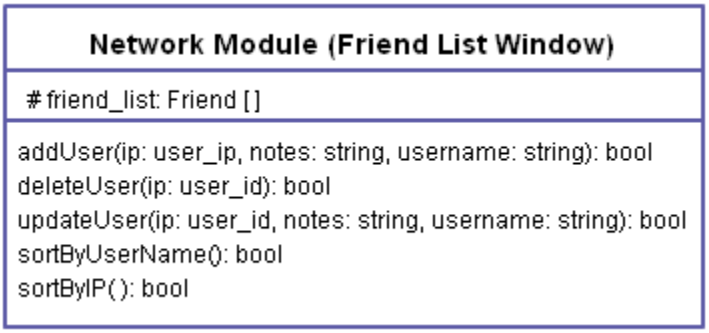


Figure 3: Network Module System Functions for Friend List

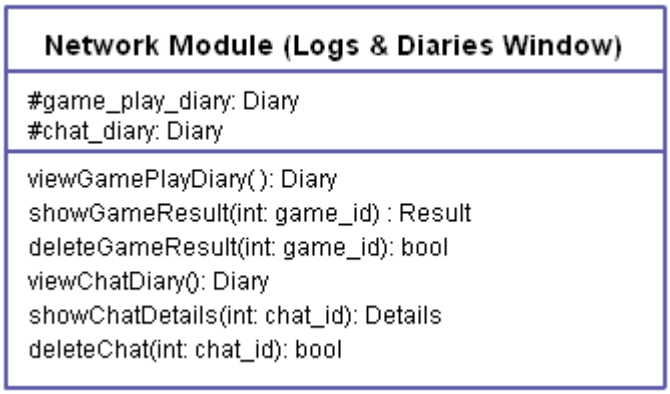


Figure 4: Network Module System Functions for Logs and Diaries

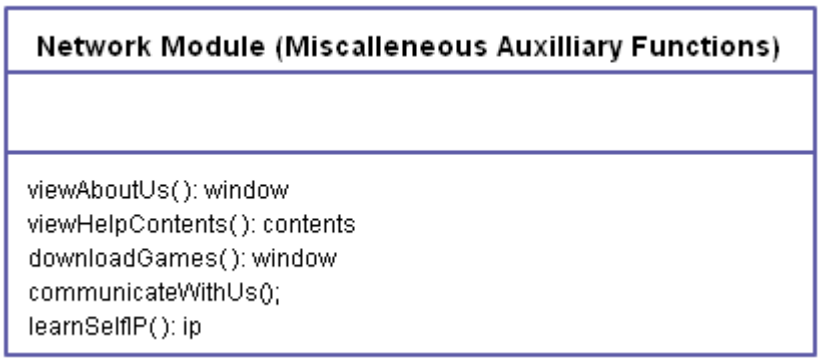


Figure 5: Network Module System Functions for Miscellaneous Auxiliary Functions

5.2.1.2 Compression Sub-Module

COMPRESSION

Compression is one of the most important fields of video processing sending the data without losing too much detail, which is a difficult problem to handle. In our project, as our aim is to design a platform which is going to respond up to 6 people, the less data we send, the less load on the network we have.

Also there are some other constraints to be handled so the ideal compression technique should produce levels of compression rivaling MPEG without objectionable artifacts. These constraints can be expressed in three sentences:

1. Can the system be played back in real time with inexpensive hardware support?
2. Can the system perform under network overload or on a slow platform?
3. Can the system be compressed in real time with inexpensive hardware support?

In the system according to the bandwidth limitation the frames we can be sent per second is limited. At that moment the compression algorithms are needed to send the data in a compressed format. There are different compression algorithms which serves different needs. They can be categorized in two forms as lossless and lossy compression algorithms. In our implementation we will work with the lossy compression algorithms. There are 2 compression algorithms which will be used in the Project in future the one which serves our needs best will be chosen or the both of them may be used according to the system and network constraints features.

MPEG

The most of the video compression algorithms takes the video divides it into images that are not moving and make the compression in a similar manner with the image compression algorithms. In fact, as we are sending those images continuously on the network to more than one people, we need a better compression algorithm. Therefore, compared to the algorithms works in this manner MPEG compression algorithm is more advantageous. The main advantage is that MPEG algorithm divides the image into different segments. After that, it detects the motion in the images meaning that the segments which do not contain any motion from beginning to the end are not send in every frame. The motion is detected on the segments and the parts containing the motion and changes according to that motion are compressed and sent. The technique may be explained with the image

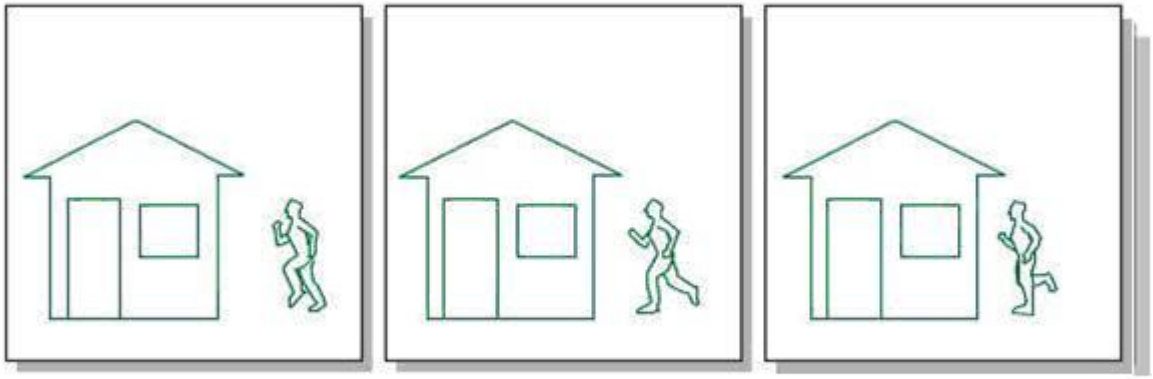


Figure 6: Segments of video used in MPEG

The segment which is shown as stripes stands still during the video so they are still segments and are not needed to be sent in each frame. On the other hand the man in the frames moves and the segments which include the man image contains motion those motions are detected and the segments containing the motion are sent in each frame.

In our project the backgrounds of the users during the conferencing are likely to be constant so that compression algorithm is likely to work fine in our situation.

Figure 7: Segments of video used in MPEG

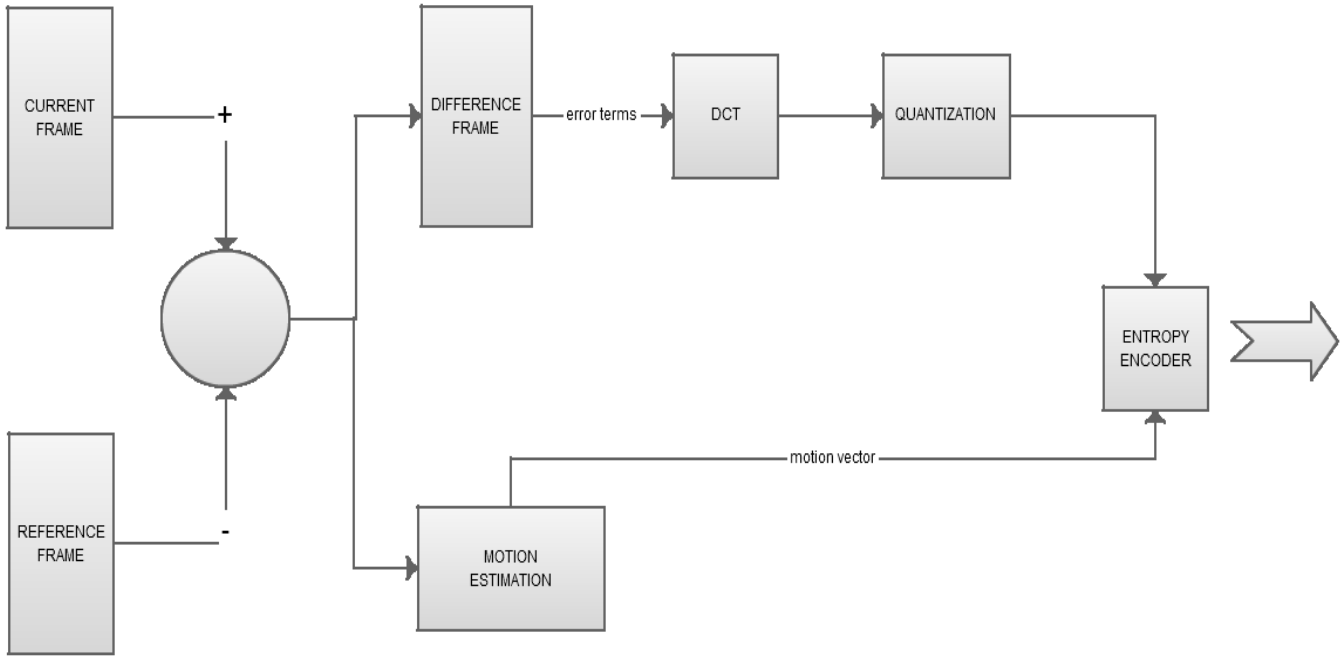


Figure 8: DFD diagram of compression module level 0

H.261 / H263

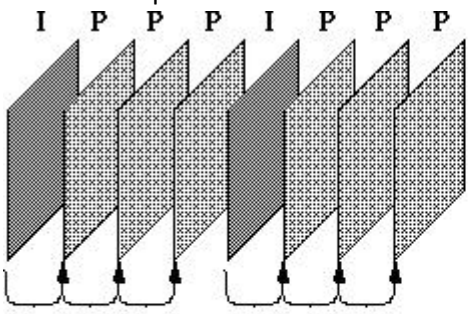
H.261 is video coding standard published by the ITU (International Telecom Union) in 1990. It was designed for datarates which are multiples of 64Kbit/s, and is sometimes called p x 64Kbit/s (p is in the range 1-30). These datarates suit ISDN lines, for which this video codec was designed for.
The coding algorithm is a hybrid of inter-picture prediction, transform coding, and motion compensation. The datarate of the coding algorithm was designed to be able to be set to between 40 Kbits/s and 2 Mbits/s. The inter-picture prediction removes temporal redundancy. The transform coding removes the spatial redundancy. Motion vectors are used to help the codec compensate for motion. To remove any further redundancy in the transmitted bitstream, variable length coding is used.
H.261 supports two resolutions, QCIF (Quarter Common Interchange format) and CIF (Common Interchange format).

The video multiplexer structures the compressed data into a hierarchical bitstream that can be universally interpreted. The hierarchy has four layers:

1. Picture layer: corresponds to one video picture (frame)
2. Group of blocks: corresponds to 1/12 of CIF pictures or 1/3 of QCIF
3. Macroblocks: corresponds to 16x16 pixels of luminance and the two spatially corresponding 8x8 chrominance components.
4. Blocks: corresponds to 8x8 pixels

The basic approach to H. 261 Compression is summarized as follows:

Decoded Sequence



Frame types are CCIR 601 CIF (352x288) and QCIF (176x144) images with 4:2:0 subsampling.
Two frame types: Intraframes (I-frames) and Interframes (P-frames)
I-frames use basically JPEG
P-frames use pseudo-differences from previous frame (predicted), so frames depend on each other.
I-frame provides us with an accessing point.

Protocol Structure - H.261: Video Coding and Decoding (CODEC)

3	6	7	8	12	17	22	27	32bits
SBIT	EBIT	I	V	GOBN	MBAP	QUANT	HMVD	VMVD

- SBIT - Start bit. Number of most significant bits that are to be ignored in the first data octet.
- EBIT - End bit. Number of least significant bits that are to be ignored in the last data octet.
- I - INTRA-frame encoded data field. Set to 1 if this stream contains only INTRA-frame coded blocks and to 0 if this stream may or may not contain INTRA-frame coded blocks.
- V - Motion Vector flag. Set to 0 if motion vectors are not used in this stream and to 1 if motion vectors may or may not be used in this stream.
- GOBN - GOB number. Encodes the GOB number in effect at the start of the packet. Set to 0 if the packet begins with a GOB header.
- MBAP - Macroblock address predictor. Encodes the macroblock address predictor (i.e., the last MBA encoded in the previous packet). This predictor ranges from 0-32 (to predict the valid MBAs 1-33), but because the bit stream cannot be fragmented between a GOB header and MB 1, the predictor at the start of the packet can never be 0.

- QUANT - Quantizer field. Shows the Quantizer value (MQUANT or GQUANT) in effect prior to the start of this packet. Set to 0 if the packet begins with a GOB header.
- HMVD - Horizontal motion vector data field. Represents the reference horizontal motion vector data (MVD). Set to 0 if V flag is 0 or if the packet begins with a GOB header, or when the MTYPE of the last MB encoded in the previous packet was not MC.
- VMVD - Vertical motion vector data (VMVD). Reference vertical motion vector data (MVD). Set to 0 if V flag is 0 or if the packet begins with a GOB header, or when the MTYPE of the last MB encoded in the previous packet was not MC.

5.2.1.3 Filtering Sub-Module

As we are compressing the videos when sending and decompressing them after receiving there are some artifacts occur during these processes. To reduce those artifacts and improve the quality of the video we use deblocking filter. The filtering part of the video process can be optional or it can be implemented both in the encoding and decoding part of the processing. According to the quality of the videos we had and we needed in our project we may chose to implement the filters both in the encoding and decoding or we may just use it on the decoding part. In the implementation we will use the deblocking filter which is suitable for MPEG and H261 formats.

The Deblocking Filter

A deblocking filter is applied to blocks in decoded video to improve visual quality and prediction performance by smoothing the sharp edges which can form between macroblocks when block coding techniques are used. The filter aims to improve the appearance of decoded pictures.
The deblocking filter exploits spatial redundancy. There are in loop filters and post filters available for video compression techniques we used in our system. We plan to use the in loop filters in our implementation. The in loop filter works in the same loop during the encoding and decoding process that no additional processing duration is needed for filtering after the encoding operation is done. Also deblocking filter takes part in the decoding part and processed in the decoding loop. As we could have videos of different qualities from different users for some of them after the compression techniques could be needed but in case of the too much complexity and latency in the response time we may choose the post filtering algorithms instead.

This module will compress the video frames using the MPEG and h.261 video compression techniques and prepare them to be sent to the receivers over the network. This module will both filter the videos for noise reduction that during the compression the noisy parts are ignored and less amount of data is dealed with. After the decoding the filtering module filters the compressed data to receive fine quality videos.

5.2.1.4 Streaming Sub-Module

Streaming module takes the data and divides it into different layers before packaging and sending them to the receivers. After quantization and downsampling the basic layer is determined and it is sent to the receivers.

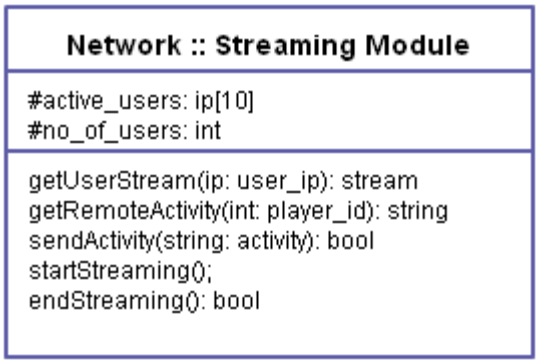


Figure 9: Streaming Module System Functions

MACRO BLOCK CODING

- ***Inter – Block Encoding***

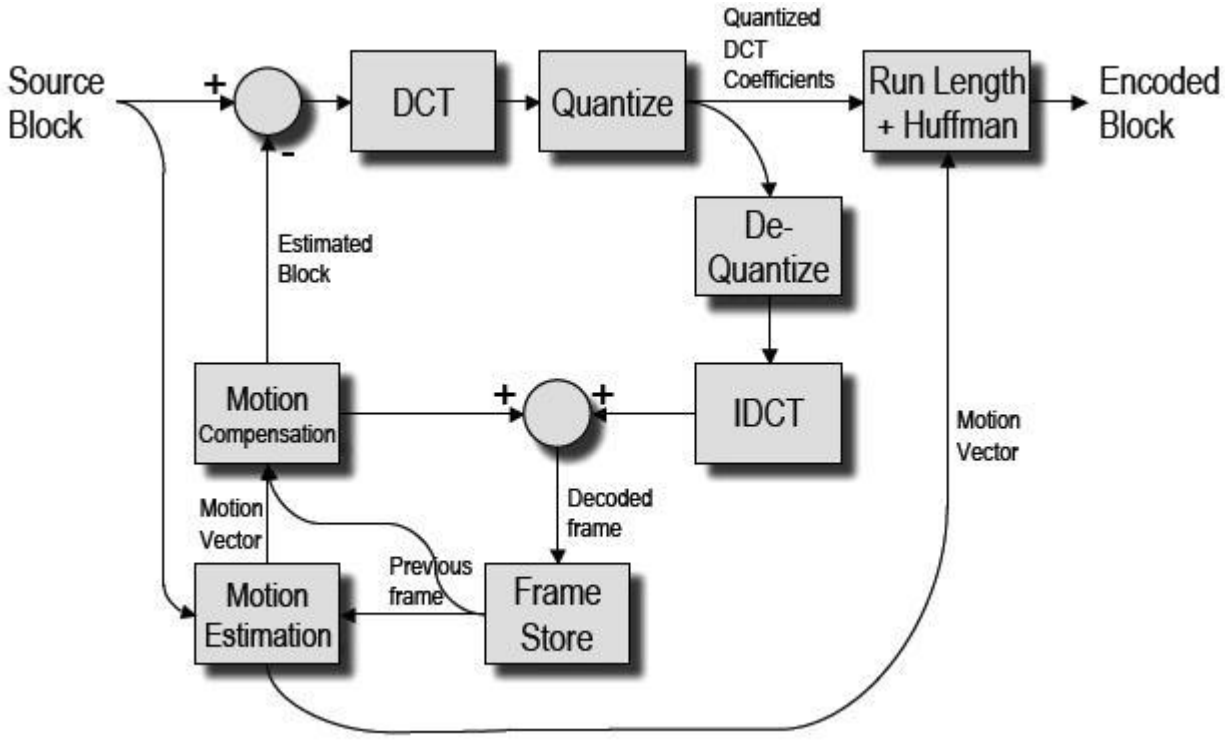


Figure 10: Inter Block Encoding

- **Intra Block Encoding**

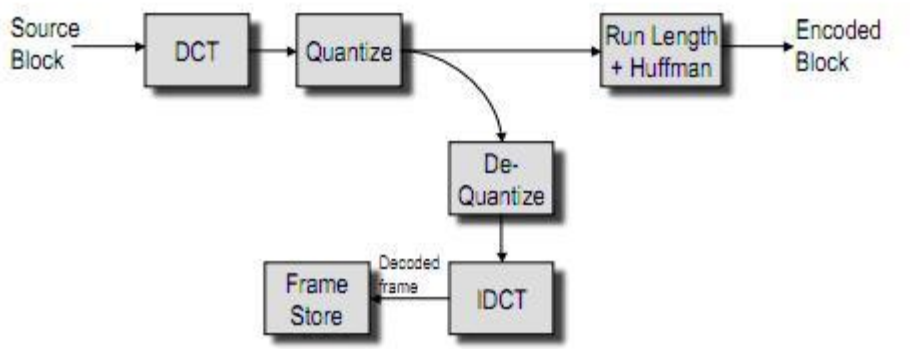


Figure 11: Intra Block Encoding

Bit Stream Structure

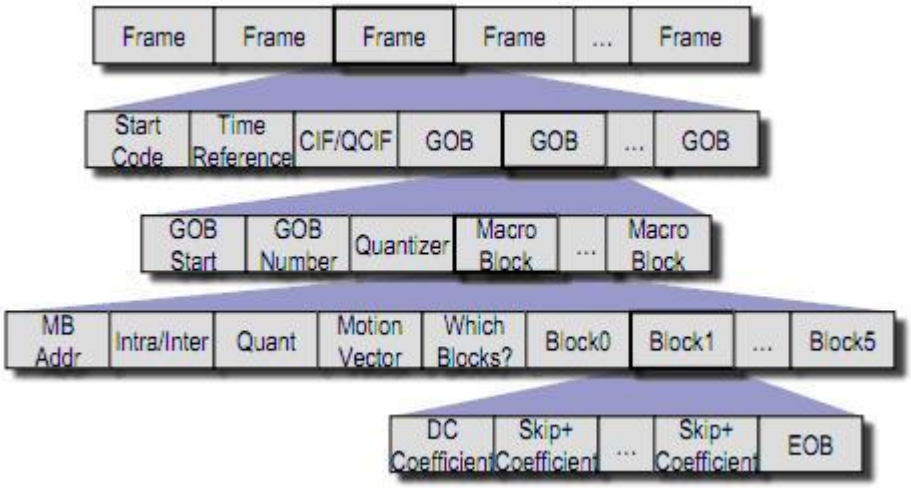
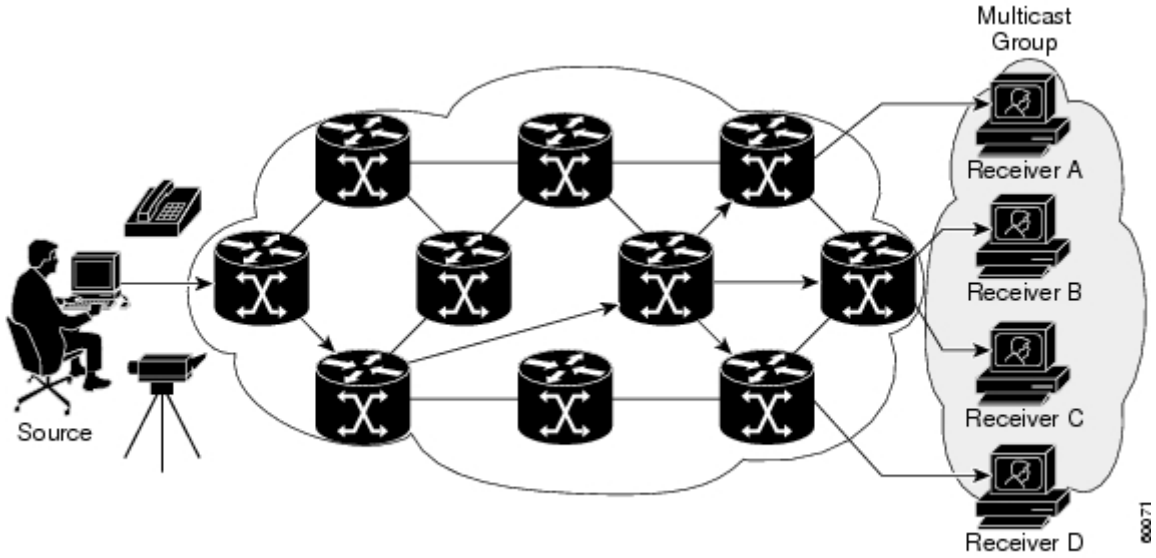
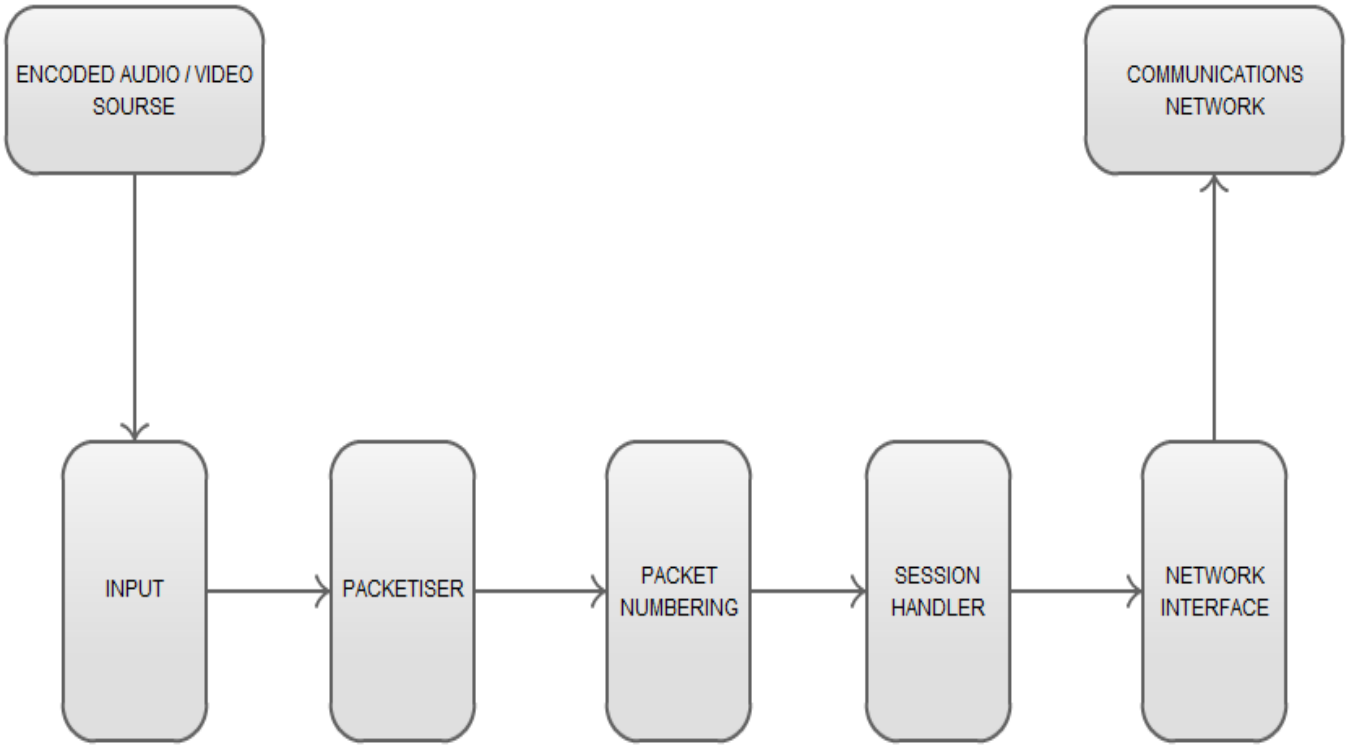


Figure 12: Bit Stream Structure

- **Layered multi-cast Streaming**



As we will receive videos from the client side in different qualities we have heterogeneous multi-cast sessions. To come over with that problem we choose the layered multi-cast streaming method. In layered approach the raw video is encoded into o set of cumulative (or non-cumulative) layers. The basic layer contains the essential video information with a low basic quality and a set of enhancement layers are used to improve the quality of the received video on client sides. In the MPEG-4 coding, which will be used as our compression format in the project, coding layers can be obtained by applying Temporal scaling, Spatial scaling of fine Granularity scaling. The set of layers are sent over separate multicast groups. A receiver joins first the basic layer to obtain the basic video quality and then adapts the rate depending on the capabilities by joining or leaving the enhancement layers. Depending on the set of joined layers, heterogeneous receivers will obtain different video quality. By using that streaming approach we intend to solve the deletion of response time due to the different hardware and software equipments of the users. In addition the commands given by the players during the game the changes on the game interface the outputs of the functions the communication between the players is also sent by the streams. The video sequences and the function outputs will be sent as numbered packages to the receivers. On the receivers side if there are any problems in receiving the package or processing by communicating with the other players and processing the same data package with them the system is tried to be synchronized.



5.2.1.5 Network Settings Sub-Module

Network settings module sets the bandwidth limitations for each user according to the current user number. It works in corporation with the compression filtering and streaming submodules. Network settings submodule can start and stop the compression and filtering processes. After that the data is ready to be sent as streams.

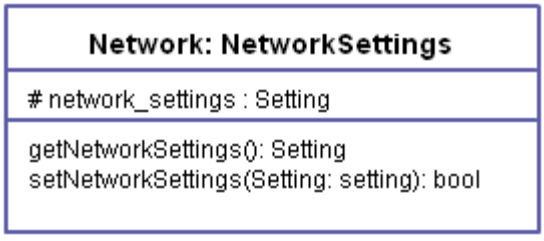


Figure 13: Network Settings System Functions

5.2.1.6 Graphical User Interface of Network Module

Graphical user interface module provides the interaction of the system with the user. The user can see his or her own preferences by clicking the view preferences choice. After that, if any preferences are changed the system is refreshed and the new preferences are presented to the user by the system.

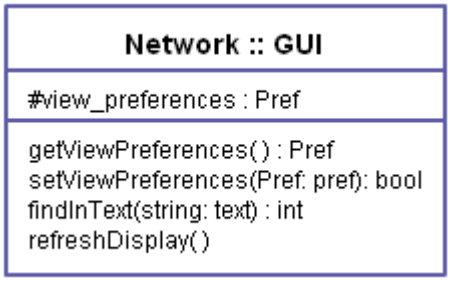


Figure 14: GUI System Functions

5.2.2 GamePlay Module

5.2.2.1 GamePlay Module General Description

The game play module consists of the functions related to starting a game inviting people to games and other related functions. Some of the GamePlay module functions are invoked by the buttons of the Graphical User Interface Module. When the user clicks on the "*start game*" button he or she is assigned as host after that using the invite player button other players are invited to the game. After the responses are received from the invited users the getPlayerCount and setPlayerPositions functions are invoked and the player count and order is determined by the system. Finally the game is ready to be started. After the game is started the movements of the players are sent and received through the sendActivity and getRemoteActivity methods and the state of the system is set.

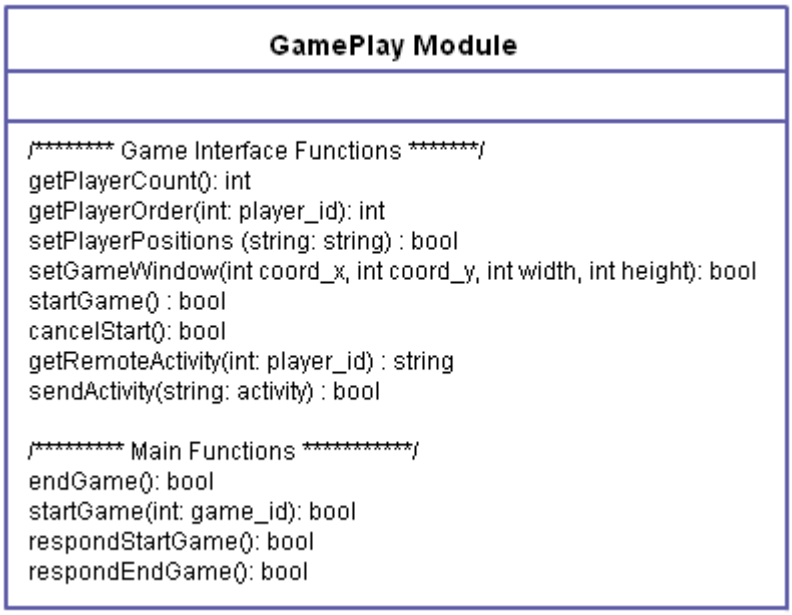


Figure 15: GamePlay Module System Functions

5.2.2.2 GamePlaySettings Sub-Module

GamePlaySettings sub-module basically gets the game setting from the user through the interface module and invokes the system about the new settings and keeps the new ones using the getGamePlaySettings and setGamePlaySettings methods.

Figure 16: GamePlay Settings System Functions

5.2.2.3 GamePlayInterface Sub-Module

External Game Play interface module is related to the interface of the game which is currently played by the users. According to the chosen game the player count and player order is inherited and the positions of the players and game window are set. If the host decides to cancel the game by pressing the cancel button the method is invoked. During the game the players are informed about the other users movements by the getRemoteActivity methods and the movements they have made are sent to the other users by the sendActivity method.

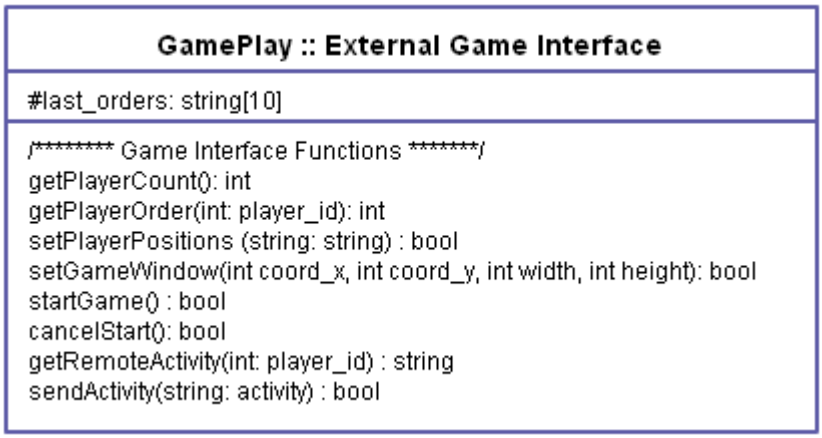


Figure 17: GamePlay External Game Interface System Functions

5.3 Design Rationale

The sequence diagrams of the functions the communication between them are explained in the following diagrams.

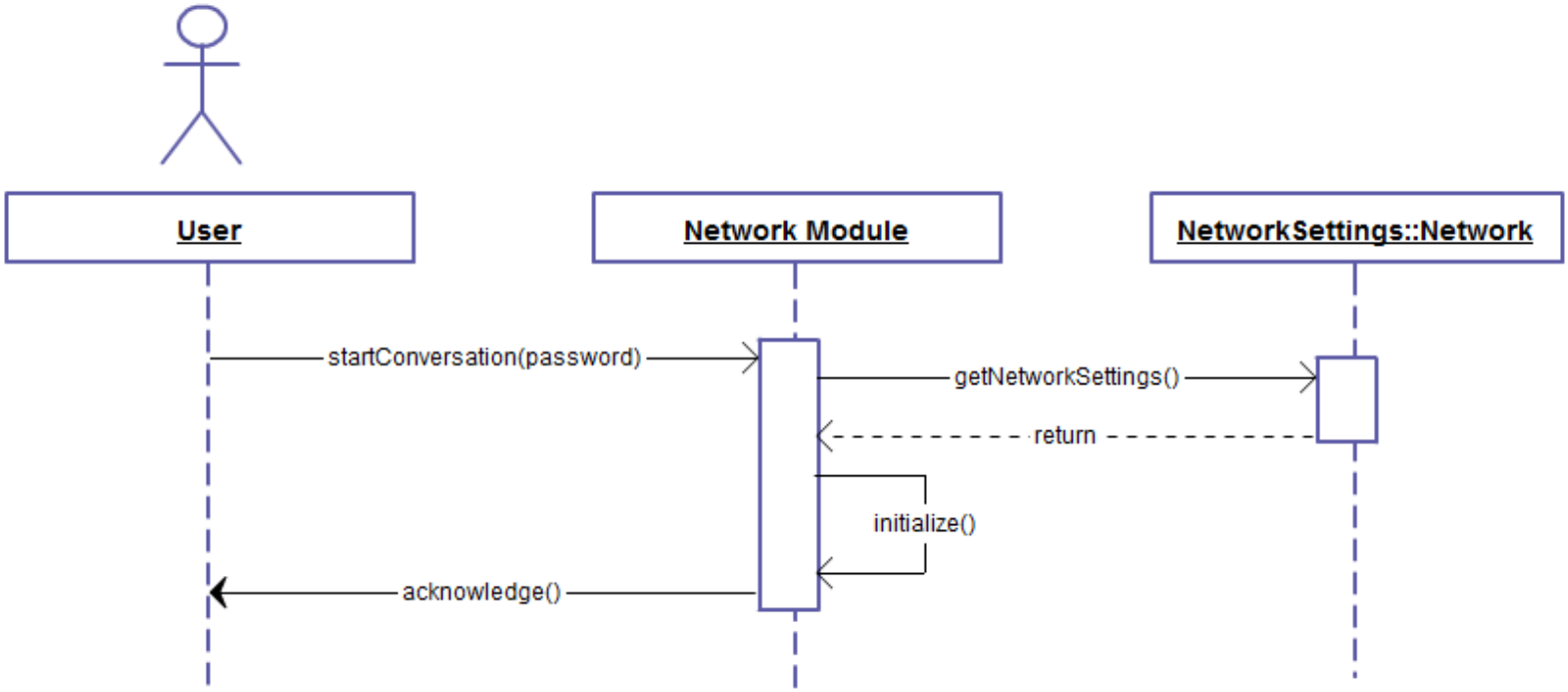


Figure 18: Start Conversation System Function

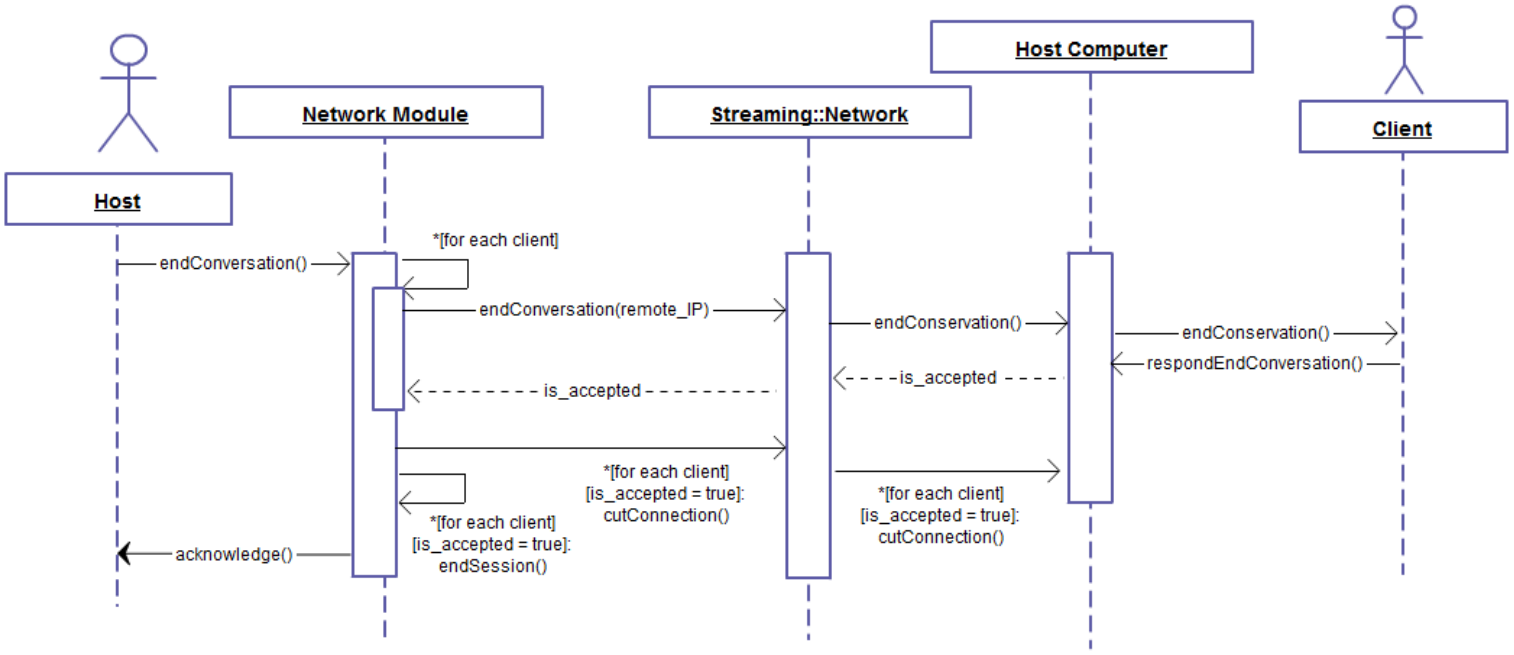


Figure 19: End Conversation System Function

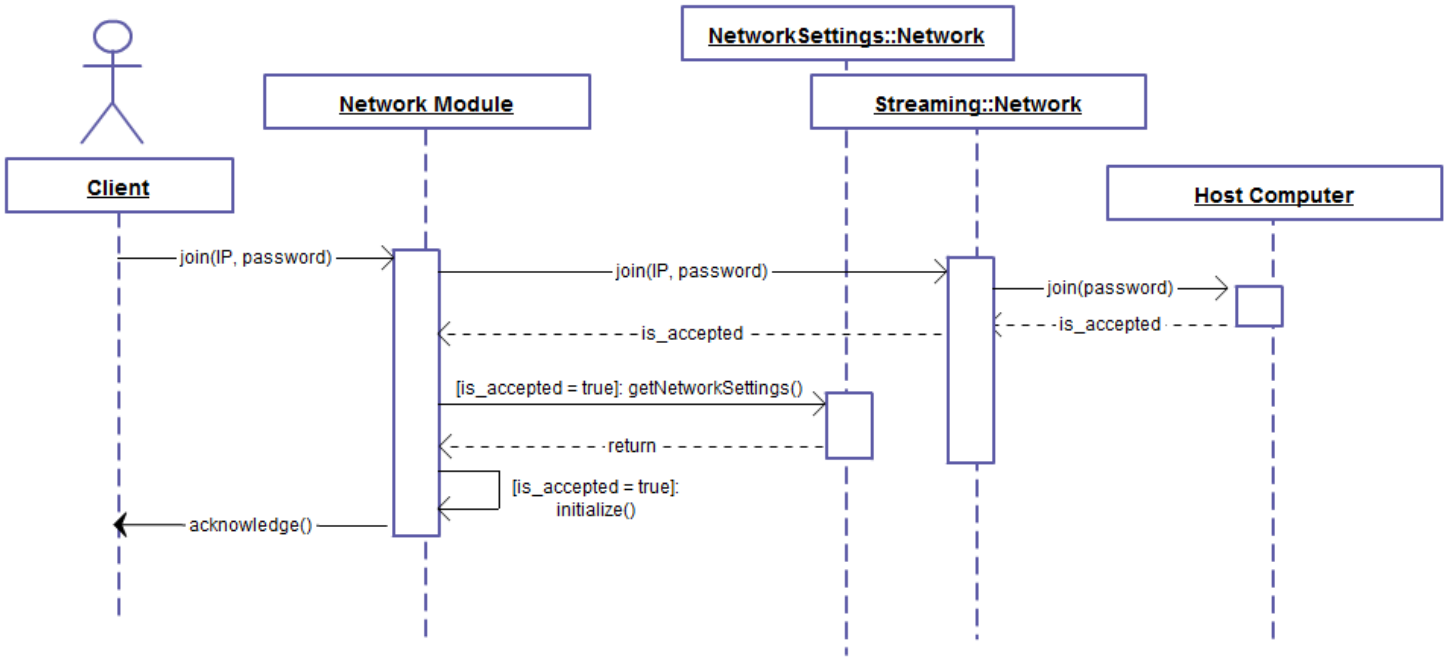


Figure 20: Join System Function

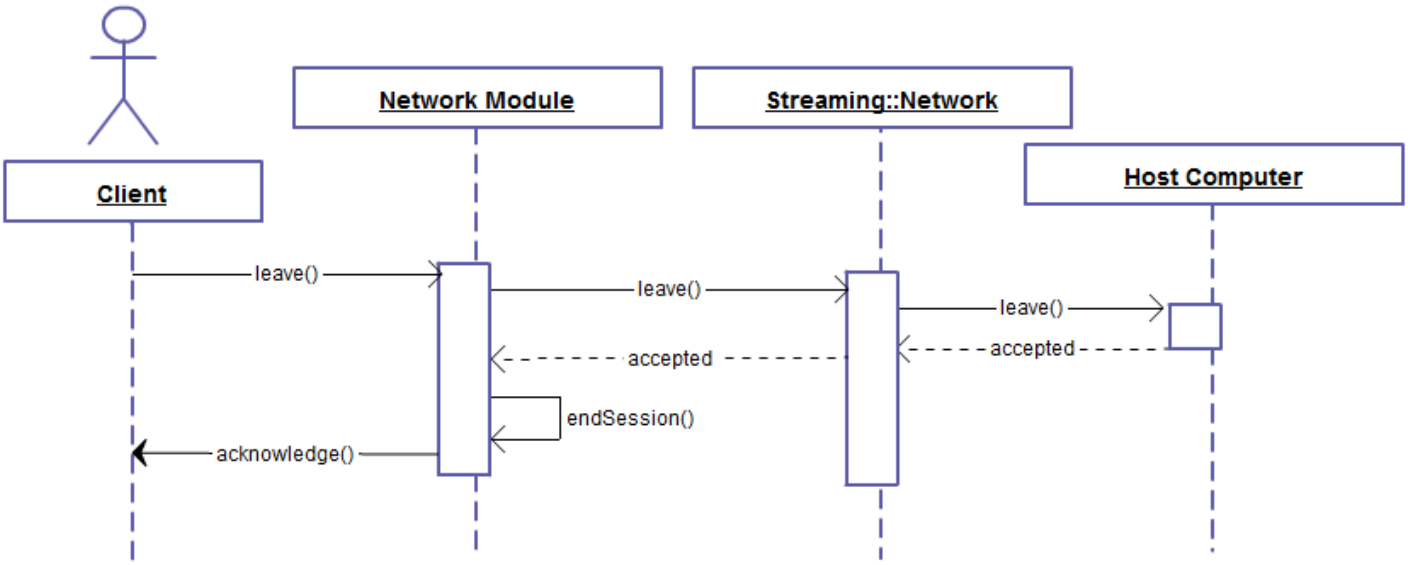


Figure 21: End Session System Function

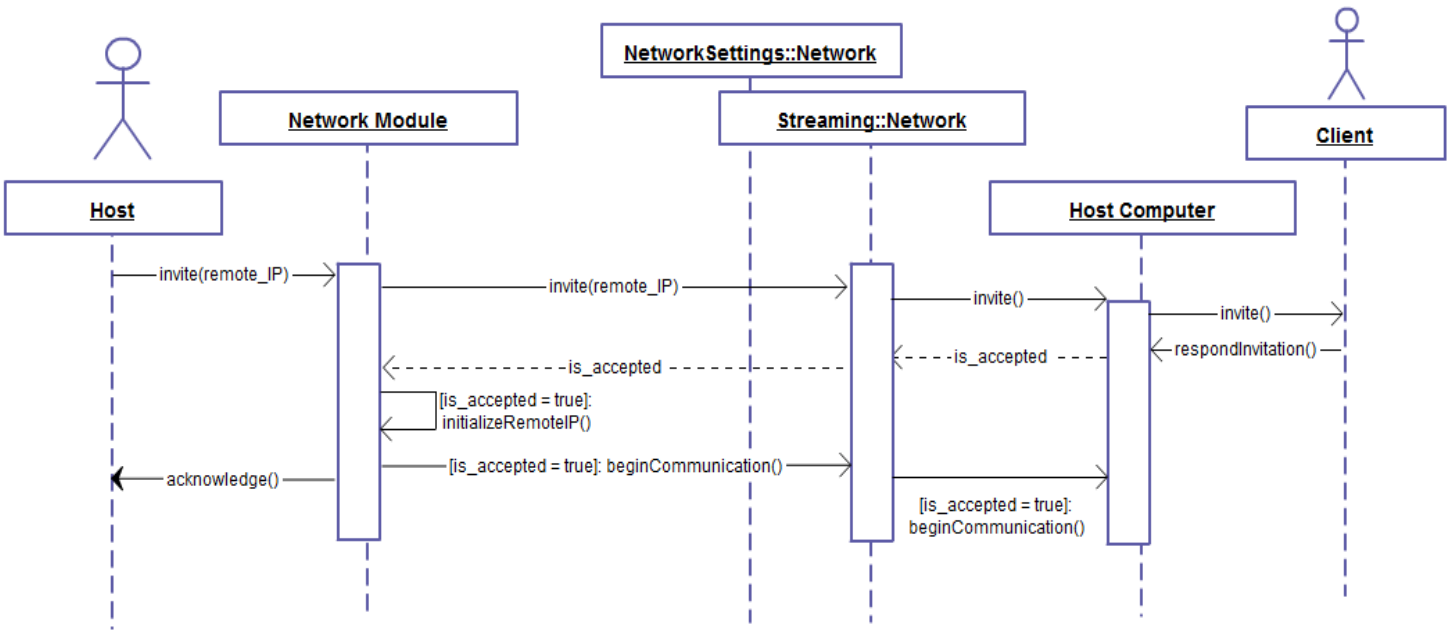


Figure 22: Invite System Function

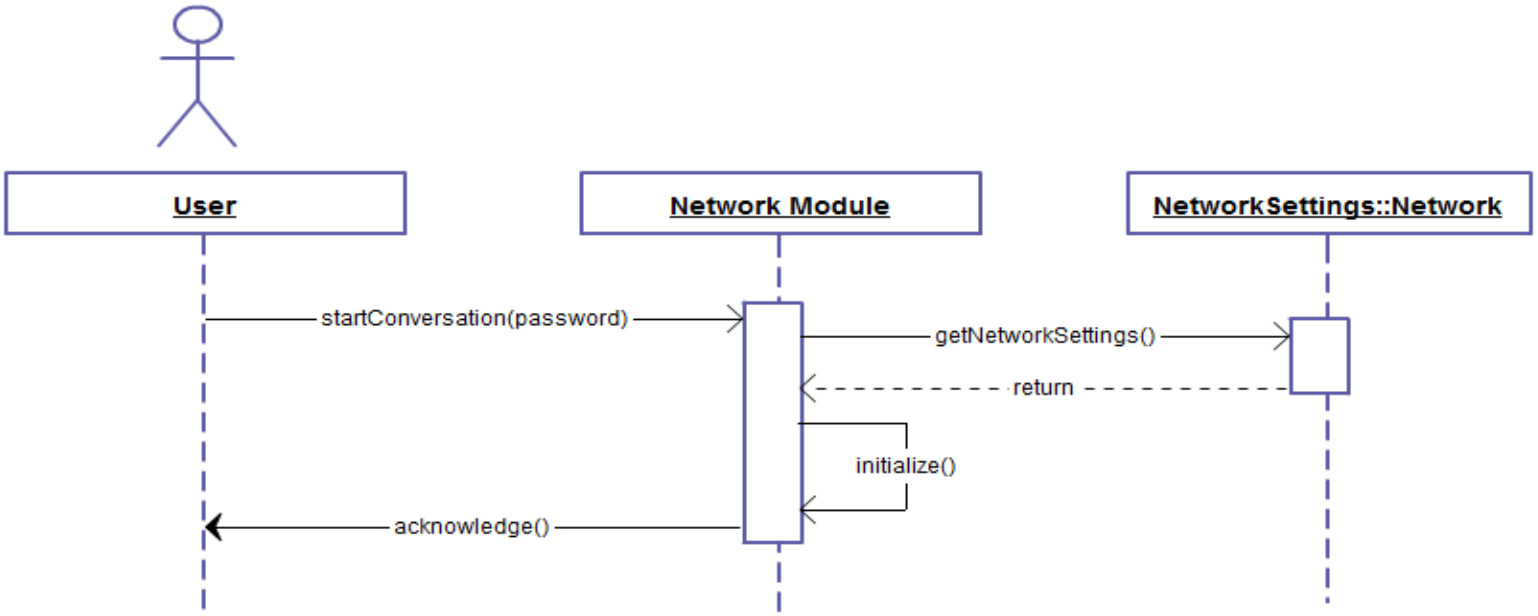


Figure 23: Start Conversation System Function

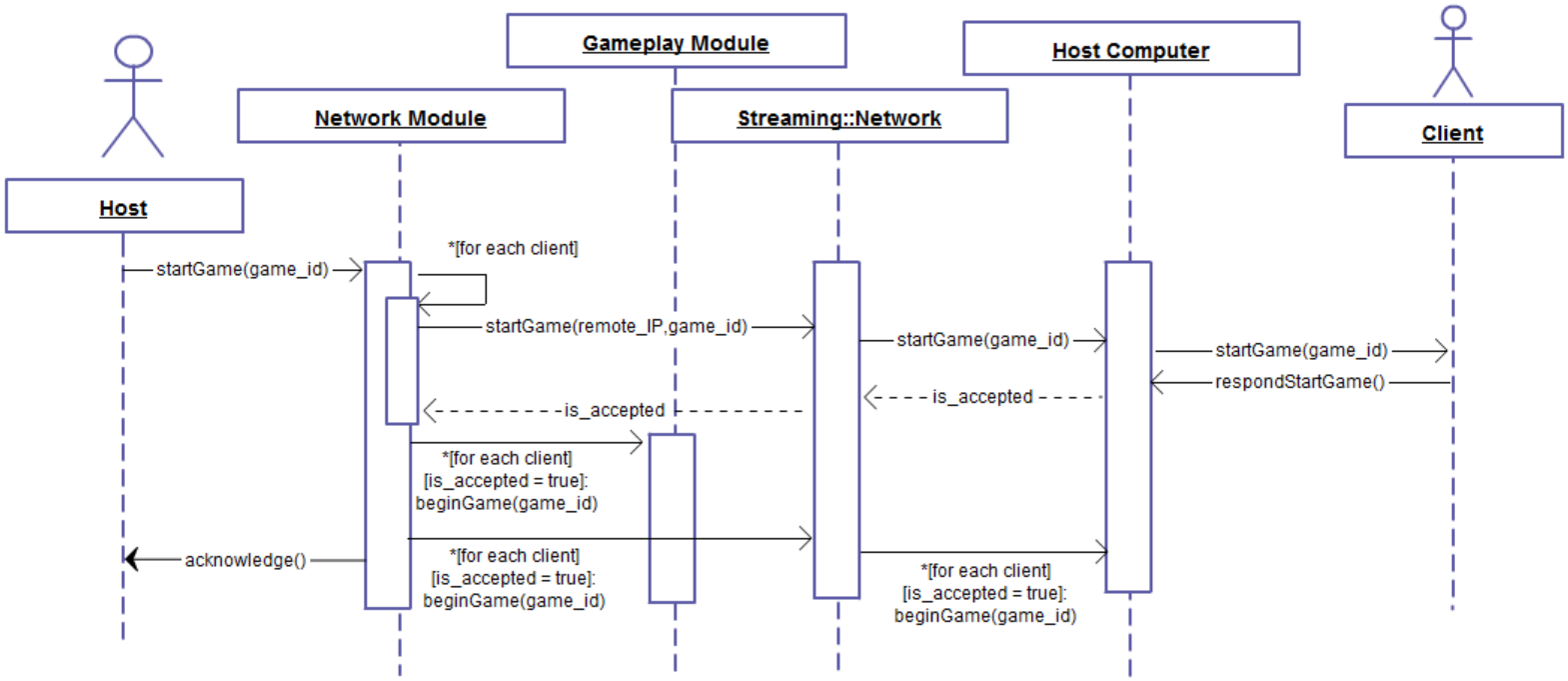


Figure24: Join System Function

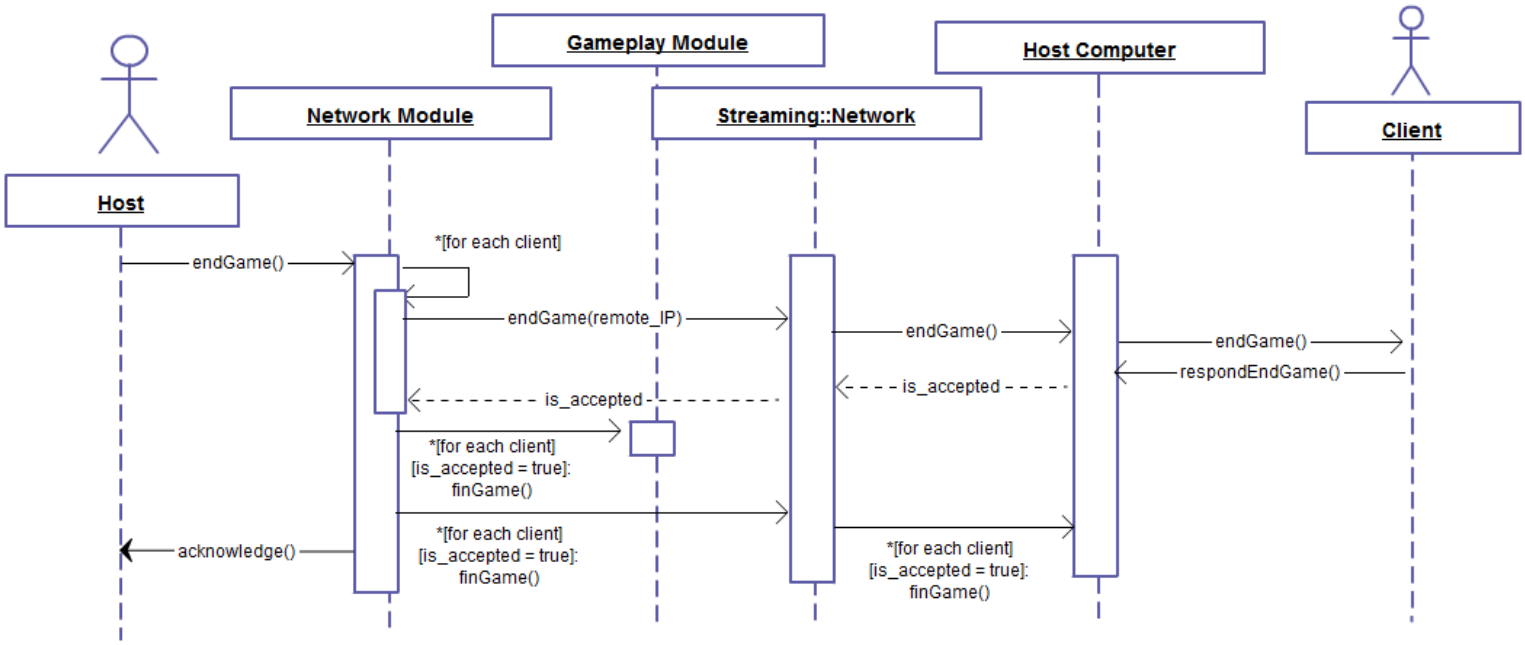


Figure 25: End Game System Function

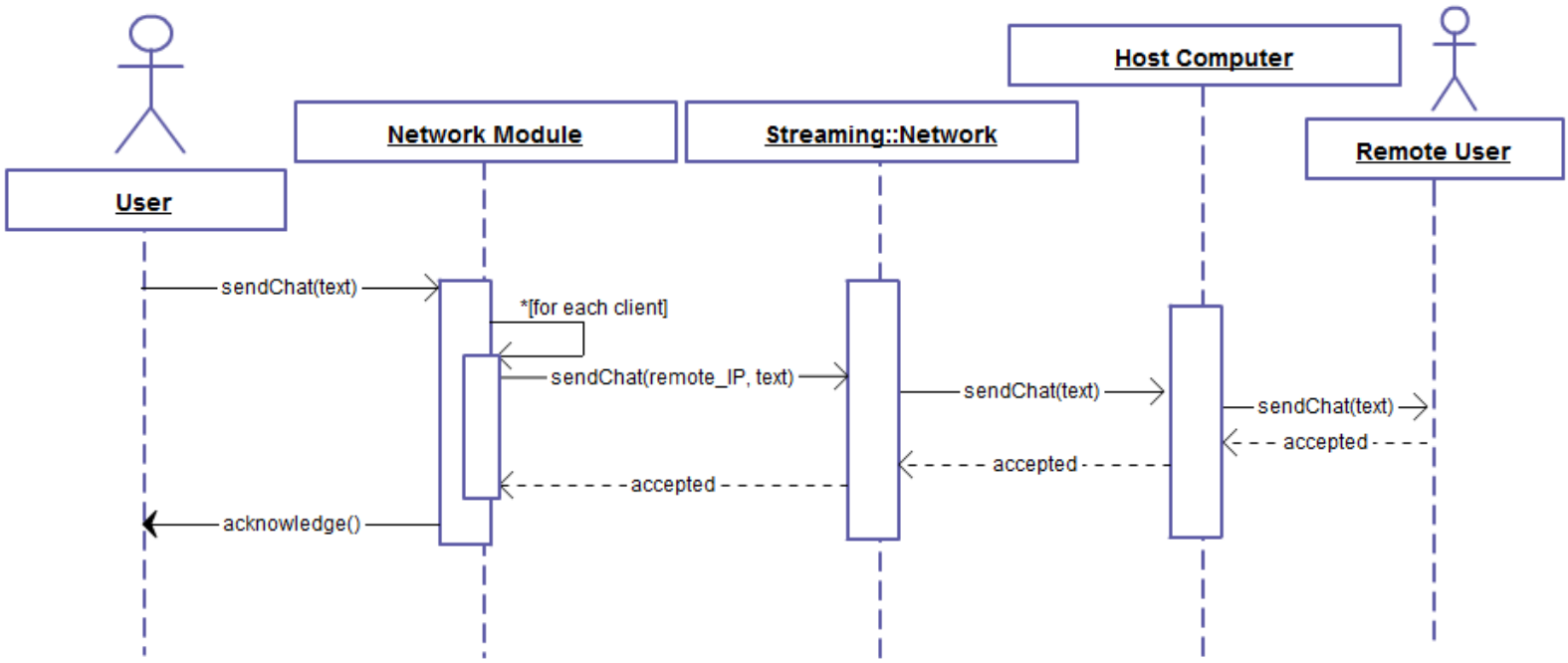


Figure 26: Send Chat Message System Function

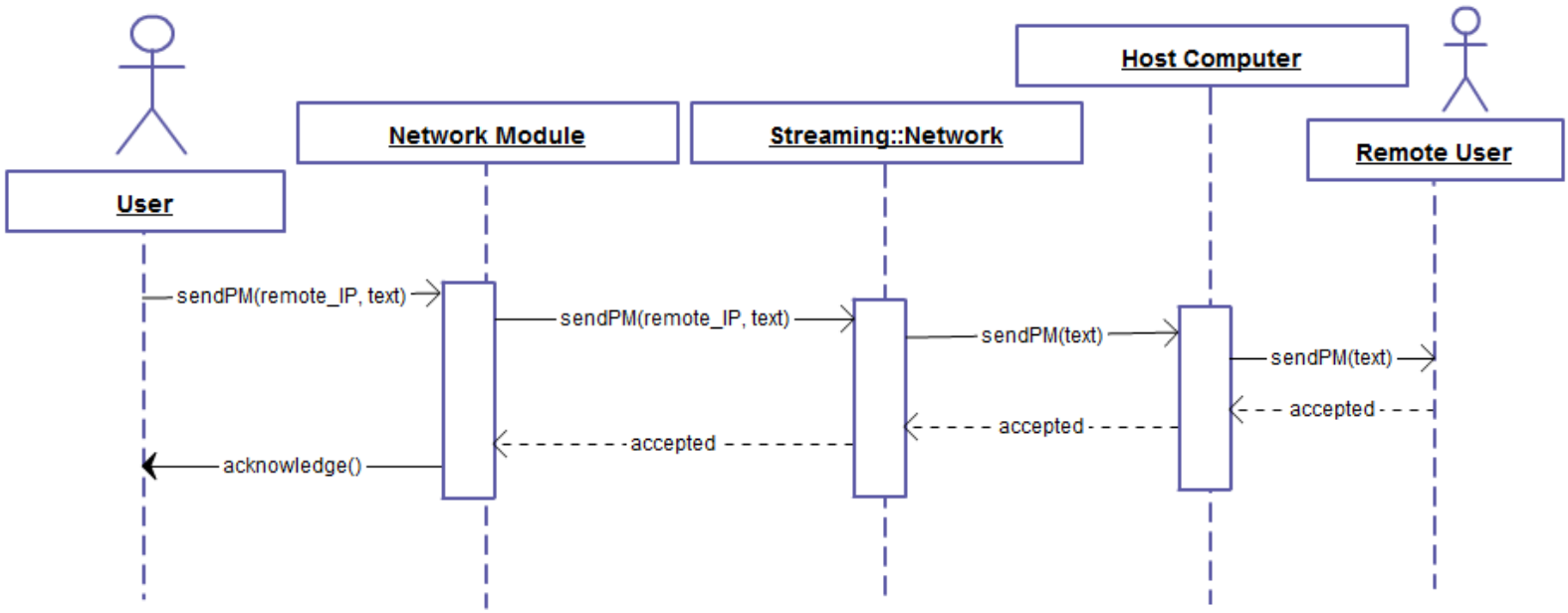


Figure 27: Send Private Message System Function

6 User Interface Design

6.1 Overview of User Interface

For our software, the type of user interface that will be implemented is a graphical user interface that will provide users with effective operation on the software. While designing the GUI, several criteria are taken into consideration to increase the usability and user friendliness of the GUI as much as possible. For instance, the system should always keep users informed about their state through appropriate feedback written in a suitable part of the GUI. Moreover, in case of an error occurs, GUI helps users to recognize, diagnose and recover from errors. In addition to this a help documentation will be available through the GUI menu. Additionally, system uses the platform conventions, i.e., the system does not oblige users to wonder whether different words, situations, or actions mean the same thing.
The system functions designed to answer all the needs of the users are listed below. They are mainly grouped into two: One group of functions stands for general operations, and the other includes the system functions

6.1.1 General System Functions

- ***Select Game***

Users are able to select the games that have been embedded into system already. The system function *Select Game* initializes the system state according to the chosen game. The properties such as, the number of users, the game board interface, etc, are set depending on selected game via the *Select Game* system function.

- ***List Existing Conversations***

A user, in case of not he/she does not prefer to create his/her conversation, can join existing conversations. The *List Existing Conversations* system function displays the sessions that have already been created.

- ***Start Conversation***

It is used to start a conversation. User using this functionality turns out to be *Host.*

- ***Join***

User may prefer to join an existing game instead of creating ones on her/his computer. *Join* system function handles that case during system progress.

- ***Invite***

Additional users may be invited to an existing conversation. The invitation can be sent by the host to the users who have not joined any game session yet.

- ***Respond to Invitation***

Invited user may or may not attend to the game to which she/he has been invited. *Respond to Invitation* system function returns the information whether or not the invited user will join the game.

- ***Set Minimum Number of Players***

Sets the required number of players value to pre-determined default value.

- ***Get Minimum Number of Player***

This system function returns the default number representing the minimum number of players in order for the host to start the game.

- ***Set Maximum Number of Players***

By sending invitations to users or accepting the demands of the users who wants to join the game, the maximum number of the players can be changed. *Set Maximum Number of Players* system function handles the situation.

- ***Get Maximum Number of Players***

This system function returns the current maximum number for a particular conversation.

- ***Open up Camera View Window for the User***

For each newcomer user a subwindow where the user's camera view is displayed is opened up. The size and orientation of this window is determined according to the number of users who have joined the game at that time.

- ***Leave Conversation***

Users, except the host, can leave the session they have been joint. Hosts are not allowed to leave the session unless they are not received the approval from the other participants.
More detailed explanations given in the related parts.

- ***End Conversation Demand***

In case of the user who wants to leave the conversation is the host for the current conversation, he/she has to make the others know that the session will end. The *End Conversation Demand* system function operates together with another system function, namely *Respond to End Conversation Demand.*

- ***Respond to End Conversation Demand***

Respond to End Conversation Demand system function returns the information whether or not participants approved the host's demand to end the conversation.

- ***Start Game***

After the conversation is established between all participants by joining to existing conversation or accepting the invitations, users may start to play. *Start Game* system function can only be used by the *Host* of conversation.

- ***Respond to Start Game Demand***

Before the game is started by the *host*, the *clients* respond him/her to give the information whether they will join or not. Depending on the result of this system function, a new game is started or not.

- ***End Game***

End Game system function enables the host to terminate the current game. This system function requires the approval of the end game demand by the clients.

- ***Respond to End Game Demand***

Like the *Respond to Start Game Demand* case, this system function also operates to inform the participants, in case the host demands to end the current game. The current game is ended according to the return value of this function.

- ***Send Chat Message***

There will be an area visible to all users, via which all the players can chat each other. *Send Chat Message* function provides group based communication in written case.

- ***Send Personal Message***

Users are capable of sending messages only to the ones that they choose. A call to *Send Personal Message* function prompts the chat feature for only the chosen users.

- ***View Chat Diary***

Logs of chats are kept by the system. The public logs can be accessed by all the users of a certain session, however, the personal chat logs are made visible only to the participants of that conversation, as it is supposed to be.

- ***Show Chat Details***

Some details, such as, when the conversation was done, who were the participants, etc, are kept by the system and these data are obtained via the *Show Chat Details* function call.

- ***Delete Chat***

The logs of chats are deleted when demanded via the *Delete Chat* system function.

- ***View About Us***

This parts includes information about our group **'Eggs on the Door'**, members, and the project.

- ***View Help Contents***

A manual will be made available to users to help them while using the software.

- ***Download Game***

This system function exists for the users to get the executable of the software.

- ***Communicate with Us***

Via this system function, the users communicate with us, the members of Eggs on the Door.

- ***Learn Self IP***

Undertaking the task, the host may need to know his/her IP to supply some users with this data and make them connect to his/her conversation. The function call *Learn Self IP* gives this opportunity to the users in some cases, if they demand this.

- ***Find in Text***

Users can search their logs and chats via *Find in Text* system call.

6.1.2 Taboo-Related System Functions

Since we are going to visualize our software on Taboo game, we indicate some system functions that are required to solve the Taboo- related issues. The following ones are called only in case of the chosen game is Taboo.

- ***Change Current Card***

Users are allowed to change the cards that they want to skip. Pressing the *change card* button calls this system function.

- ***Show Remaining Time***

This system function is periodically called after the stopwatch is set by a user, which indicating that the time is started for the player trying to explain the word.

- ***Pause Stopwatch***

In case of users press the pause button, this system function is called in order to keep the remaining time for the user.

- ***Resume Stopwatch***

When the pause button has already pressed ones, meaning that the time is paused for that player's turn, pressing that button again calls *Resume Stopwatch* system function to start the stopwatch where it has been stopped.

- ***Mark Card as Correct***

When one of the teams correctly guesses the word that one of their players has been trying to depict that card is needed to be marked as correct in order to be added to the points of the team.

- ***Mark Card as Taboo***

In case of the player depicting the word written in the current card, uses one of the taboo word accidentally, the *Mark Card as Taboo* system function is called by pressing *taboo* button.

- ***Activate Drawing Mode***

When the players turn is to depicting the word by drawing, the user presses *draw* button, calling the *Activate Drawing Tool* system function.

- ***Activate Puppet Performing Mode***

When the players turn is to depicting the word by using the puppet, the user presses *puppet* button, calling the *Activate Puppet Performing* system function.

- ***Activate Default Mode***

By pressing *default* button and thereby calling the *Activate Default Mode* system function, the player exits the puppet or the drawing mode.

- ***Drag Puppet***

After activating puppet performing mode, the *Drag Puppet* system function is called when the puppet is moved by the player.

- ***Draw***

Draw system function is called in order to use the functionality in drawing mode.

6.2. Screen Images

Figure 28: Initial window

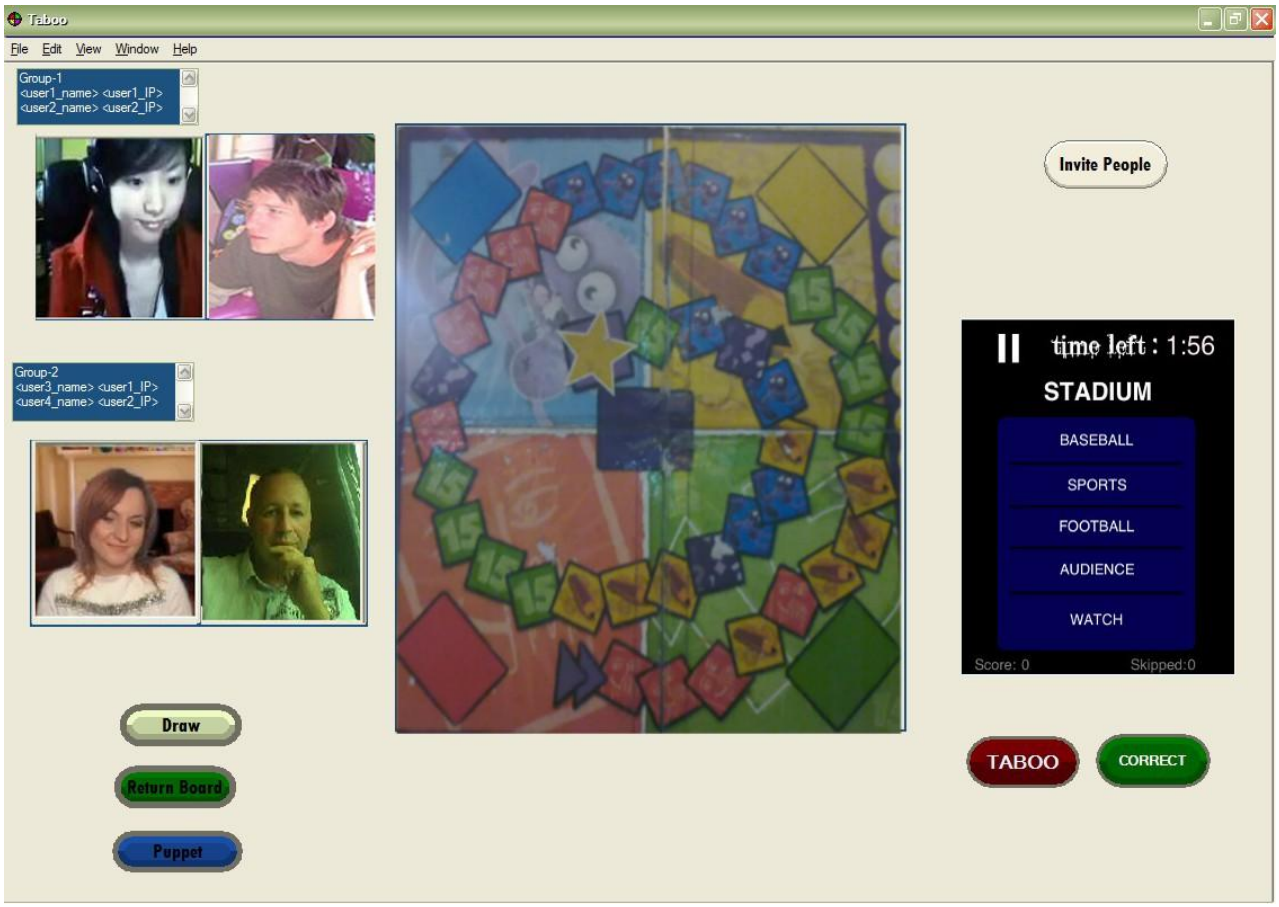


Figure 29: Taboo Game GUI

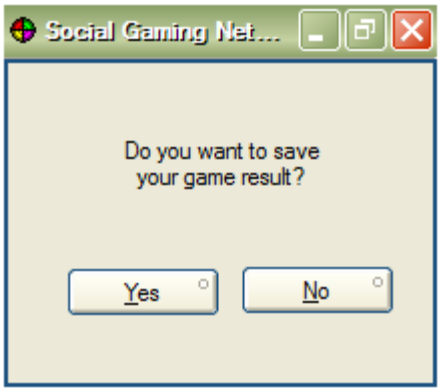


Figure 30: Save Game Results GUI

Figure 31: Save Chat Results GUI

6.3. Screen Objects and Actions

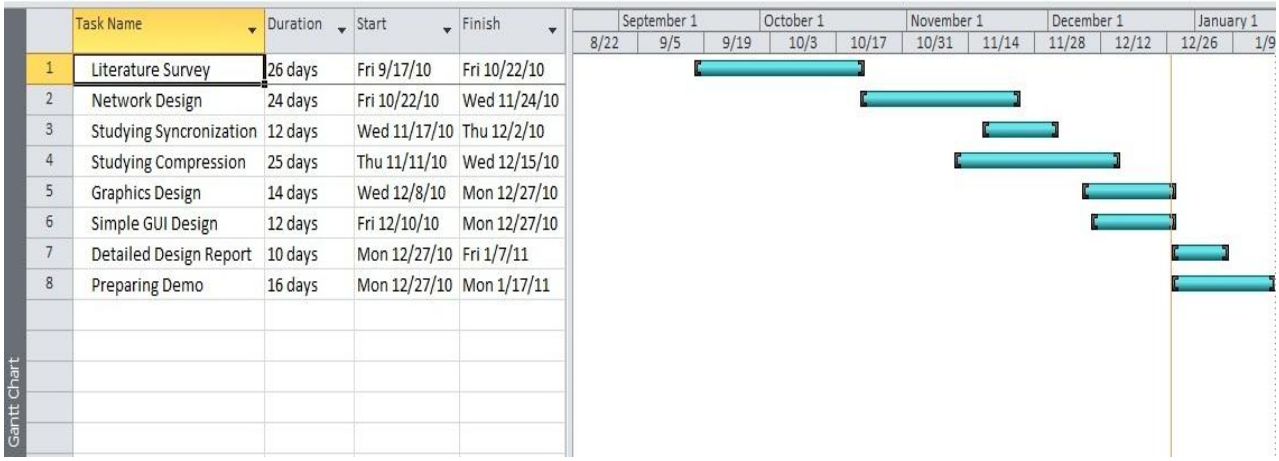
By system functions expressed above the GUI buttons are controlled. In that sort, the objects composed of the GUI parts.

7 Libraries and Tools

Public MSU video filters
MSU video quality measurement tools
C# streaming libraries
.Net streaming libraries
C# video streaming libraries

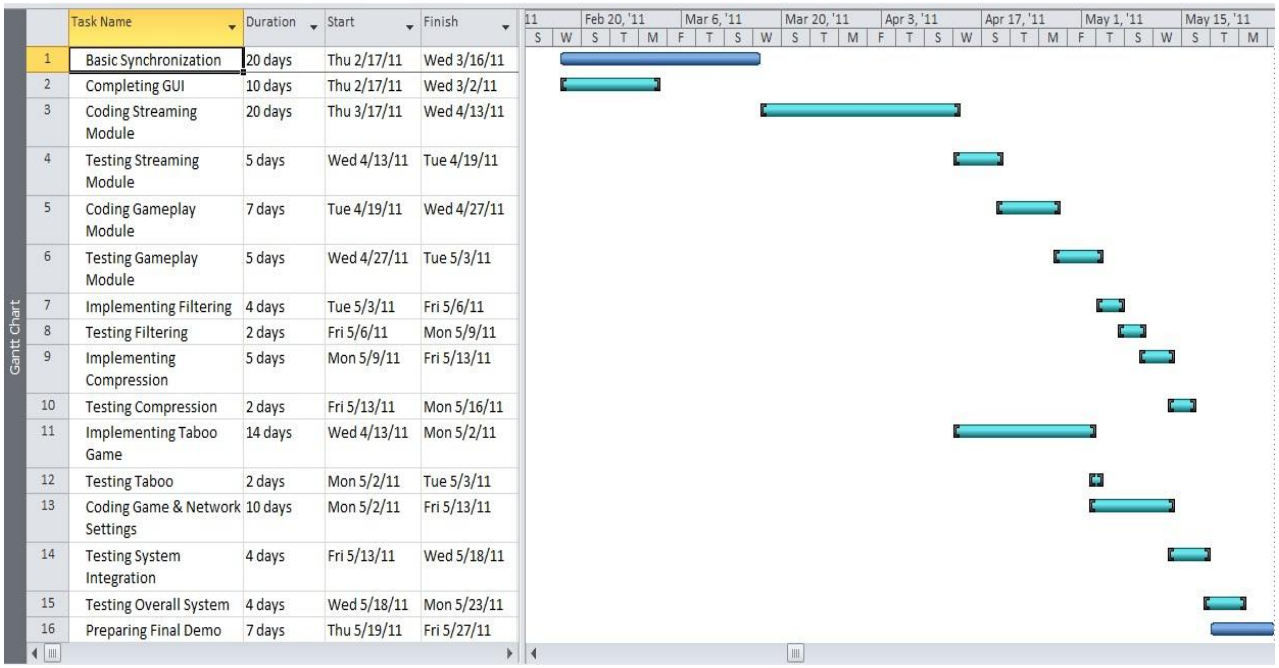
8 Time Planning (Gannt Chart)

8.1 Term 1 Gannt Chart



	Task Name	Duration	Start	Finish
1	Literature Survey	26 days	Fri 9/17/10	Fri 10/22/10
2	Network Design	24 days	Fri 10/22/10	Wed 11/24/10
3	Studying Syncronization	12 days	Wed 11/17/10	Thu 12/2/10
4	Studying Compression	25 days	Thu 11/11/10	Wed 12/15/10
5	Graphics Design	14 days	Wed 12/8/10	Mon 12/27/10
6	Simple GUI Design	12 days	Fri 12/10/10	Mon 12/27/10
7	Detailed Design Report	10 days	Mon 12/27/10	Fri 1/7/11
8	Preparing Demo	16 days	Mon 12/27/10	Mon 1/17/11

8.2 Term 2 Gannt Chart



	Task Name	Duration	Start	Finish
1	Basic Synchronization	20 days	Thu 2/17/11	Wed 3/16/11
2	Completing GUI	10 days	Thu 2/17/11	Wed 3/2/11
3	Coding Streaming Module	20 days	Thu 3/17/11	Wed 4/13/11
4	Testing Streaming Module	5 days	Wed 4/13/11	Tue 4/19/11
5	Coding Gameplay Module	7 days	Tue 4/19/11	Wed 4/27/11
6	Testing Gameplay Module	5 days	Wed 4/27/11	Tue 5/3/11
7	Implementing Filtering	4 days	Tue 5/3/11	Fri 5/6/11
8	Testing Filtering	2 days	Fri 5/6/11	Mon 5/9/11
9	Implementing Compression	5 days	Mon 5/9/11	Fri 5/13/11
10	Testing Compression	2 days	Fri 5/13/11	Mon 5/16/11
11	Implementing Taboo Game	14 days	Wed 4/13/11	Mon 5/2/11
12	Testing Taboo	2 days	Mon 5/2/11	Tue 5/3/11
13	Coding Game & Network Settings	10 days	Mon 5/2/11	Fri 5/13/11
14	Testing System Integration	4 days	Fri 5/13/11	Wed 5/18/11
15	Testing Overall System	4 days	Wed 5/18/11	Mon 5/23/11
16	Preparing Final Demo	7 days	Thu 5/19/11	Fri 5/27/11

9 Conclusion

The initial design report is one of the crucial steps of our project. The designs we have made and wrote in the SRS document are strengthened by the theoretical and practical information. The difficulties we may encounter during the process and possible solutions are determined. In accordance, we shaped our project and chosen to use technologies and algorithms which we believe to be helpful and efficient to deal with those problems. Moreover, the IDR is like a guideline for further detailed design of our project. As we made clear the main aspects of our project choose some algorithms and software technology to for development of our project.

To conclude, this initial design report will be the guideline of our project this year.