CHALMERS

See, What I See (SWIS)

Design and Implementation of a Virtual Living Room in Cyberspace

Master of Science Thesis in Software Engineering

Aily Raesi

Department of Computer Science

Division of Software Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden, 2009

Abstract

People invite friends to their home to watch their favorite movie. They gather around a TV and watch the movie while having a conversation. Watching movie and television together is a common social activity, but web video does not support this synchronous watching and conversations. Nowadays videos are watched alone without co-presence and social interactions are confined to asynchronous comment threads left by family and friends. Social networking services and internet applications are trying hard to make it easier for individuals to communicate when they are apart from each other. These services and facilities remove geographical distances between people since they are able to talk and see each other at any moment. Different applications have variety of services to keep people updated about their friends by different kinds of media. In the era of multimedia communication a rich interactive application which simulates a living room in cyberspace can make the communication much easier and interesting. In this thesis we build a service which simulates a virtual living room in the cyberspace. In this service people will be able to plan and watch a movie in someone's virtual living room. They will also be able to talk, share and interact using different communications media such as text, voice or video. This application is implemented to be used in state of art web browser without downloading any additional plug-in, except a flash viewer, which is widely available in most of the computer and devices.

Table of Contents

Table of Figures

Chapter 1

1. Introduction

1.1 Overview

Watching movie and television together is a common social activity, but web video does not support this synchronous watching and interactions. Nowadays videos are usually watched alone without presence of friends and social interactions are confined to asynchronous comment threads left by family and friends.

Social networking services and internet applications are trying hard to make it easier for individuals to communicate when they are apart from each other. These services and facilities remove geographical distances between people since they are able to talk and see each other at any moment. Different applications have variety of services to keep people updated about their friends through different kinds of media. In the era of multimedia communication a rich interactive application which simulates a living room in cyberspace can make the communication much easier and interesting. People will be able to plan and watch a movie in someone's virtual living room. They will also be able to talk, share and gossip using different communications media such as text, voice or video. Such an application can be implemented to be used in state of art web browser without downloading any additional plug-in, except a flash viewer, which is widely available in most of the computers and devices.

1.2 Problem Statement

Communication and socializing has always been a need for human being, and despite of all the different communication devices that are available today, people are getting far from each other day by day. By growing the number of video sharing web sites, this is preferred by people to watch the movie they like at the time they want. This makes people to spend more time alone and leave the habit of sitting and watching TV or movie together. New Technologies can make it possible to bring this habit back. This goal can be achieved by building a rich interactive application which meets all the essential requirements for making everybody feels at the same place.

In such kind of application the host user selects a video clip/movie and initiates a session. When the session is established everybody has a same synchronized view as the host user. Voice, video and text chat are also available to initiate different kind of interactions since each of them has its own pros and cons. Video chat enables implicit presence information between users, since by text chat, all emotions, and physical actions (such as leaving your computer) must be conveyed

explicitly through text messages. In video chat, people see when someone is paying attention, laughing, or smiling regardless of what he or she types into the text chat. Such an environment can be simulated in cyber space with help of a real-time interactive multimedia application. Real-time characteristic of multimedia applications have made it sensitive to network issues such as packet loss, jitter and end to end delay. People are allowed to use audio/video to communicate with each other while watching a video movie/clip together. Timing consideration is important because real time audio/video interaction is highly delay-sensitive. IP protocol deployed in the internet today offers a best-effort service to all the datagram it carries. Internet does not make any promises about end to end delay and the variation of packet delay within a packet stream. Since TCP [7] and UDP [8] run over IP, it follows that none of this transport protocols provides any delay guarantees to invoking applications. Since there is no special effort to deliver packets in a timely manner, it is a challenging problem to develop a successful application. While trying to have a successful application, during peak traffic periods, performance may be unsatisfactory, especially when involved links are congested.

Invitation methods, providing the shared content, synchronizing the player across different users/viewers and providing the interaction mediums are different concept that should be handled in this application. The process of choosing a video clip/movie to share is challenging. Online video sharing website is a popular source to offer video or small video clips. Video sharing websites have different policies to deliver the content, therefore each online video sharing website should be explored and compensated with a solution. Beside the technical issues, legal issue should be considered in redirecting (if necessary) the content from other website. Synchronizing of the online shared video is highly sensitive to network condition. Different geographical location of users, and different internet speeds and QoS, Quality of Service, available to the users affects directly on the synchronization process. Users with different QoS have different bandwidth available which causes buffering for some users while others are watching without any disruption. All these issues make the synchronization a challenging process.

In this thesis such a system has been investigated and a demo was implemented. The implemented demo shows the deficiencies in the system requirements. User preferences are more obvious and clear while the demo is available to test. Different ways to interact, number of people present on the same session and the methods users prefer to use to choose the shared media are more likely to finalize by having the demo.

Defining a feasible requirement list and scenarios, implementing a basic demo based on the defined requirement list, discovering the issues that application face while distributed on the network, testing different scenarios with the demo application and exploring the developed application model are the steps followed in this thesis.

1.3 Goals

The goals of this thesis is to -

- explore the ways video clip/movie can be shared.
- find the states of art technologies available for this kind of service.
- describe the architectural requirements to realize this kind of service.
- estimate the network QoS required for this kind of service.
- synchronize every user's view so that they can view the same video frame at the same time.
- implement the basic demo which meets the system requirements.

1.4 Scope

SWIS (See What I See) is the service which is delivered in this thesis. This service works in web browser and is compatible with any browser for windows such as Internet Explorer, Mozilla Firefox, and Opera. Flash Player plug-in needs to be downloaded on the web browser, because this webpage contains action scripts [21] which run in the flash player. The video movie/clip file should be in flv, a type of video file format which is defined by Adobe Flash Media, and the client needs a microphone and webcam to use all the features present on the application. The user should be able to upload movie clip or share a link to a video clip. This service is able to handle a session with six people.

1.5 Research Methodology

The following structured research approach followed during this thesis work:

- A literature survey on media sharing applications, multimedia communication tools and web based event management tools.

- Make feasible application requirement list and scenarios.

- Select the best technologies and design the architecture.

- Implement a basic demo application of virtual living room according the requirement list.

- Test different scenarios with the demo application.

- Publishing the research outcome and future works.

1.6 Company

This thesis has been done in Ericsson multimedia research office located in Lulea, Sweden. This section does research on multimedia and communicating multimedia over the network and around 20 people work in this division.

1.7 Structure of the Thesis

In this book chapter 2 describes the background study and literature review. It discusses the implemented already available applications which use the synchronize player concept and the technologies used in this service. Chapter 3 outlines the design of SWIS service. This chapter explains different phases to develop this service. It also presents different scenarios considered while designing SWIS service. Chapter 4 describes the implementation of SWIS service. This chapter elaborates the implementation part with the help of diagrams and pictures. Chapter 5 summarizes the research work and proposes future works related to this subject

Chapter 2

2. Technology

SWIS is a real-time multimedia service. To implement this service, we need to explore different type of servers which support delivering media over the network. There are two methods for streaming audio/video over the internet, streaming server and progressive download. Section 2.1 describes these two approaches. While streaming server is chosen over progressive download in this service as it is described, we need to find the right streaming media present on the market. Adobe flash media is one of the most popular and powerful brands in the streaming media technology market. Section 2.2 gives a brief introduction on Adobe Flash Media. Section 2.3 explains different data transfer protocols present in this service and the protocols involved in transferring the multimedia are explained in more details. Finally section 2.4 describes the related works to this service that has been developed and their advantageous and pitfalls.

2.1 Video streaming technologies

Recent advances in computing technology, high-bandwidth storage devices and high speed networks have resulted more services of real-time multimedia over the internet. Delivering media over the internet has evolved from download mode to progressive download and streaming mode. In download mode user download the entire file and then plays the video file. Since users were suffering from long waiting time, new techniques have been developed to eliminate the waiting time for the whole file to be transferred. These techniques are progressive download and streaming servers [3].

2.1.1 Progressive download

In progressive download, stored audio/video resides on an ordinary web server which delivers the audio/video to the client over (Hyper Text Transport Protocol) HTTP [1]. The multimedia file is received and stored by the end user device. Client may begin playback of the media before the download is complete and when a specified amount of data becomes available to the local playback device, the media will begin to play. This particular amount of buffer is embedded into the file in the encoder setting by the producer and is imposed by the media player in additional buffer setting. The difference between progressive download and download-play (download the file completely and then play) relies on how the client functions and initiates playing as soon as some amount of data is buffered [2].

On the client side media player and browser are functioning together. The browser requests a metafile that contains information (for example, a URL and type of encoding, so that the appropriate media player can be identified) about the multimedia file. HTTP response message that carries the metafile contains a content-type header line which indicates the audio/video application [3]. The browser examines the content-type header line and launches the associated media player and passes the metafile to the media player. Media player request the file and HTTP server sends the multimedia file over HTTP. These steps

are illustrated in Figure 2.1. This makes it clear that this intermediate, reading the content-type header, step is needed for the browser to launch the appropriate media player [4].

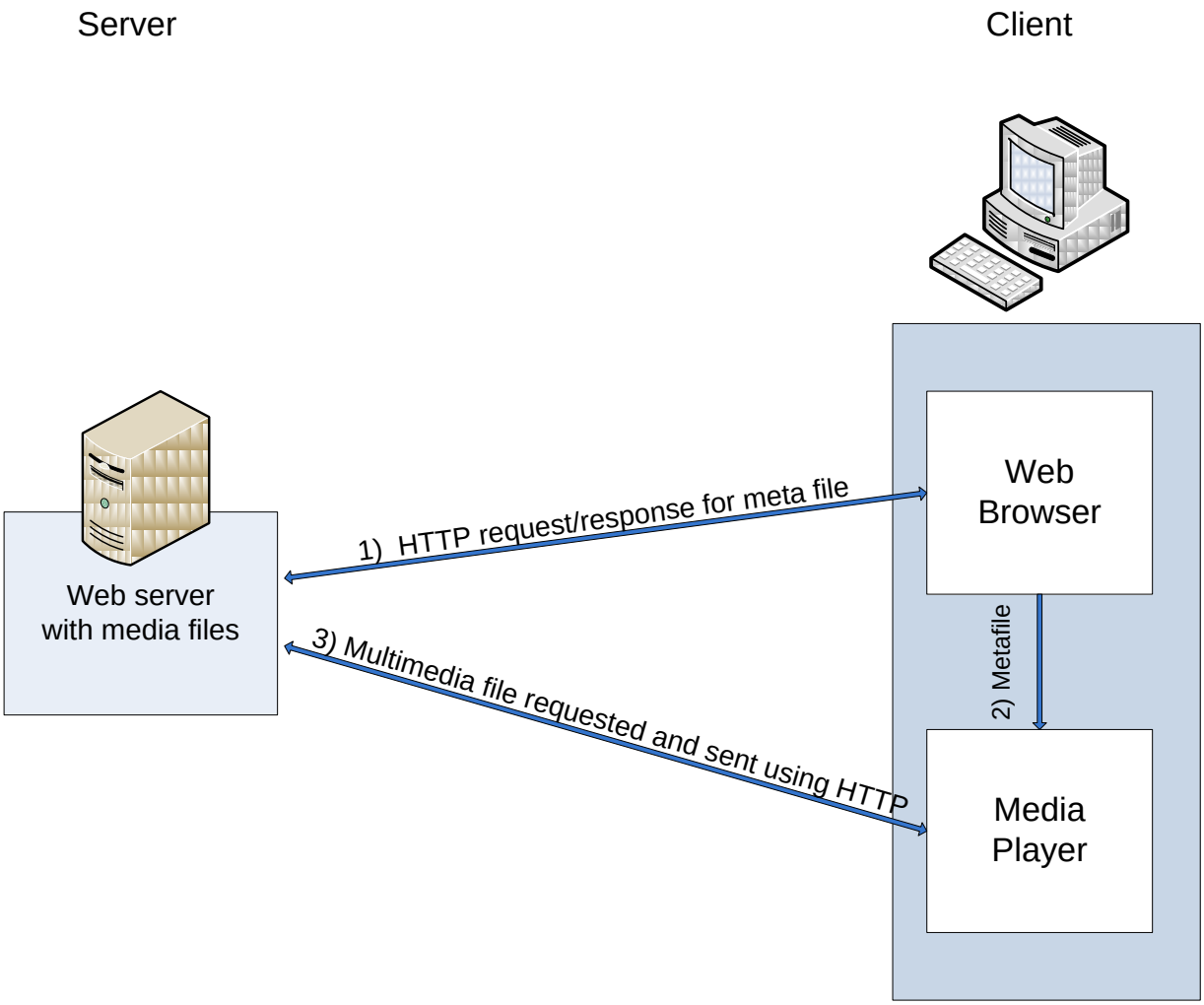

Figure 2- 1: Web server sends audio/video directly to the media player

2.1.2 Streaming server

A streaming server is a specialized server for delivering multimedia file over the internet while it could be a proprietary or public domain streaming server. Audio/video can be sent over HTTP/TCP or over UDP using application-layer protocols. In the streaming server the data is actively and dynamically sent to the client. It means that streaming server sends the content at the same data rate associated with the compressed audio and video streams. The server and the client communicate during the delivery process and the streaming media server is able to respond to any reaction from the client. The video content is not required to be downloaded completely, and is being played out while data are being received and decoded [2].

Streaming servers need two servers, one server to serve the webpage and the other to serve the audio/video file. The two servers can run on two distinct end systems or on the same end system. The media player requests the file from a streaming server instead of Web server. The media player and streaming server are able to interact using their application specific protocols [19].

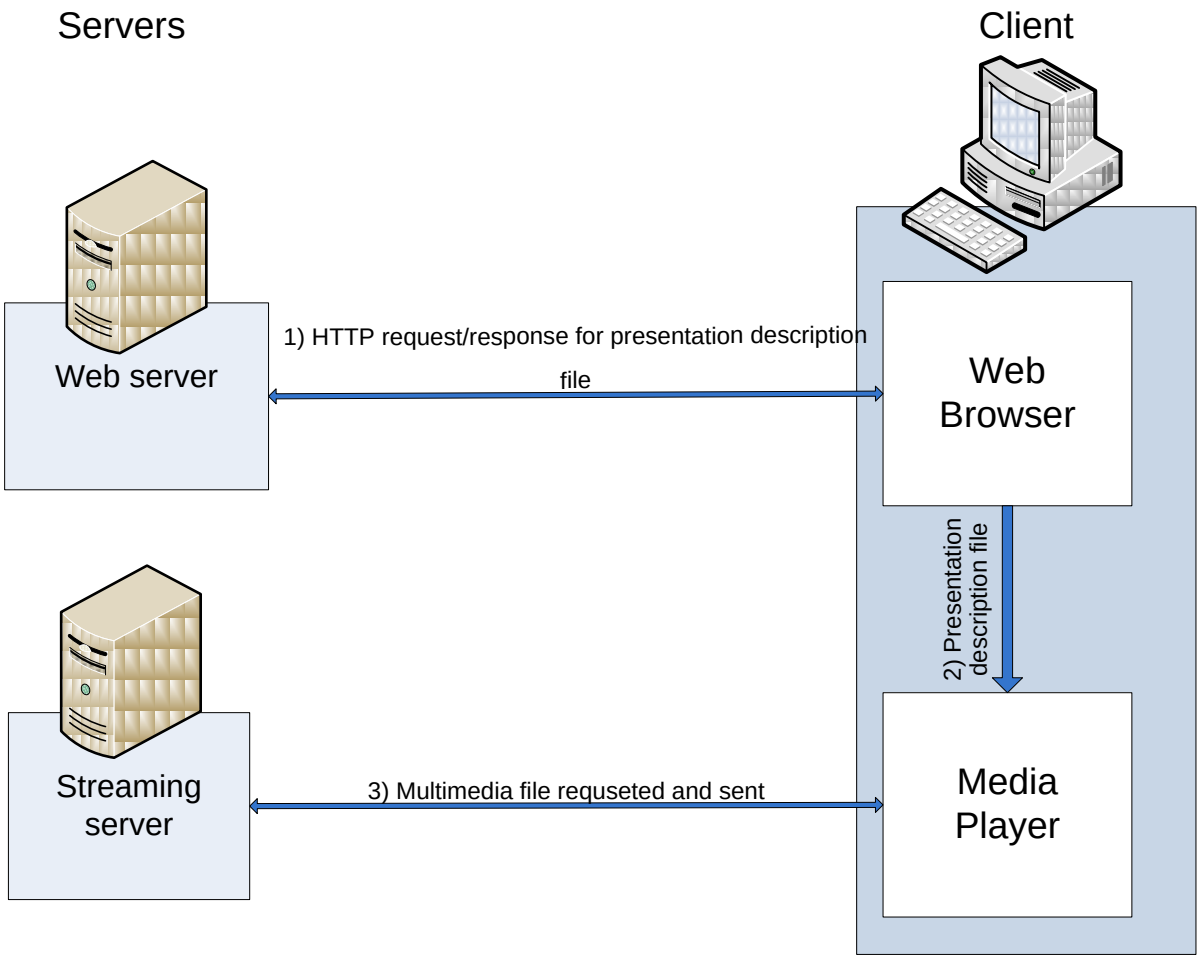

Figure 2- 2: Streaming from a streaming server to a media player

The audio/video could be sent over UDP at a constant rate equal to the encoded rate of the audio/video. Client decompresses the audio/video as soon as it receives compressed audio/video and plays it back. Sometimes media player delays playout for two to five seconds to remove network-induced jitter by placing the compressed media that it receives from the network into a client buffer as shown in Figure 2.3. The fill rate is equal to drain rate (the encoded rate of the audio/video) except when there is packet loss. The audio/video could also be sent over TCP. The server sends the media file into TCP/IP socket as fast as it can and the client reads from TCP socket as fast as it can and places the compressed video into the media player buffer. The fill rate fluctuates with time because of window flow control and congestion [4] control. If client buffer is large enough to hold the media file, then TCP will make use of all the bandwidth available to the connection, so that fill rate can become larger than drain rate.

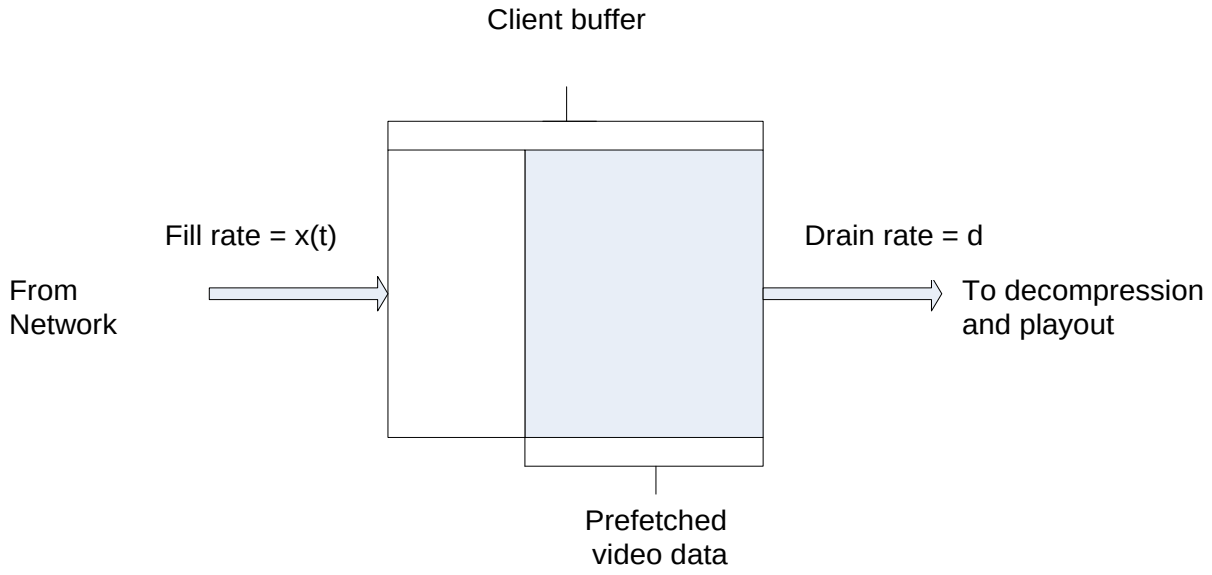

Figure 2- 3: Client buffer being filled at rate x(t) and drained at rate d

Streaming server uses the network bandwidth more efficiently and results in better audio/video quality to the user while supports large number of users [24]. In SWIS application streaming server is used to deliver the media over the network.

2.1.2.1 Adobe Flash Media

The Adobe Flash Media server family is one of the most popular products in streaming video industry. In this section we give a brief description on Adobe Flash Media.

- Media format: Video file formats defined by Adobe Flash are FLV and F4V [5].
  - Supported media types in FLV file format are:
    - Video: On2 VP6, Sorenson Spark (Sorenson H.263), Screen video, H.264
    - Audio: MP3, ADPCM, Linear PCM, Nellymoser, Speex, AAC, G.711 Supported media types in F4V file format:
    - Video: H.264
    - Audio: AAC, HE-AAC, MP3

- Interactive programming: MXML and ActionScript are two different languages to produce SWF files. The Adobe SWF format is a license-free specification. ActionScript is a scripting language which is based on ECAMScript the same standard for javascript. ActionScript provide sefficient programming of Flash applications covering everything from basic animations to complicated, data rich, interactive application interfaces [21].

MXML is an XML markup language which is used to layout components in user interface. It is used for non-visual aspects of an application like accessing to the data sources on the server [5], [20].

- Playback browser plug in: Adobe flash player is a cross-browser, cross-platform and cross-device standard for delivering web application. Flash player runs SWF files that are created by Adobe flash. Flash Player 9 and above supports HD video quality and streaming or progressive playback [22].

- Data transfer protocol: Adobe flash uses RTMP and HTTP protocols to transfer data. RTMP protocol is a proprietary protocol which is developed by adobe flash systems. RTMP transfer data over the internet between a flash player and server. Flash Media Server offers different configurations of RTMP:

  - **RTMP**: Standard, unencrypted Real-Time Messaging Protocol.

  - **RTMPT**: RTMP which is tunneled over HTTP; the RTMP data is encapsulated as valid HTTP data.

  - **RTMPS**: RTMP sent over a Secure Sockets Layer (SSL). SSL is a protocol that provides secure TCP/IP connections.

  - **RTMPE**: An enhanced and encrypted version of RTMP. RTMPE is faster than SSL and does not require certificate management as SSL does. The key benefits over SSL (RTMPS) are performance, easiness of implementation, and restricted impact on server capacity.

  - **RTMPTE**: RTMPE which is tunneled over HTTP.

- Streaming server: Adobe flash media server is a proprietary data and media server developed by Adobe Systems. This server works with the Flash Player runtime to create media driven, multiuser rich internet applications. Flash Media Server acts like a hub, Flash based applications connect to the Flash server using RTMP. The server allows sending and receiving data to and from the connected users with Flash player installed. Connected clients are able to make Remote procedure calls on the server-side and the server is able to call methods on specific clients. A SharedObject can be described to synchronize complex data structures and call remote methods on several clients in one go by having clients subscribe to a shared object. Server and flash client transfer ActionScript objects across the NetConnection using Action Message Format [6]. A number of features that is offered by adobe flash are dynamic streaming, secure HD-quality video, delivery to mobile handset and devices and integrated HTTP server (ensure that the content is delivered even when RTMP delivery is not supported) [5].

In our demo we use Red5 server which is an open source flash server written in java, since it offers the required features in SWIS service while it is free and open source.

2.2 Protocols

A few protocols have been designed and standardized for communication among streaming servers and clients. Based on their functionalities, the protocols that are directly related to internet streaming video with adobe flash server can be categorized as follow.

1) Network-layer protocol provides primary network service support like network addressing. The IP [25] provides the network-layer protocol for Internet video streaming.

2) Transport protocol offers end-to-end network transport functions for video streaming applications. Transport protocols include UDP [7] and TCP [8].

3) RTMP [9] and RTMP Chunk Stream [9] are two protocols that are used for transmitting real time data over the network.

I describe the protocol stacks for media streaming in Figure 2.4 to illustrate the relationship among the three types of protocols. For the data plane, at the sending side, the compressed video/audio data is retrieved and packetized at the RTMP layer. The RTMP-packetized streams provide timing and synchronization information, as well as sequence numbers. The RTMP-packetized streams are then passed to the RTMP Chunk Stream to break the packets into smaller packets and then RTMP Chunk Stream passes them to the UDP/TCP layer and the IP layer. The outcome IP packets are transported over the Internet. At the receiver side, the media streams are processed in the reversed manner before being presented. [9]

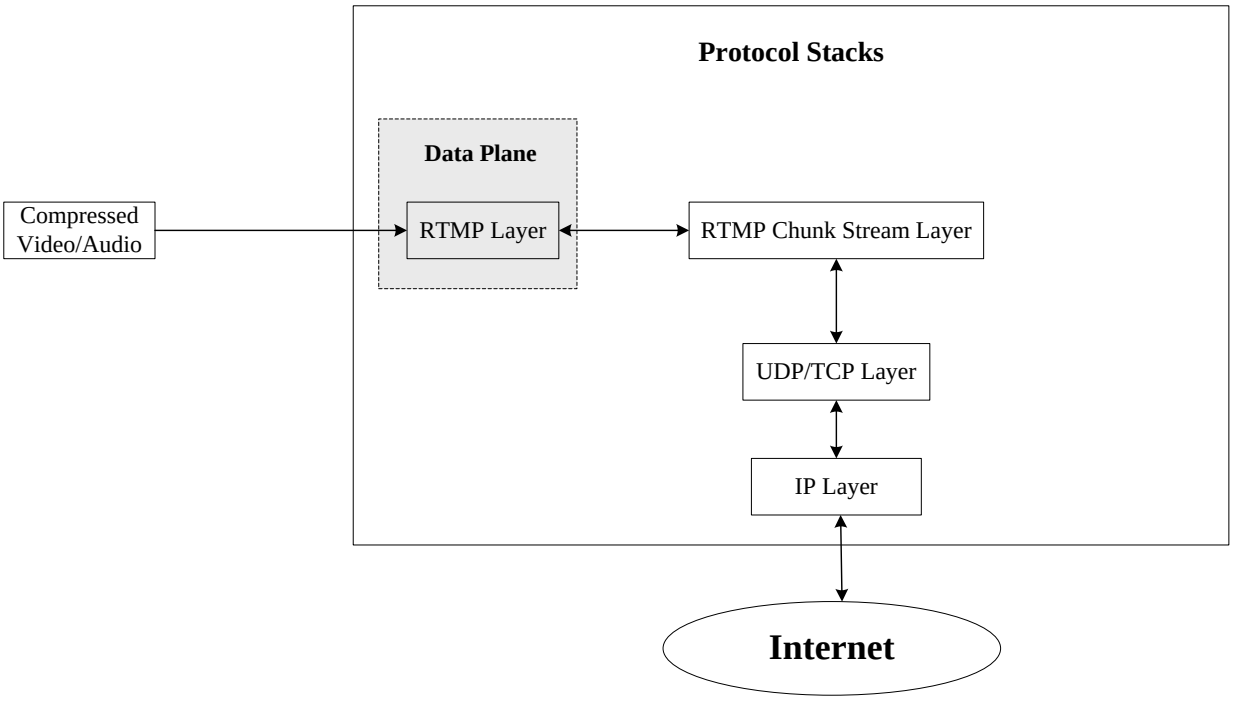

Figure 2- 4: Protocol Stacks for media streaming by Adobe Flash Server

2.2.1 TCP/UDP

Transmission Control Protocol (TCP): TCP is a process-to-process protocol using port numbers. TCP supports error control, congestion control and flow control. TCP can multiplex data streams from multiple applications running on the same machine with the same IP address. To detect bit errors TCP employ checksum. If a single or multiple bit-errors are detected in the incoming packet, TCP layer discards the packet so that the upper layer (e.g., RTMP) will not receive the corrupted packet. TCP uses retransmissions to recover lost packet, and provides reliable transmission. TCP employs congestion control to avoid sending too much traffic, which may cause network congestion. Finally TCP employs flow control to prevent the receiver buffer from overflowing [10].

User Datagram Protocol (UDP): UDP is a connectionless, unreliable transport protocol. UDP provides process to process communication instead of host-to-host communication and does not add anything to the services of IP. UDP can multiplex data streams from multiple applications running on the same machine with the same IP address. UDP does not have any flow control and congestion control mechanism. UDP does not guarantee packet delivery, so the receiver needs to rely on upper layer to detect packet loss [10].

2.2.2 RTMP and RTMP Chunk Stream

According to the RTMP specifications, Real Time Messaging Protocol and Real Time Messaging Chunk Stream Protocol are two independent protocols which RTMP works on top of RTMP Chunk Stream. While RTMP can use any other transport protocol and RTMP Chunk Stream can handle any protocol that sends a stream of messages, together they are suitable for a vast majority of audio/video applications. In this section we give a brief introduction on these two protocols.

Real Time Messaging Protocol: RTMP messages are sent between the client and the server over the network. These messages include video, audio, data or any other messages. RTMP message header specifies the type of the message, length, timestamp and message stream Id. These messages can be of any type including audio messages, video messages, command messages, shared object messages, data messages, and user control messages. A message type value is reserved for each of these messages. The client and the server use audio/video messages to send audio/video data to the peer. Command messages are sent to perform operations such as connect, createStream, play, publish, pause on the peer. Remote Procedure calls takes place over streams that are communicated using the command messages to the peer. A shared object is a flash object which synchronizes multiple clients, instances and so on. Data messages are metadata or any other user data that client and server send to each other. User control messages are sent between client and server to notify the peer about the user control events.

Types of command exchanged between client and server are categorized in two object classes. Net Connection is an object class which is a higher –level representation of connection among the client and the server. Connect, call, close and createStream are commands which can be sent over a NetConnection.

NetStream is an object that represents the channel over which video/audio streams and other data are sent. Play, deleteStream, publish and pause are commands which can be sent over a NetStream. The message flow for connect command and play are illustrated in the Figure 2-5 and Figure 2-6.

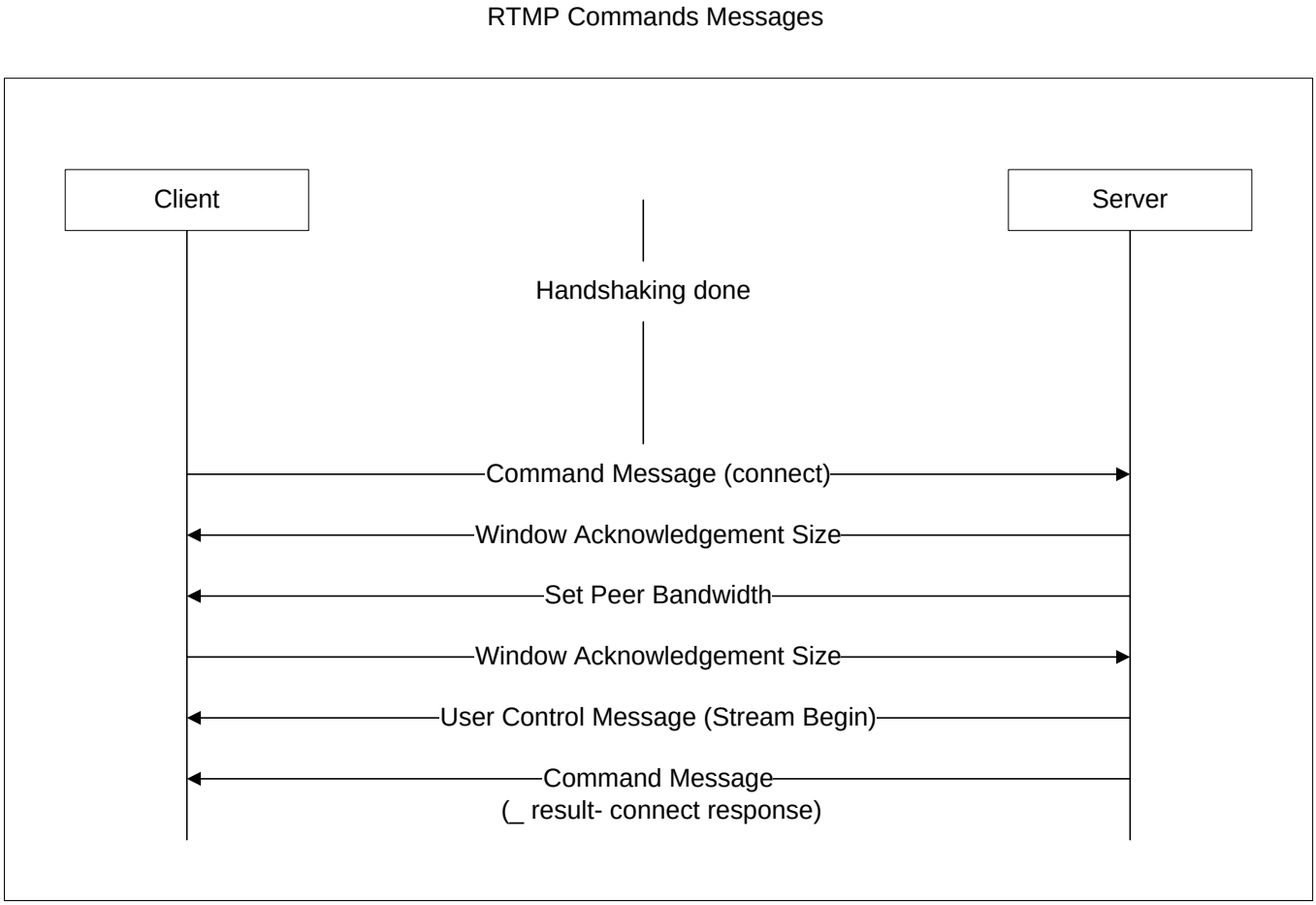

Figure 2- 5: Message flow during Connect command

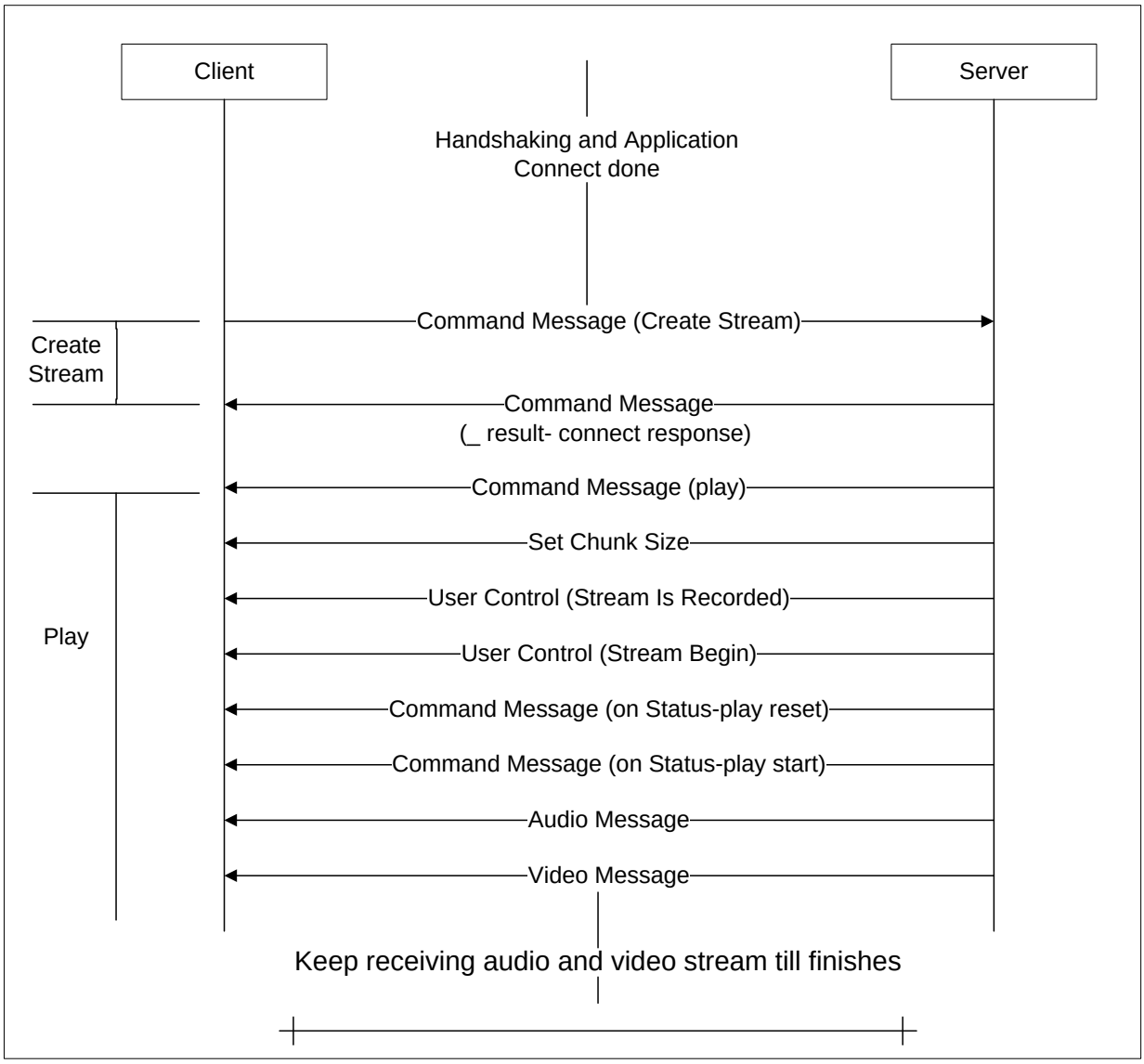

Figure 2- 6: Message flow during Play command

Real Time Messaging Chunk Stream Protocol: RTMP Chunk Stream offers packetizing and multiplexing services for a higher-level multimedia stream protocol. RTMP Chunk Stream is designed to work with the Real Time Messaging Protocol (RTMP], but it is able to handle any protocol that sends a stream of messages.

RTMP Chunk Stream offers guaranteed timestamp-ordered end-to-end delivery of all messages, when used with a reliable transport protocol such as TCP. RTMP Chunk Stream does not support any prioritization or similar forms of control, but a higher-level protocol which is used can provide such prioritization. RTMP Chunk Stream message header includes timestamp, length, type Id and message stream Id. The format of the message depends on the higher level protocol but it should contain the header fields.

An RTMP connection starts with a handshake. The handshake is different from the rest of the protocol. It contains three static-sized chunks. The client and the server each send the same three chunks. These chunks has been designated C0, C1, and C2 when sent by the client; S0, S1, and S2 when sent by the server . The Figure below shows the handshake between the client and the server.

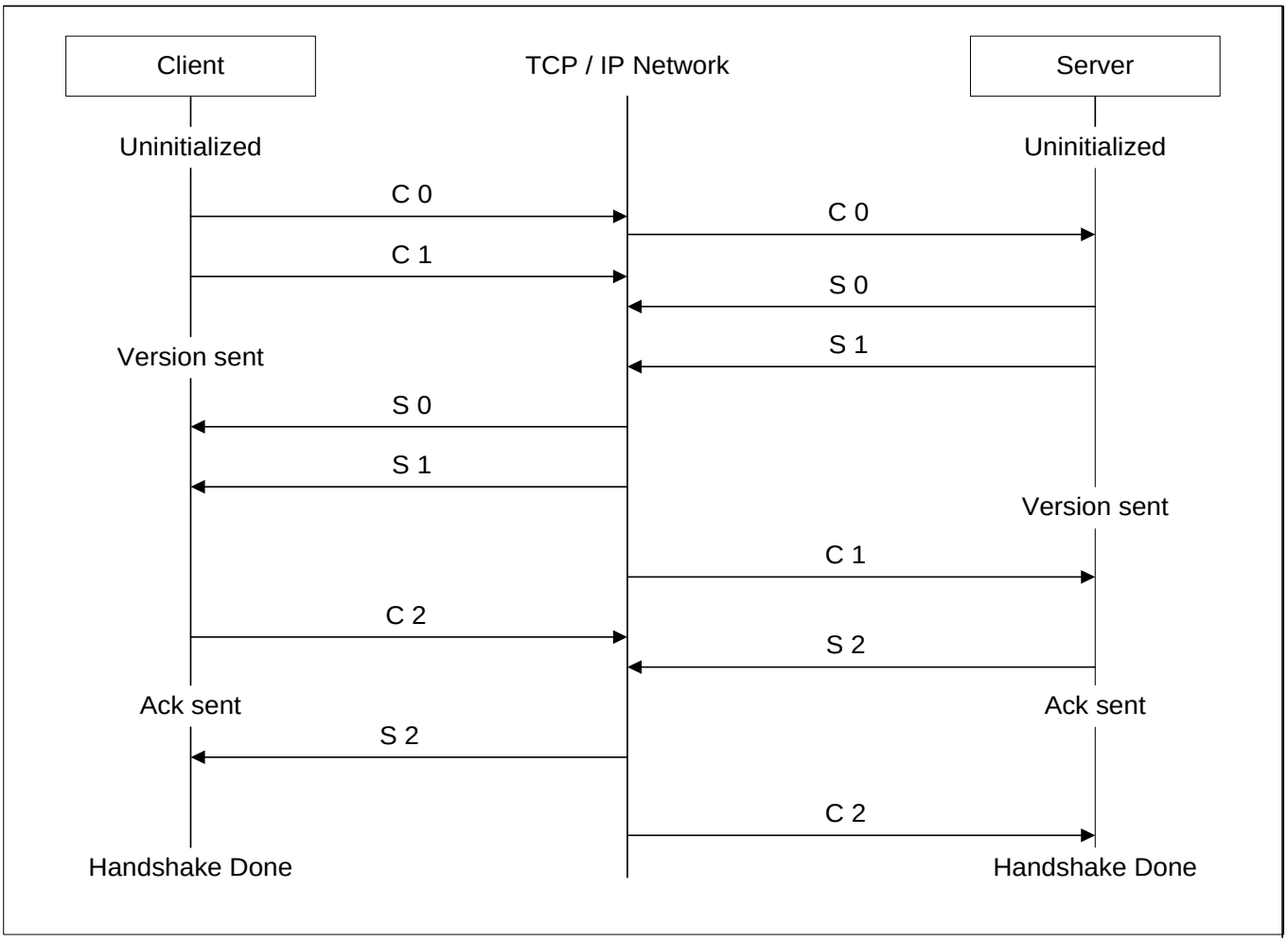

Figure 2-7: Representation of Handshake diagram

While handshake is done, the connection multiplexes one or more chunk streams. Every chunk stream contains messages of one type from one message stream. Each chunk has a unique ID associated with it called chunk stream ID. The chunks are transmitted over the network and while transmitting, every chunk must be sent in full before the next chunk. The chunks are assembled into messages based on the chunk stream ID, at the receiver end. Chunking enables large messages at the higher-level protocol to be divided into smaller messages and also enables small messages to be sent with less overhead [9].

2.3 Related Works

SeeToo is a web-base service which delivers viewing a same content at the same time. The SeeToo service enables viewers in multiple locations to watch pre-recorded videos together. The service allows the users in different locations to watch video together at the same time and in synchronization. To use this service the host user need to download and install a plug-in on the web browser [11].

Another product which provides a similar service is Zync by Yahoo Messenger. Zync is a plug-in which should be installed on yahoo messenger and works inside the instant message client. By using this service users are able to watch a youtube video simultaneously. This service enables users to watch the video at the same time on the instant message window and send text to each other while watching the video [12].

SWIS is a web-based service which works in a web browser and we don't require users to install a plug-in (except flash player which is widely available in most of the computer and devices). Voice and video chat are also available beside the synchronized player. These are the main features that distinguish SWIS from previously developed services.

Chapter 3

3. Analysis

This chapter explains the design process of the SWIS service. First the scenarios covered in the SWIS service are described and then the requirements for design are extracted from the scenarios. These are the scenarios which describe the functionalities that are expected from SWIS application:

Scenario 1

Bob is sitting on his couch on a Friday evening and planning to watch a movie. He called his friends if they can also come to his place and watch movie. Some of them agreed to come and some said they can't come but can be online from their place. He is also missing his friend Alice who is traveling outside the country. She is a good critic of movies and it's always fun to watch movie with her. Bob decided to watch this movie using "SWIS" service. He logins in to his page and selects his friends form his friend list and sends the invitation to them. He selects the movie from his laptop and uploads it to the server. After uploading the movie he waits for his friends to join online. He knows some of his friends are always late. He decides to give those couple of minutes and then he will start to play the movie. His TV is connected with internet so in his living room he will watch the movie directly on TV. Alice was on her way to hotel when she got the invitation message on her mobile phone. She immediately accepts the invitation and she logins to her page when she arrives at hotel. She finds a notification that her friend Bob is sharing his living room today and gets a link to join there. She clicks the link and goes to bob page. They start watching the movie. Carol was in sleep while she receives the message from bob and when she gets up she saw the message and decides to join them. She logins to her page and goes to bob page and joins them while Bob, Alice and others were already watching the movie. When she joins them they were talking to each other and she just unsubscribe to their voice chat and stop listening to their conversation while she is not in the mode for talking. Bob thanks the SWIS service for making it possible to have a wonderful evening.

Scenario 2

Alice is traveling by train. It's a long journey. She starts to record movie of outside scenarios and suddenly she captured a hilarious event. She decides to share this with her friends. She uploads the movie on "youtube". She also notices that the clip does

not have any sound in it. The event was so exciting that she cannot help talking with her friends about it. She logs in to her "SWIS" page and invites her friends to watch this video with her; she enters the link of the movie in her page. Bob is at home and receives an invitation message from Alice. He logins to his "SWIS" page; Alice saw bob entering the page and starts the clip. Steve is on the way home while he gets the message and he selects the SWIS application on his mobile phone and sees that Bob and Alice are in the middle of watching the movie. He presses the talk button and asks what is going on there, and Alice explains the event for him.

Scenario 3

Alice likes to watch TV series. There is a new series every Friday but she is busy every Friday and can not watch it. She found out that "megavideo" website has the episodes of this series. She decides to make a schedule to watch every episode with her friends Bob, Charlie and Trudy. She sends the invitation which says that every Saturday at 10 P.M she is going to watch one episode of the series and they can join her. Every Saturday at 10 P.M her friends get a message that Alice is watching this movie and you can join her. On Saturday at 10 PM, Bob was already waiting to watch the movie with Alice, he logins to his page and waits for Alice to share the movie. He also wants to see her so he presses the talk button and tells her to share her webcam. She agrees and starts to share her webcam. While Charlie gets the message she is on her way home and she is bored and tired. So she just wants to talk to them. She logins to her page and just subscribe to the voice chat and listen to her friends talking and mutes the movie.

Scenario 4

There is party at Alice's place and she really misses Bob and Susan that aren't there. She was about to make a video call to Bob and realized that Bob's mobile phone does not support video call. She knows that bob has internet connection. She decides to invite them on her webpage to the party. She broadcasts the party live and her friend can watch it thorough their webpage. Bob logins to his page and see what is going on at the party. Susan does the same thing and they both see and hear what is going on at the Alice's party. People are arguing on political matters. Bob wants to share his opinion. He presses the talk button and starts speaking and people at party listen and chat with him. Alice has high quality home theater system and Bob was feeling virtually there.

By looking at the scenarios, we conclude that:

- The system should initiate an event which is responsible to invite family and friends to watch a movie together.
- The host user should be able to choose a video clip/movie from either local drive or video sharing websites.
- While the host user initiates a session, player's view should be synchronized for all the users present on the session.
- Every user should be able to initiate voice, video and text chat at any moment.
- Every user should be able to listen to others, watch their videos and notes at any moment.

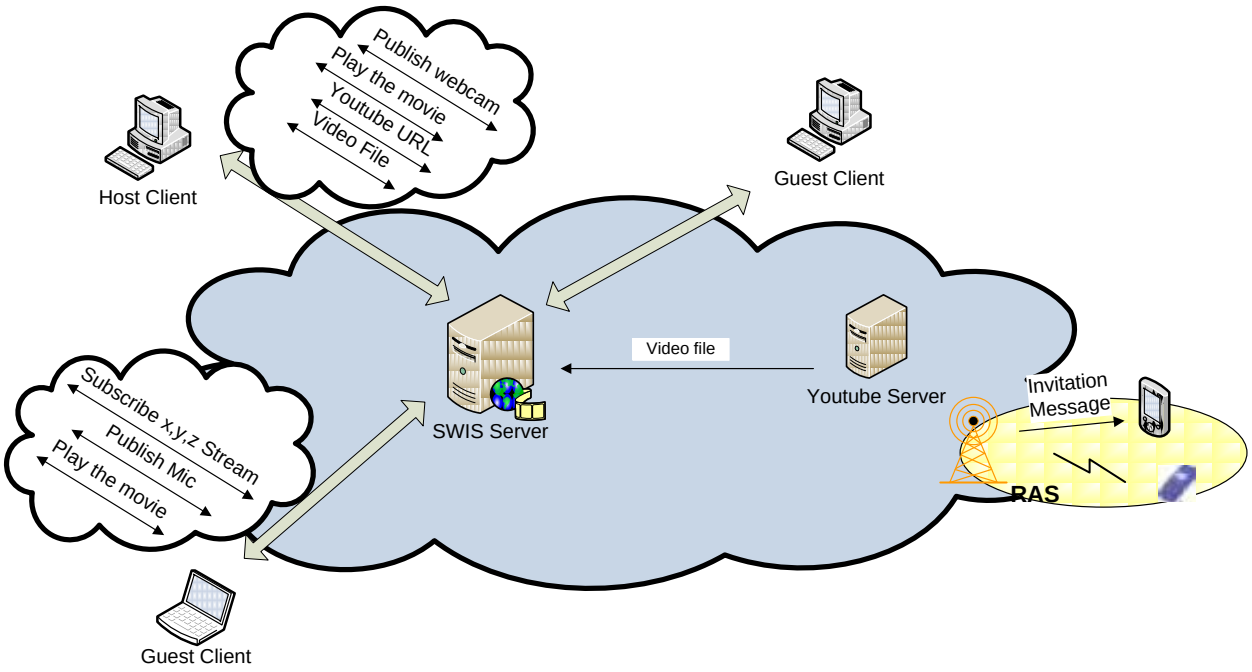

Figure 3- 1: System architecture of SWIS

Figure 3-1 shows the final system architecture. To explain the entire system, we break it into four different subsystems:

- **Event organizer**: The event of inviting friends and family to watch the movie with us is handled in this subsystem. I represent different approaches and methods to organize the invitation event.
- **Video clip/movie selector**: The options for choosing a video clip/movie to share are confined in this subsystem. We explore the potential ways that user prefer to use to share the video.
- **Synchronized player**: This subsystem handles building a synchronized player. I explore different approaches and methods to deliver a synchronized player.
- **Interactive mediums**: This subsystem handles different communication mediums such as voice, video and text. Each part is built separately and finally integrated.

3.1 Event organizer

In this subsystem we explore different approaches to handle the invitation event. The first step to establish a session of SWIS service is to inform people that they are invited to watch a movie. In ordinary cases people call, record voice message or send SMS to each other. These are considered as fastest ways to reach people and deliver a message. Two very useful methods to organize the invitation event are sending SMS and emails. As we mentioned in the scenarios user need to make an account and add friends like any other social networking application. When user makes an account, he/she can choice on how to get the invitation from other users. He / she can enter his phone number or email address or both to receive the invitation. The host choice the persons from the friends list andsends invitation instantly without caring about the methods of sending invitation. The system decides how to send invitation to the invited person depending on their profile settings. Invitation message contains a link, a password and the session start time.

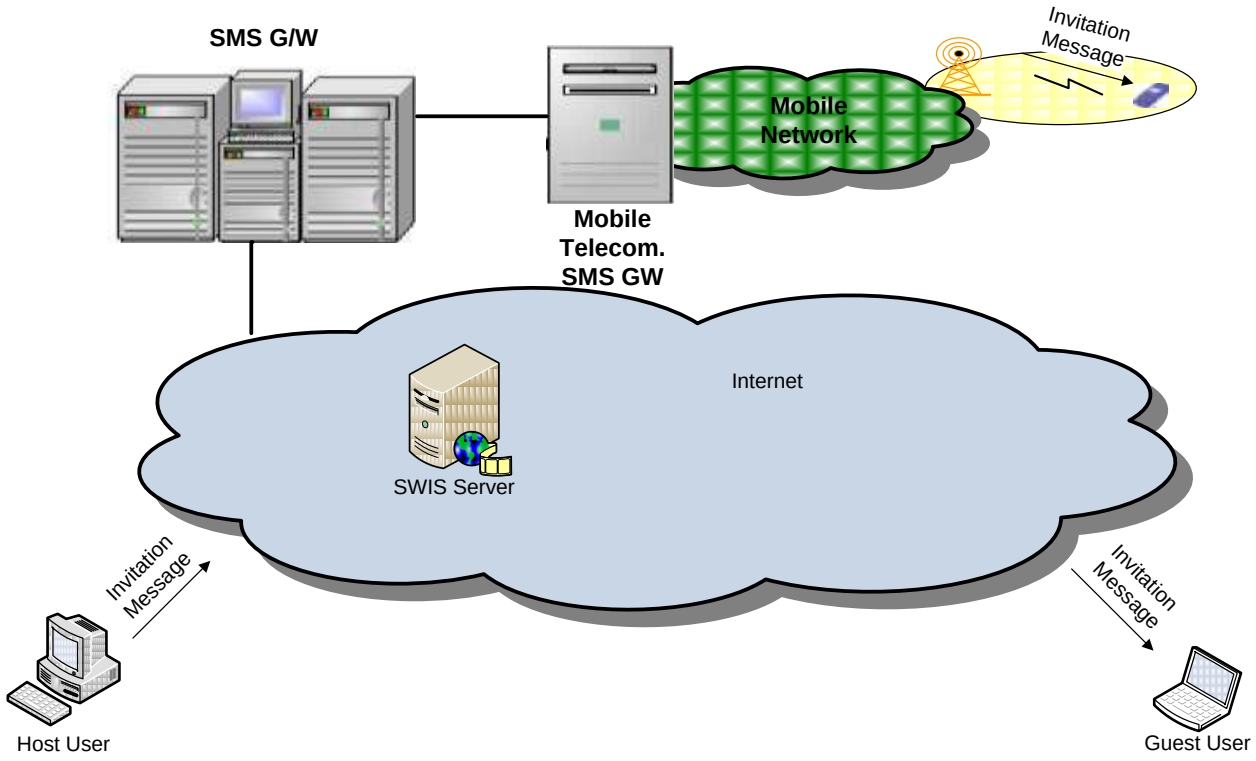

Figure 3-2: system architecture of the invitation event over the network.

3.2 Video clip/movie selector

The choice of selecting of a video clip can be divided into two categories, 1) selecting a video from local drives and 2) selecting a video from video sharing websites. If the video is located on local drives, there are two ways to use the file, one is to stream the file directly from the user machine and the other way is to transfer the file to the server. Security issues prevent to stream the file directly from user machine; therefore the file needs to be transferred to the server. When the user decides to share a video from a video sharing website, then the server needs to work with the other server which hosts the video. User just passes the link to the server to process the link. Our server needs to have an access to the original video file on the remote server, but every website needs a different method to access the video file and download it to our server. In our demo we just allow the user to choose video from youtube website which is the most popular video sharing website. In the section 4.2 we give a brief description on different approaches compatible for different websites.

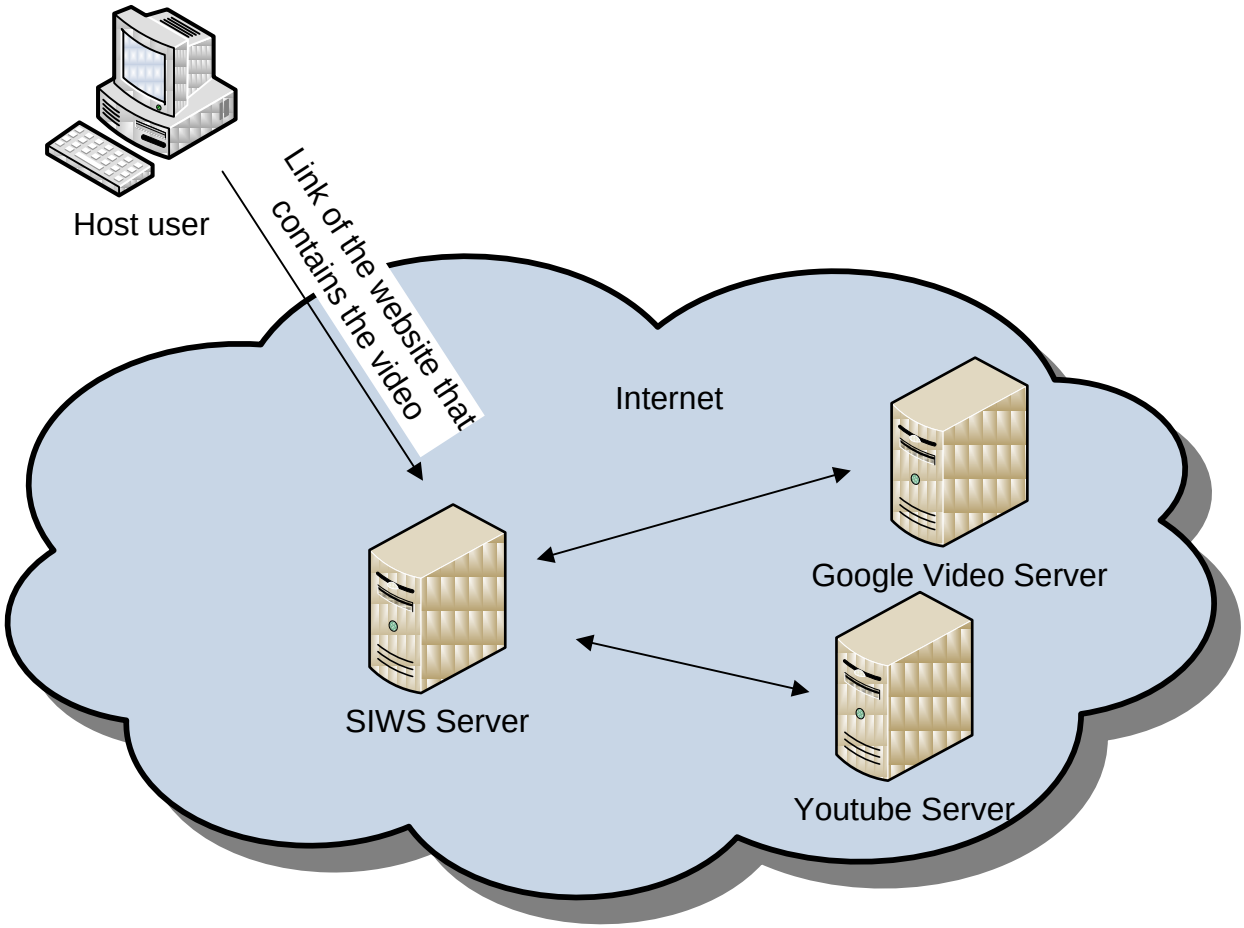

Figure 3-3: delivering the video content over the network

3.3 Synchronized player

One of the primary services in this application is the synchronized player. Synchronized player provides a way to view the same video frame simultaneously at every users current view. This is similar to traditional live television broadcasting except that transmission takes place over the Internet and the content providers are the users themselves. Nowadays live audio/video distribution is usually achieved through application-layer multicast or through multiple separate server-to-client unicast streams. Application-layer multicast works with multicast routing in the network layer. Multicast routing enables a single source node to send a copy of a packet to a subset of the network nodes. Multiple separate server-to-client unicast streams means a live stream is being published on the server and every client connect to the live stream separately. In our demo we are using separate server to client unicast streams i.e. each client receives a separate stream for each media it subscribes to, since Flash Media Server supports only unicasting.

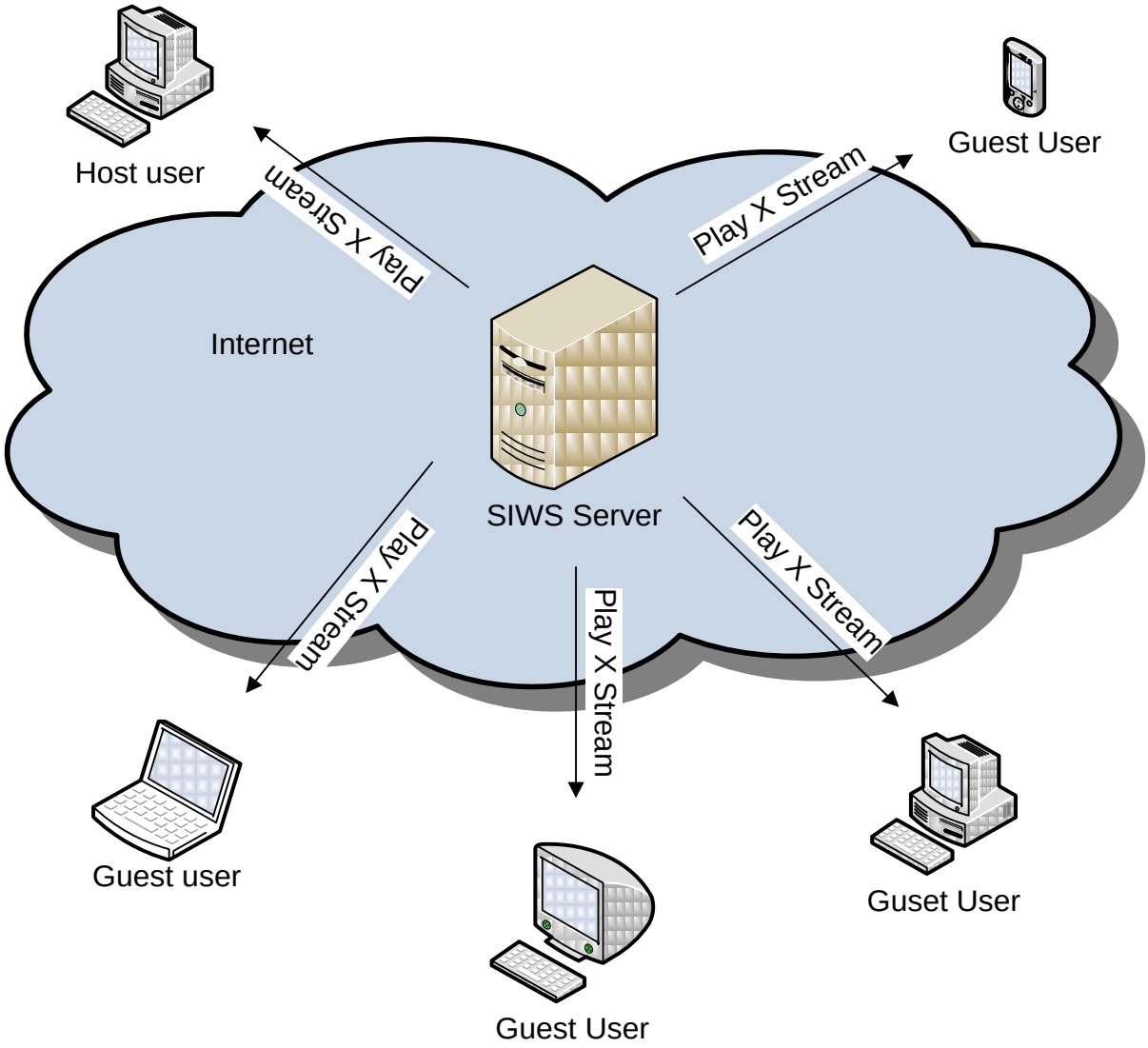

Figure 3-4: Building a synchronized player over the network

2.4 Interactive mediums

We described in the scenarios that how each medium of communication plays a significant role in SWIS service. Real-time interactive audio/video over the Internet is often referred to as Internet telephony/video conferencing. Each user publishes its own stream to the server while attaching his voice and video.

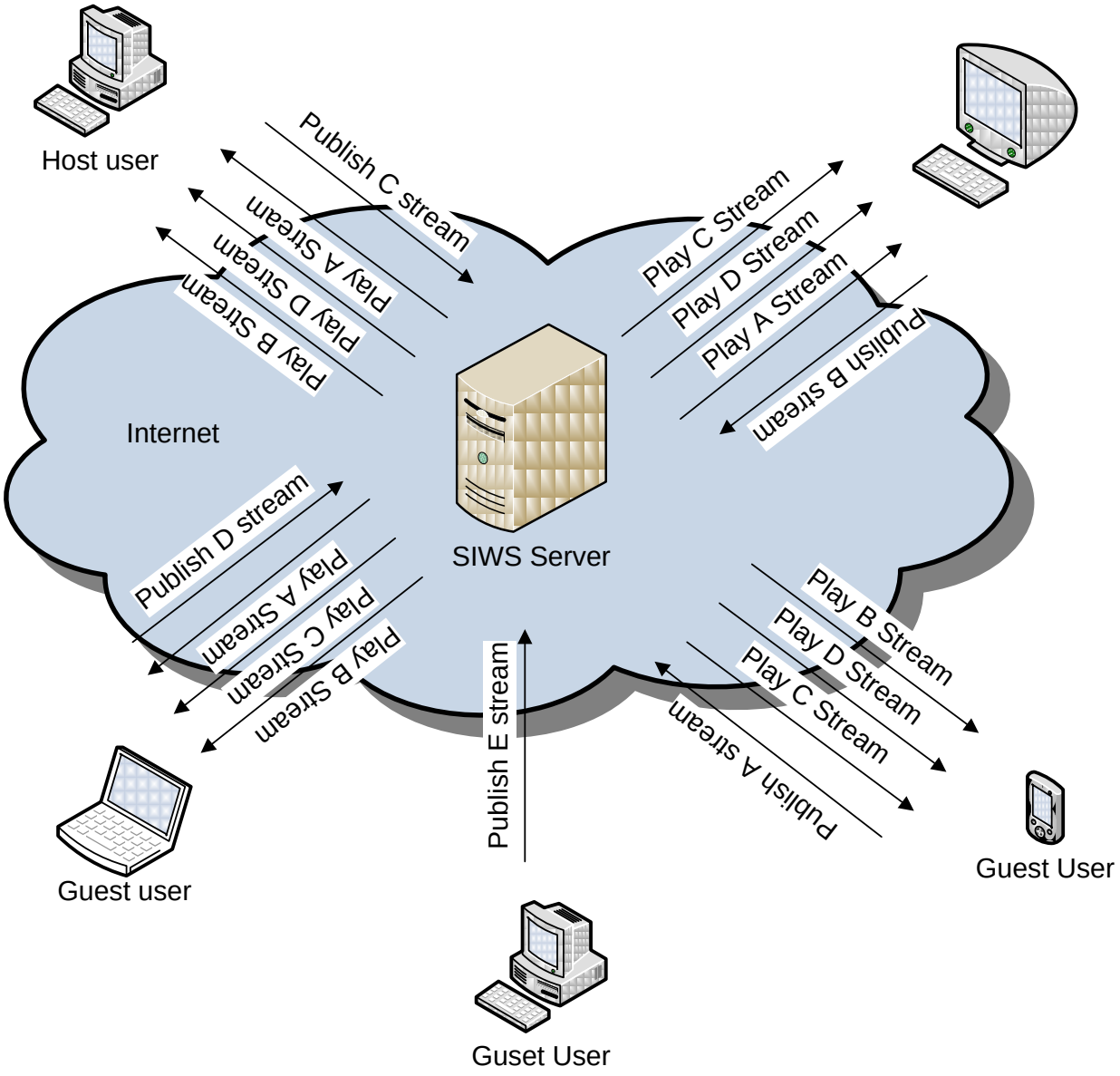

Figure 3-5: Video interaction and Internet telephony over the network

These four subsystems need to be explored in details. For each part we will suggest different solutions and chose one of them and implement the demo based on that solution. These parts will be integrated in the final demo and result in a service which covers every aspect of this service.

Chapter 4

4. Design

In previous section we explained the system design and divided the system into four subsystems. In this section we describe the implementation of each subsystem.

4.1 Event Organizer

This subsystem handles the invitation event. As mentioned in chapter 3, invitation message is sent via either SMS or via E-mail or both. The invitation message includes a url to the host page and a pin code for the session. It also indicates the time and date that the session is planed to start. Therefore, when the time arrives the guest user click on the url and enters the pin code and joins the session. In this section we describe the components and technologies involved in providing this service.

4.1.1 Short Message Service (SMS)

To provide the service with the SMS service the end systems attached to the internet should be able to communicate with telephones attached to a circuit-switched telephone network. Session Initialization Protocol (SIP) [??] is one of the protocols that offer this service. The international standard for SIP has been prepared for the SMS service by both 3GPP (3rd Generation Partnership Project) [13] and ETSI (European Telecommunications Standards Institute) [14]. We describe sending the SMS message of SIP which uses IMS (IP Multimedia Subsystem) [15] of 3GPP. Figure 4-1 illustrates the message flow of transmitting the SMS message from the SIP terminal of IMS to the mobile terminal, in IMS environment.

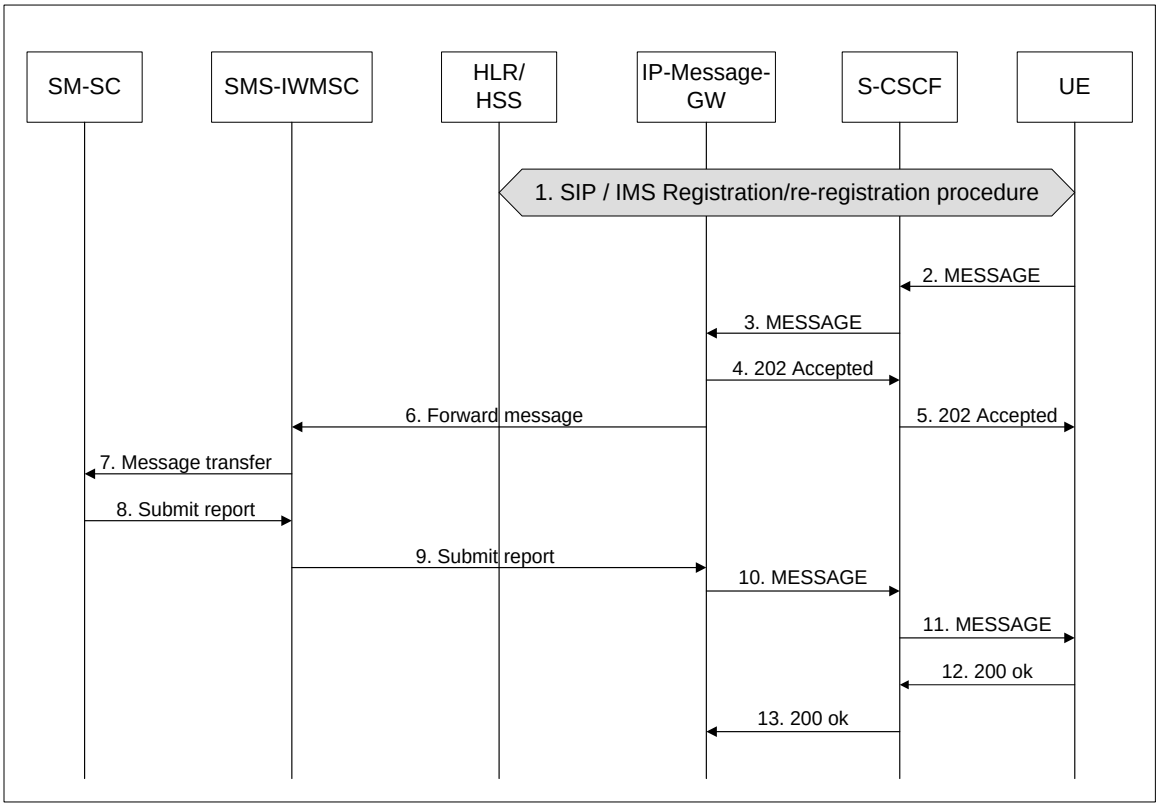

Figure 4-1: the message flow when the SIP terminal is originated and the mobile terminal is received [16]

The SIP terminal ends the registration process in '1. Step'. UE(User Equipment), the SIP terminal, sends '2. MESSAGE' with the SMS contents to IP-Message-GW(IP-Message- GateWay) through S-CSCF(Serving - Call Session Control Function). IP-Message-GW answers '4. 202 Accepted' to UE via S-CSCF. It means IP-Message-GW just received 'MESSAGE'. IPMessage-GW sends '10.MESSAGE' to UE through SCSCF. It means IP-Message-GW has finished transmitting final destination terminal. In the IMS environment, figure 4- 2 illustrates the message flow of transmitting the SMS message from the mobile terminal to the SIP terminal of IMS [16], [17].

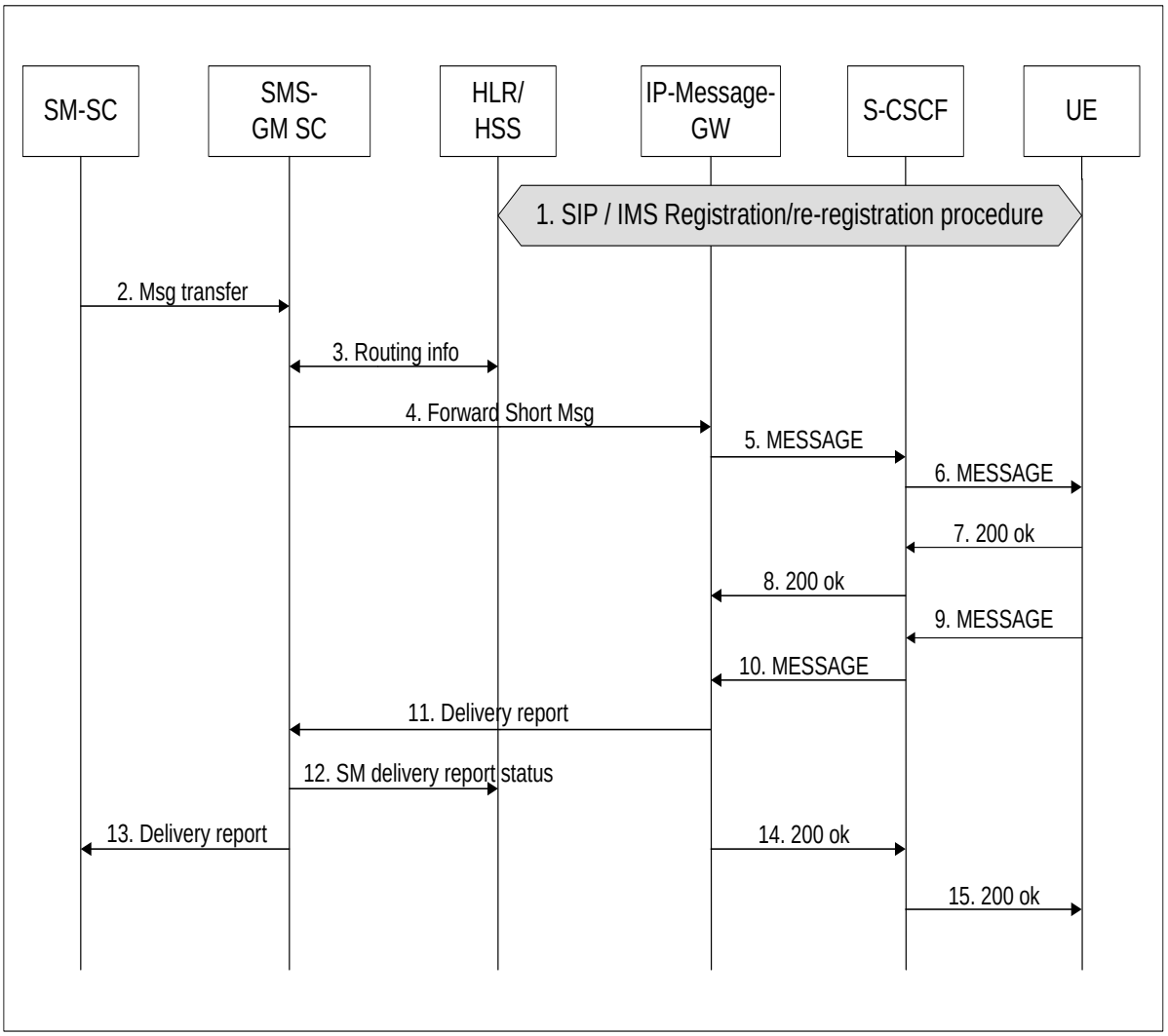

Figure 4-2: The message flow when the mobile terminal is originated and the SIP terminal is received [16]

The SIP terminal ends the registration process in '1. Step', too. UE which is the SIP terminal receives '6. MESSAGE' with the SMS contents from IP-Message-GW through S-CSCF. UE answers '7. 200 OK' to IP-Message-GW through S-CSCF. UE receives the message well. UE sends '9.MESSAGE' to IP-Message-GW through SCSCF. After IP-Message-GW finished processing messages, send '14. 2000K' to UE through S-CSCF. Finally when UE received that message, all message processing ends.

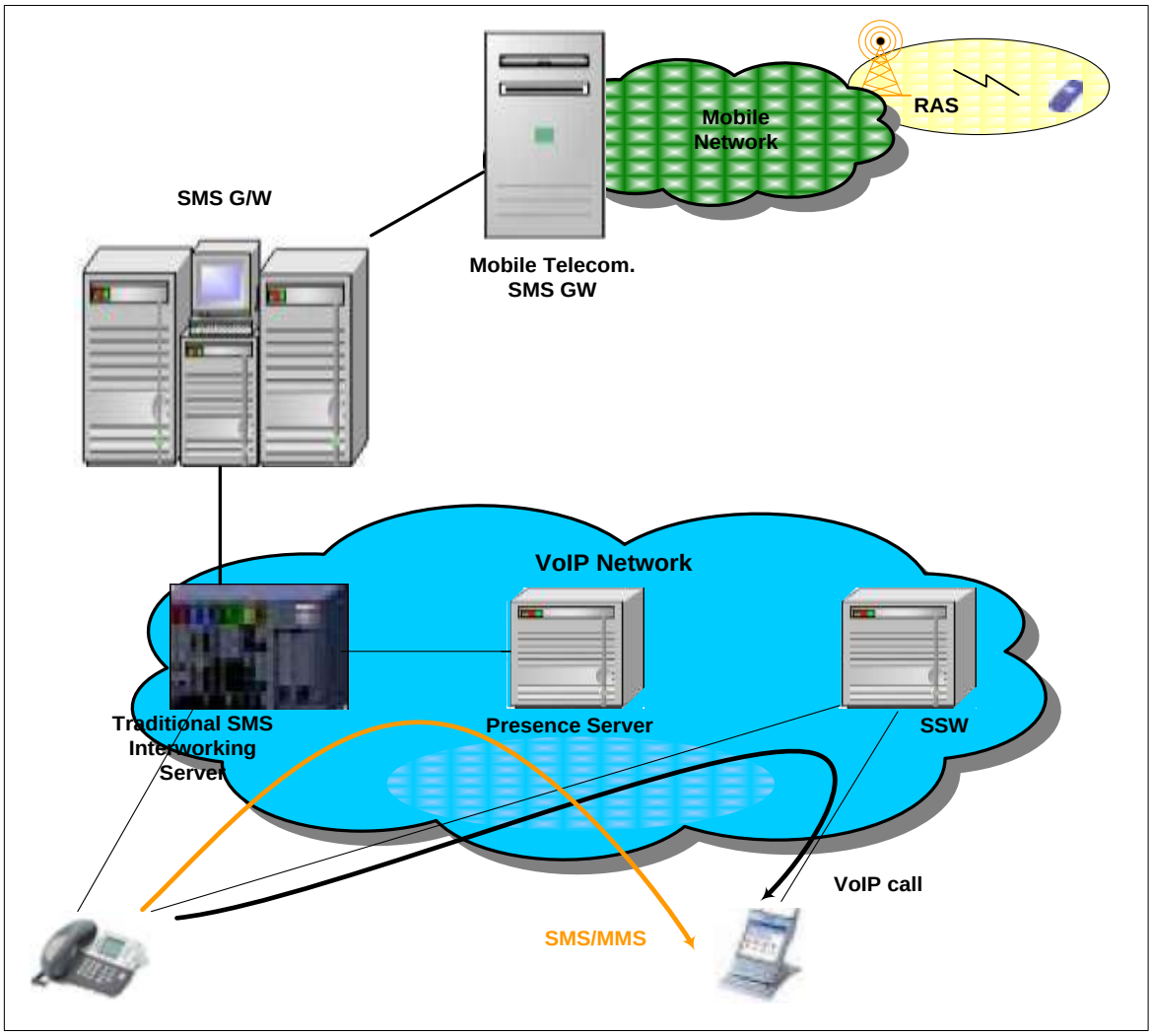

Figure 4-3: The example of SMS/MMS network using 3GPP standard [16]

4.1.2 E-mail

Sending the invitation via E-mail could be done by using scripting language like PHP and ActionScript. A PHP file on the server gets parameters from flash and sends the message to already defined e-mail addresses which are added to the database. ActionScript gets the content of the message and send it to the PHP file on the server. PHP file receives the content and the address of the receiver and sends the E-mail automatically.

4.2 Video clip/movie Selector

User can select video clip/movie from local drives or video sharing websites. This section describes these two methods:

4.2.1 Selecting the video from local drive

This is the scenario when the user owns the video. In this case either the video should be uploaded to the server or it can directly be streamed from the user computer. In the first case, user selects the video from his/her computer and uploads it to the server. The problem with upload is that user has to wait until the file is transferred to the server. We consider eliminating the time for uploading by starting to play after an initial delay time. This approach is explained in the next section.

Another approach is to directly access the file on user machine. Server is able to work directly with the video file on the user machine, but I faced some problem from the client-side. The problem is the security issues of client side do not allow directly accessing the file on user computer. We tried java applet and action scripts as two different client-side languages. Java Applet requires a certificate and adobe flash restricts any access to file on user computer. These restrictions are applied to enhance the security while connecting to the internet. Therefore in our demo the video file needs to be uploaded to the server.

4.2.2 Selecting the video from video sharing websites

This is very likely that user decides to watch a video from one of the video sharing websites on the internet. In this section I examine how to access the video file which resides on a video sharing web site.

Internet includes a large variety of multimedia applications which streams stored audio/video. In these applications, client requests on-demand compressed audio/video files that reside on server. Thousands of sites provide streaming of stored audio and video today, including YouTube, CNN and Google video. As we explained in chapter 2 these servers can be web servers or special streaming servers. Protocol used to transfer the video between client and server varies based on the type of the server. We briefly explain the basic approaches to catch the video file from video sharing websites based on type of the servers.

Web servers: Numerous video sharing websites are using ordinary servers and progressive download method for delivering their videos such as youtube. In our application we redirect videos from youtube. I download the video file directly to our server from youtube server. I find the url for downloading the video that resides on youtube server by capturing the packets which

has been received from youtube server. There is a packet that carries a request for downloading the video. The message for requesting the video is extracted from that packet. This message is the same for all the videos and there is just the video ID that differs. Therefore I define an application on the server and add a java class which gets youtube url as input and generates the download url. Our server sends a request to download the video from youtube server automatically. These steps are illustrated in the sequence diagram in the figure below.

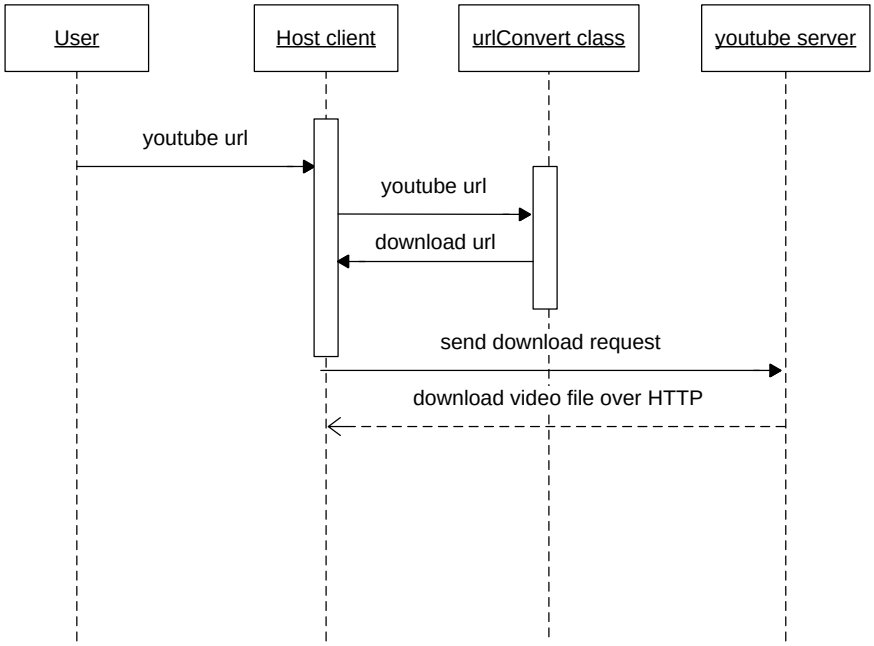

Figure 4-4: The sequence diagram of downloading video file from youtube server

This method works specifically for youtube website and is implemented in our demo.

Streaming servers: A streaming server could be a proprietary streaming server, such as those marketed by Adobe Flash and Microsoft, or could be a public domain streaming server. With a streaming server, audio/video can be sent over HTTP/TCP, or it could be sent over UDP. For different protocols that are being used in streaming servers, we should consider different solutions. We describe a solution for adobe flash server which uses RTMP protocol. Red5 java library enable us to make a connection to the remote flash server and stream the video to our server and allow us to save it as a flv file. Later this flv file is used as media source file. The steps of this process are shown in the sequence diagram below. In SWIS service I used the streaming server approach which is more flexible and useful for transmitting multimedia over the network.

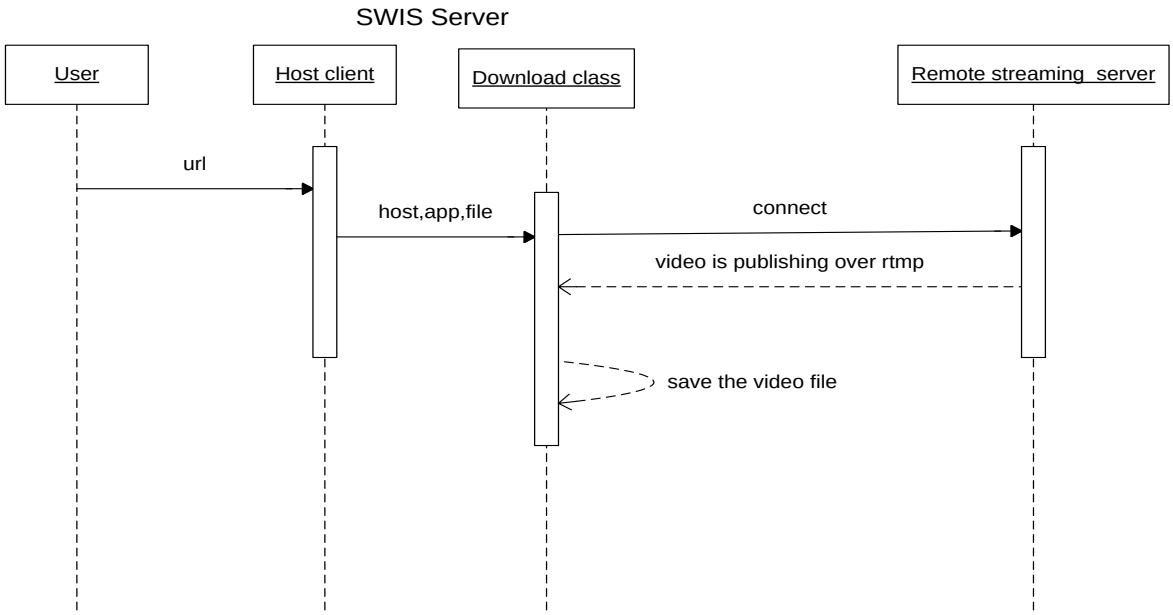

Figure 4-5: The sequence diagram of downloading a video from a flash streaming server

This section gave an introduction on the subject of accessing to the original video file on the video sharing websites. There is software available on the internet called Replay Media Catcher. Replay Media Catcher is an application that captures the video and save the HTTP and RTMP streams to a flv file as they are streamed to the end user.

4.3 Synchronized Player

One of the goals in SWIS service is to build a synchronized player. Every user present on the same session should view the same video frame at the same time. Live streaming and shared timer are two different approaches to build a synchronized player. Section 4.3.1 and 4.3.2 describe these methods in details.

4.3.1 Live Streaming

Live streaming is based on publish and subscribe technology. The video is published live at the server and users connect to the server and subscribe to the published stream. Red5 library has RTMPClient class which acts as a client that connects to other flash servers and transfer data over RTMP protocol. We build an application on the server and add a java client class that publishes the video file to the server. Flv file is passed as input to the java client class and it makes rtmp packets and publishes them on the server. Figure 4-6 shows the activity diagram of this class.

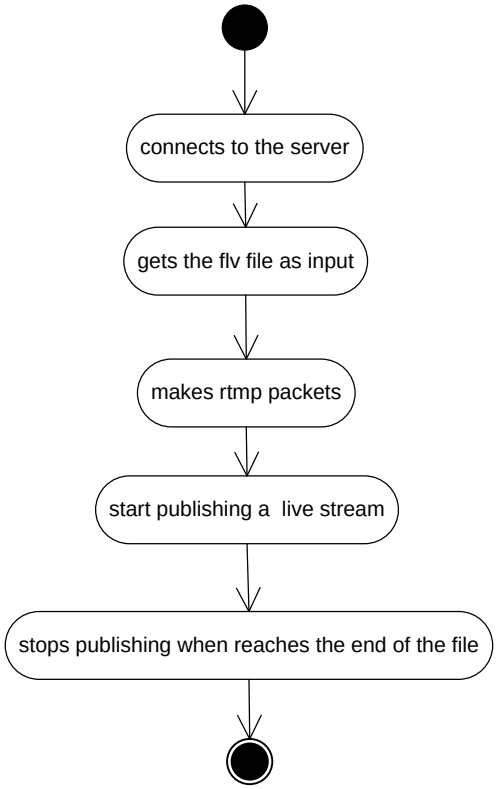

Figure 4-6: The activity diagram of publisher class on the server

The java client which is responsible to publish the video file can be run on the server in two different ways. It can be written as a java applet or a java application. A java applet runs by fetching the web browser. The java application resides on the server and runs with RPC by Actionscrip. In our demo the java application is used.

Java Applet rtmpClient: The rtmpClient applet is placed on the server. When the user presses the play button, the browser runs the rtmpClient Applet on the end user machine. Java applet requires the flv file as input to start publishing. But security issues prevent the applet from getting the file from user machine. This problem could be solved by requesting for a certificate. And when the user wants to play the movie he should accept the certificate and continue, otherwise if the user denies the certificate the processes fails.

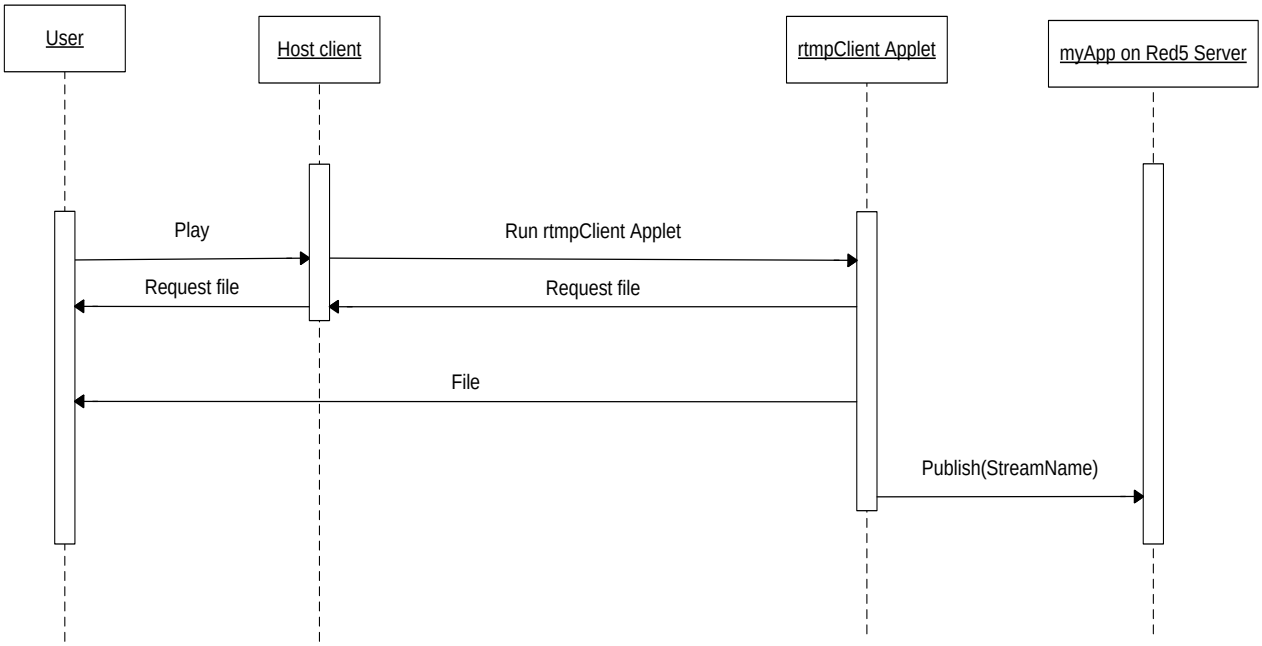

Figure 4-7: The sequence diagram of publishing file with java applet

Java Application rtmpClient: The second approach is when the file is uploaded to the server. In this case there should be a component which performs the publishing job. An application on the Red5 server can perform this task. The main class of the application is rtmpClient class. When the user presses play button after uploading the video file, actionscript connects to the Red5 server and calls a method from the rtmpClient class on the server. This method runs the rtmpClient class and start publishing the selected video live on the server.

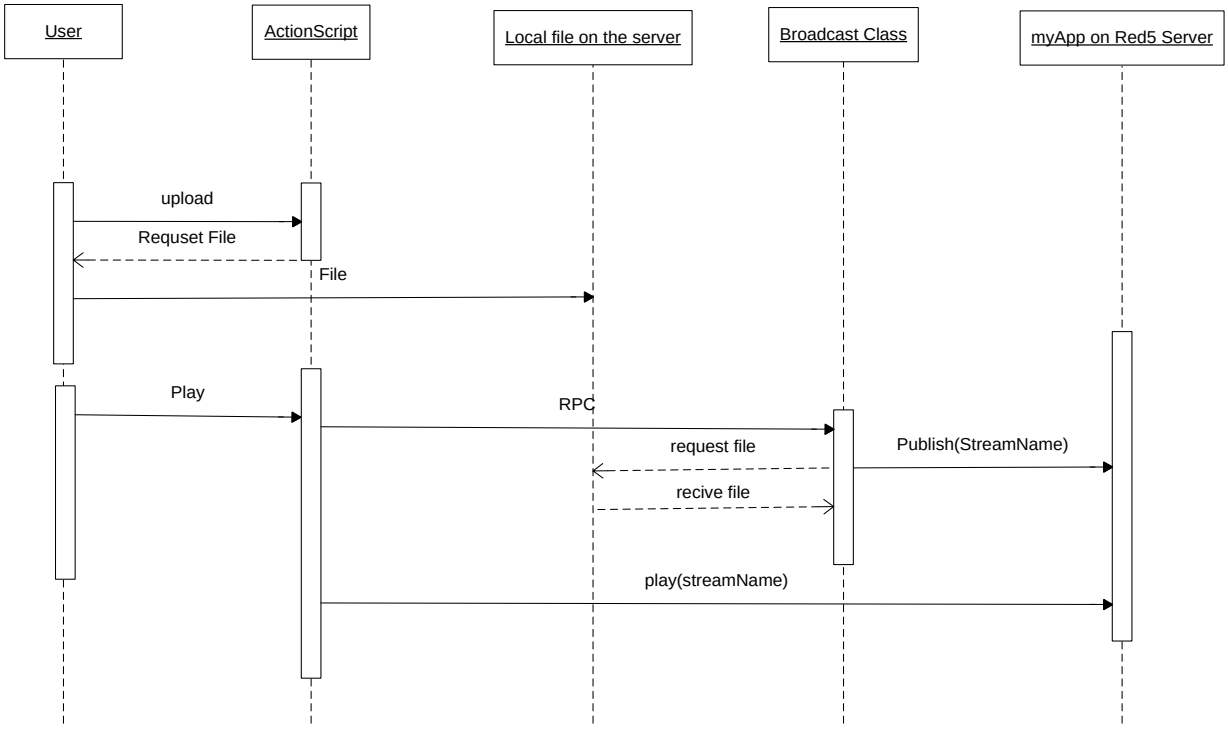

Figure 4-8: The sequence diagram of publishing with application on Red5 server

As it is shown in the figure 4-8 after the file is uploaded and placed on the server, the user is able to start playing. When the playing starts, actionscript on the client side sends RPC to the broadcast class on the server. The broadcast class gets the file which is located on the server and start publishing the file under the name streamName, and the actionscript on the user page plays the streamName stream.

When the chosen video is from another websites, user enters the url of the page that contains the video. Server's job is to work on this url and produce the download URL get the video file. In our application we can just get video from youtube server. This process is shown in the sequence diagram below.

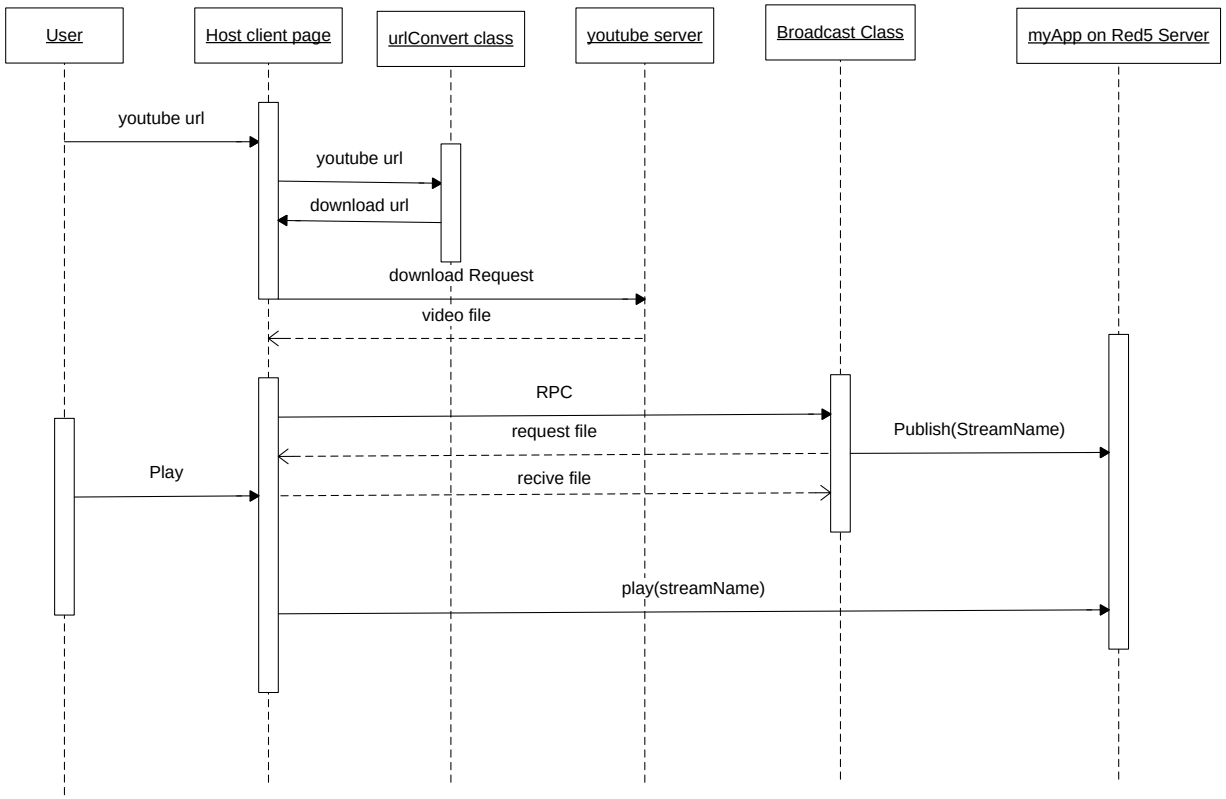

Figure 4-9: The sequence diagram of downloading a video from youtube and publish it to the server

Either the file is sent to the server by user or by youtube website, in both cases SWIS Server does not wait until the file is completely transferred. After initial part of video is placed on the server, publishing can start. The host user starts playing the movie and the server gets the first version of the video file which is not complete and starts to publish it. When it reaches to the end of the first version of the file, it goes in a loop which continues until the file is completely placed on the server. In this loop, the publisher will automatically get the new version of the file and start to publish it. But this time it checks the meta file (a description file which contains data and information about the video file) to see the last key frame position that has been published and start the publishing from that position. This checking continues until the size of the file on the server equals the size of the file on the meta file.

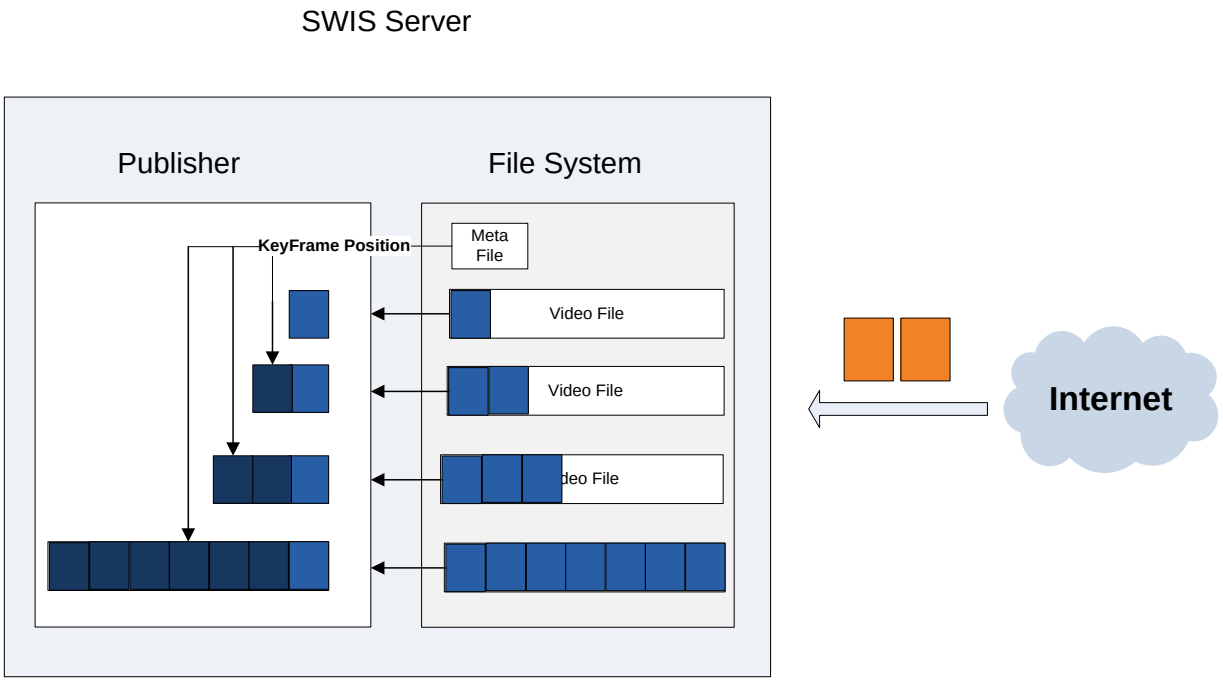

Figure 4-10: The process of playing the video while it is being downloaded

As it is shown in the figure 4-10 the file is being transferred to the server and I use this time to play the available part. It means as soon as server gets the video file it starts to play and while it is playing it continues to receive the file. This help to reduce the time for waiting until the file is completely transferred. The activity diagram below shows how the publisher acts in this approach.

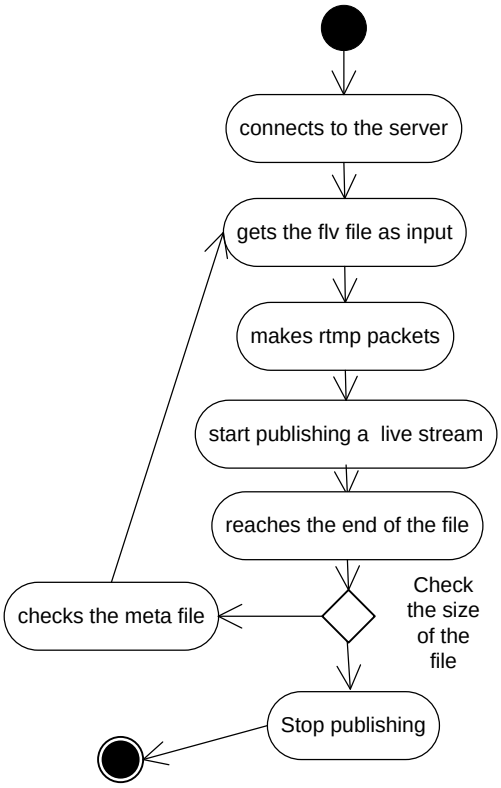

Figure 4-11: the activity diagram of getting the file recursively

4.3.2 Shared Timer

Another approach to build a synchronized player is to use a shared timer that sends the current time on the host player to the server. The host continuously sends the current time in the media player to the server and while guest clients start playing, player catches the current time from the server and seek to it, and they continuously update themselves. Therefore the result is the same video frame as the host user.

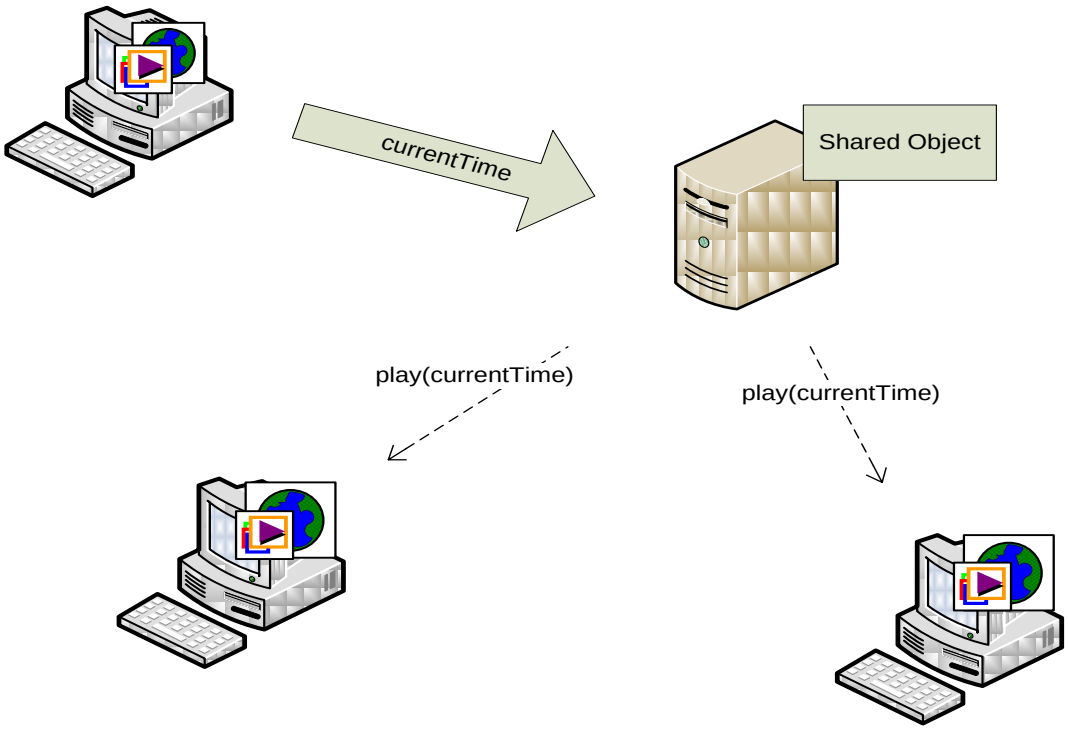

Figure 4-12: Time Sharing Architecture

The host client follows the steps shown in the Host Client sequence diagram: 1) Host plays the movie and start watching. 2) Player sends the current time to the server recursively. 3) Current time is written on a sharedObject on the server.

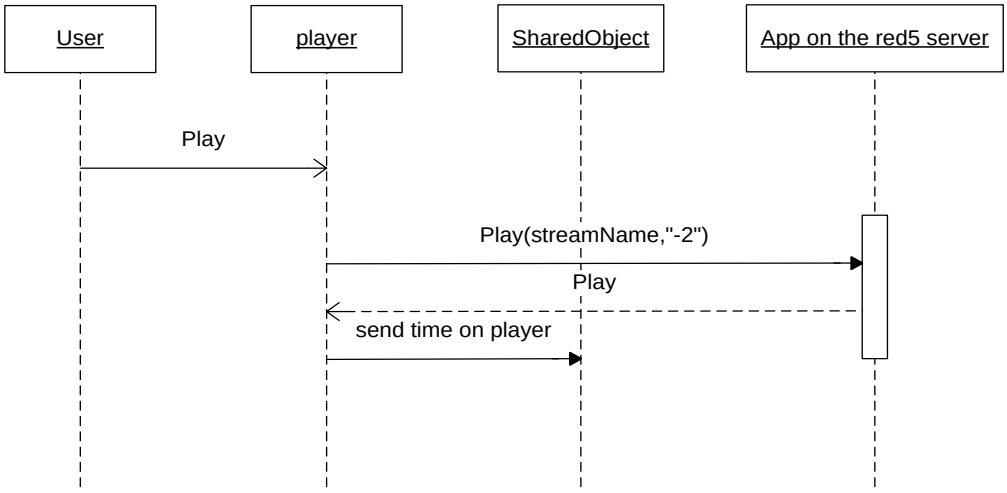

Figure 4-13: Host Client Sequence Diagram

The guest client follows the steps shown in the Guest Client sequence diagram: 1) Guest client joins the session, and then the player asks for the current time from the sharedObject on the server. 2) The player receives the current time from the server. 3) Player sends a play message to the server and set the start time as current time.

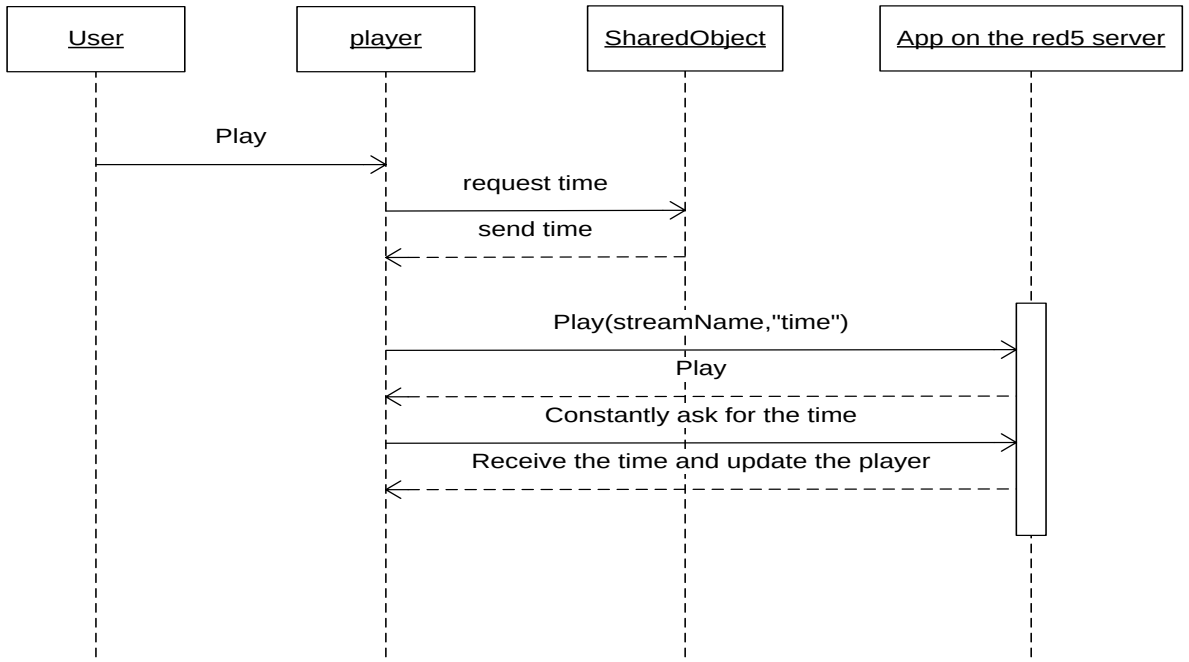

Figure 4-14: Guest Client Sequence Diagram

4.4 Interactive Mediums

Adobe flash offers the function of broadcasting webcam and microphone to the server. Video from webcam and audio from microphone attach to one stream and the stream is published to the server. Others connect to the server and subscribe to that stream. In SWIS service each user should be able to talk and send his/her webcam to others. When a session is started, every user present on the session needs a different name for the publishing stream. This is the server job to give a different name to each user. A new application is placed on the server to support voice and video chat. A java class that implements IStreamAwareScopeHandler interface acts as a server component that replies to client. This class has a method which specifies a new stream name for each user and sends the name of streams present on the server to subscribers. These steps are shown in the sequence diagram below.

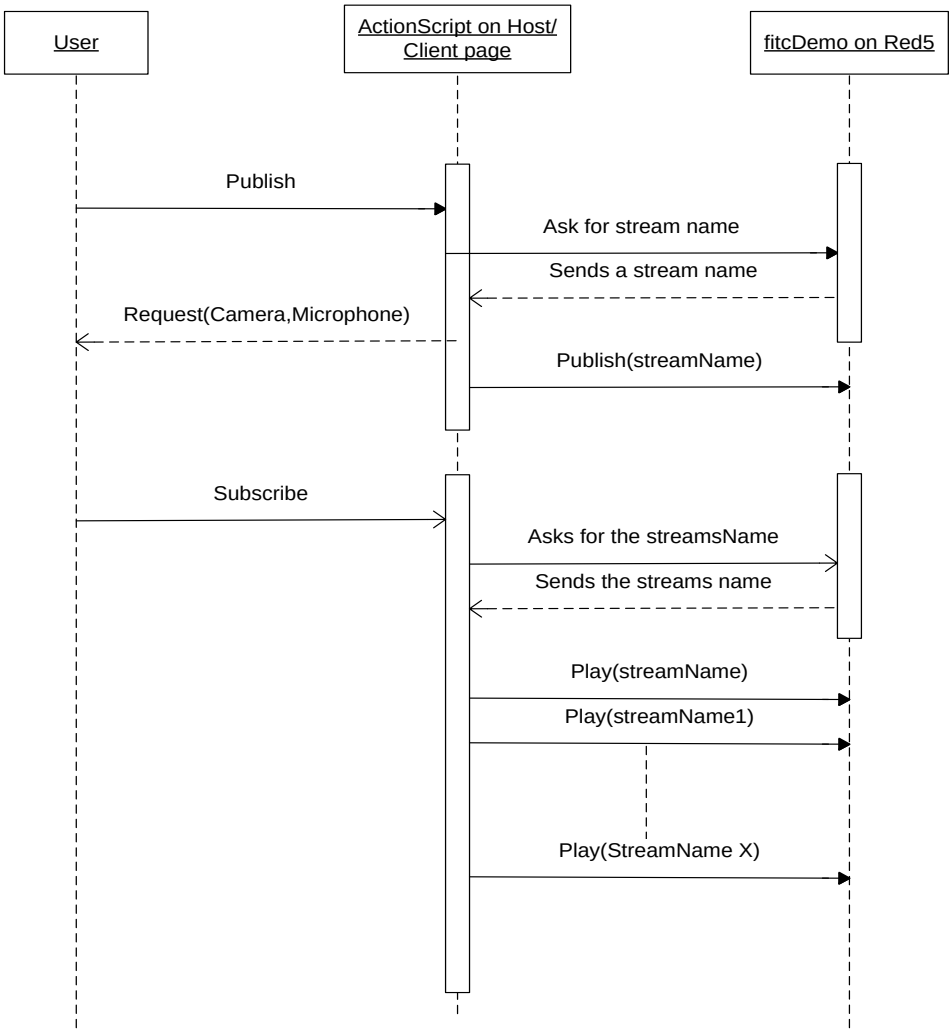

Figure 4-15: The sequence diagram of voice and video publishing to the server

4.5 Media Adaptation

Media Adaptation is the process of efficiently delivering streaming video to users by selecting among different streams of varying quality and size. This provides users with the best possible viewing experience considering their bandwidth. Dynamic streaming copes up with different network situations. The idea behind dynamic streaming is that during the playback server switches between the video file by considering the network bandwidth fluctuations [18]. Server sends the stream by detecting the initial bandwidth of the client. But the problem is when the actual available bandwidth of the user is moving, this can leave users continually having to buffer and wait for their video if the selected stream bandwidth is not sustainable on their network. Or some users may start the stream with low available bandwidth, and then free up more bandwidth after the video is started. In this scenario, dynamic streaming can offer the ability to up-scale the video quality to a higher level, and improves the user's experience.

In SWIS service is possible to use dynamic streaming, but since all the clients are connected to a single stream it is not possible to switch between the streams considering just one client bandwidth fluctuations. Therefore each client has to perform the dynamic streaming. The server can estimate the bandwidth between client and itself, and sends the stream which suits the client better. To achieve this goal, server should have the video with different qualities. When the host user starts playing the shared video, for each copy of the video, server publishes a different stream and for each client, server sends the stream that suits the client better. This is the initial part of the streaming. While the streaming is started, server constantly checks on the clients' bandwidth, and notifies the user if there is a change in the bandwidth. While the client gets the notification, it will switch among the streams. In the next paragraph I explain the process in more detail.

Assume there is a video file with both high and low quality available on the server. When the host starts publishing, server will start publishing both video files and produce two streams. At first the server checks the client bandwidth, and passes the suitable stream for the client. Assume the there is enough bandwidth to send the high stream to the client. Client is watching the movie with high quality while it gets notified by the server that there has been a change in the available bandwidth and the bandwidth has been reduced. Therefore the user switches between the streams and play the stream with low quality.

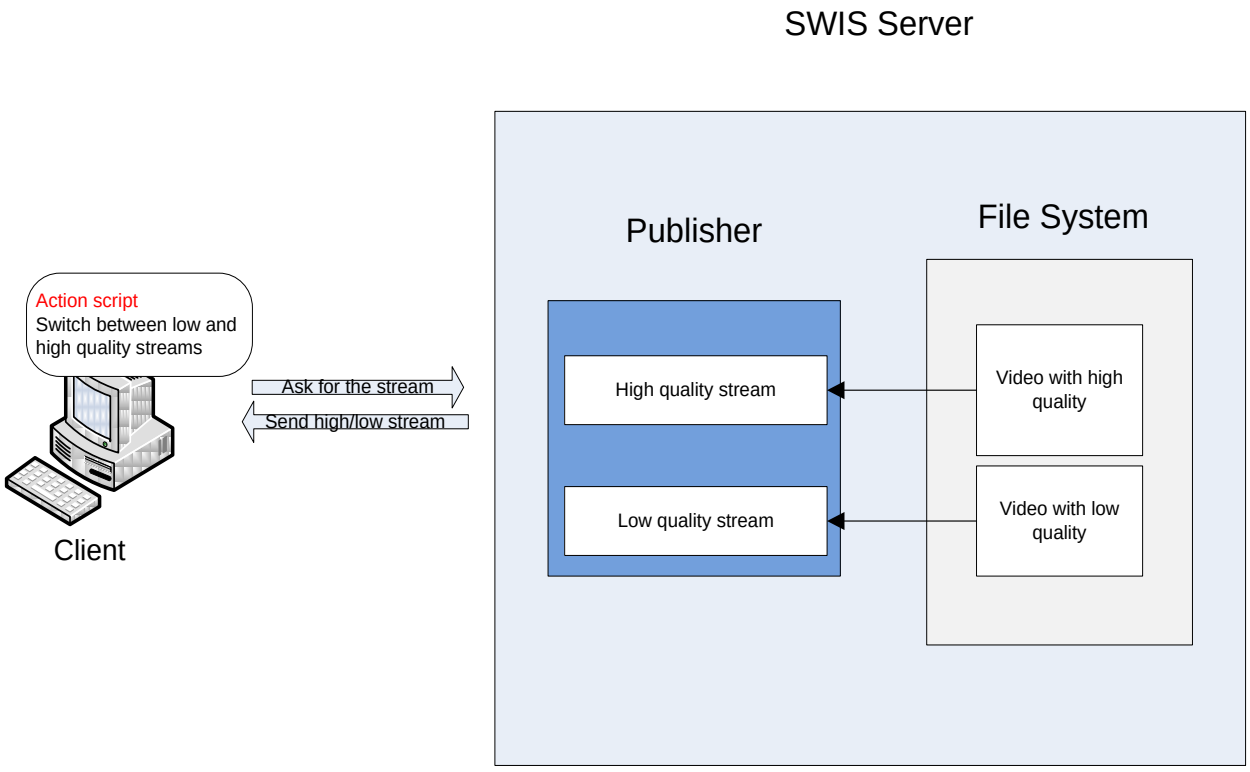

Figure 4-16: dynamic streaming on the server

4.6 legal issues

This service is considered legal since watching the video is done through a private channel and only people with invitation have access to this session. The videos are not saved anywhere on the client or server after the session is completed. This service needs approval from other video sharing website to access their video files.

Chapter 5

5. Conclusion and Future Works

This chapter describes the conclusions made during this work and explains the future works that can be performed related to this study.

5.1 Conclusions

By exploring the recent technologies and implementing the basic demo we have seen that this is possible to build this service on the internet. We have implemented a demo which simulates a virtual living room in cyberspace for a limited number of users. Therefore this service is feasible and there are enough technologies available to develop this service. The observed flash media quality was good and we found some solutions to enhance the quality by considering the client network QoS and users can enjoy this service while watching a movie with the expected quality.

While attempting to have a synchronized view for every user present on the same session, we have faced lack of support from network layers. Server does not receive any feedback from the clients, and it makes the synchronization process more difficult and less reliable. With the help of network we can assure that the synchronization performance is right.

This service can be integrated by a social networking application. While users already have their own personal pages and have their friends in their friends list, this service can be offered to them as one the services available on the application. This makes the process of using the service faster and easier. And users appreciate this kind of service since these social networking applications are mostly not real-time with passive interactions.

To enable users select movies from video sharing websites, this service need agreement with video service providers to avoid legal issues. This would be an easy solution if before developing the service, we specify the video sharing websites that is covered in the service and make an agreement with them to let us use their database.

5.2 Future works

The following future work can be done related to this thesis:

- In this thesis we didn't focus on the first subsystem and it wasn't implemented on the demo. Exploring the methods and approaches for sending the invitations, finding the best solutions and implementing the final solutions are suggested as a future works in this part.
- Our demo just accepts videos with flv format. Considering time limitations, a solution should be found that convert videos in a real time.

- This service can be implemented to deliver different QoS to users with different internet connections. This option enhances the quality of the video and makes users aware of their connection condition and alerts them even if they are not able to use this service.
- Covering more video sharing websites to provide the content.

References:

[1] Hypertext Transfer Protocol, http://www.ietf.org/rfc/rfc2616.txt, 9/12/2009.

[2] Progressive Download, http://en.wikipedia.org/wiki/Progressive_download, 9/12/2009.

[3] Streaming Server vs Web Server, http://www.microsoft.com/windows/windowsmedia/compare/WebServVStreamServ.aspx, 12/10/2009.

[4] James F.Kurose, Keith W.Ross, "Multimedia Networking Applications", in Computer Networking A Top-Down Approach, 5nd ed, Addison–Wesley,2007.

[5] Adobe Flash Media Server, http://www.adobe.com/devnet/flashmediaserver/articles/wm_flash_transition_guide/wm_flash_transition_guide.pdf, 10/10/2009.

[6] Adobe Flash Media Server, http://en.wikipedia.org/wiki/Adobe_Flash_Media_Server, 25/09/2009.

[7] User Datagram Protocol, http://www.faqs.org/rfcs/rfc768.html, 22/10/2009.

[8] Transmission Control Protocol, http://www.faqs.org/rfcs/rfc793.html, 22/10/2009.

[9] RTMP specification, http://www.adobe.com/devnet/rtmp/pdf/rtmp_specification_1.0.pdf, 10/11/2009.

[10] Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu,Ya-Qin Zhang, and Jon M. Peha "Streaming Video over the Internet: Approaches and Directions", vol 11, 2001.

[11] SeeToo, http://www.seetoo.com/home.php, 10/12/2009.

[12] Liu, Y. Shamma, D.A. Shafton, P. Yang, J. , "Zync: the design of synchronized video sharing", 2007.

[13] 3GPP TR 23.804 V7.0.0, "Support of SMS and MMS over generic 3GPP IP access"

[14] ETSI DES/AT-030036, "Fixed SMS over IP between TE and Service Centre"

[15] 3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2"

[16] Kwihoon Kim, Jinsul Kim, Hyun-Woo Lee and Won Ryu, "Method for transmitting SMS for VoIP service supporting Multi- protocol", 2007.

[17] Kyu Chul Lee, Jong Hyup Lee, "SMS Transmission Mechanisms for Multi-Protocols on VoIP", vol 3, 2007.

[18] Dynamic Streaming, http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_advanced_pt1.html, 20/11/2009.

[19] James F.Kurose, Keith W.Ross, "Streaming Stored Audio and video", in Computer Networking A Top-Down Approach, 5nd ed, Addison–Wesley,2007.

[20] Mxml, http://en.wikipedia.org/wiki/MXML, 10/10/2009.

[21] Action Script, http://en.wikipedia.org/wiki/ActionScript, 10/10/2009.

[22] Adobe Flash Player, http://en.wikipedia.org/wiki/Adobe_Flash_Player, 17/11/2009.

[23] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Transporting real-time video over the Internet: Challenges and approaches," *Proc. IEEE,* vol. 88, pp.1855–1875, Dec. 2000.

[24] Q. Zhang, Y.-Q. Zhang, andW. Zhu, "Resource allocation for audio and video streaming over the Internet," in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'2000), Geneva, Switzerland, May 28–31, 2000.

[25] Internet Protocol, http://www.faqs.org/rfcs/rfc791.html,06/01/2010