ÇANKAYA UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

MASTER THESIS

A CLIENT-SERVER ARCHITECTURE FOR LIVE VIDEO STREAMING USING OBJECT RELATIONAL DATABASE

SERKAN ÖZDEMİR

JUNE 2013

Title of the Thesis : **A Client-Server Architecture for Live Video Streaming Using Object Relational Database**

Submitted by : **Serkan ÖZDEMİR**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University

Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assist .Prof. Dr. Murat SARAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist .Prof. Dr. Murat SARAN
Supervisor

Examination Date: 18 June 2013

Examining Committee Members:

Assist .Prof. Dr. Murat SARAN	(Çankaya Univ.)	----------
Assist. Prof. Dr. Reza HASSANPOUR	(Çankaya Univ.)	----------
Assist. Prof. Dr. Erol ÖZÇELİK	(Atılım Univ.)	----------

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Serkan ÖZDEMİR

Signature :

Date : 18 June 2013

ABSTRACT

A CLIENT-SERVER ARCHITECTURE FOR LIVE VIDEO STREAMING USING OBJECT RELATIONAL DATABASE

ÖZDEMİR, Serkan

M.S.c., Department of Computer Engineering
Supervisor: Assist .Prof. Dr. Murat SARAN

June 2013, 99 Pages

This thesis focuses on live video streaming and offers a new approach based on client-server architecture using relational database. The thesis also analyzes the traditional live video streaming concepts and challenges such as performance problems. On the other hand, this study aims to implement client-server architecture in order to gain performance and provides a faster retrieval and storing time, better download time with minimum metadata by using relational database. This architecture also provides multiple accesses on different domains like embedded devices, Internet based smart TVs etc. The study also covers a Windows desktop application which consists of two live video streaming approaches. Implementation tries to compare traditional video streaming using TCP sockets and client-server model using relational database. MySQL and Apache web server were used to support the thesis proposal. Implementation was tested with various amounts of clients and parameters such as frame rate, buffer size and picture quality. Test results and conditions were also included in the thesis text. Briefly, this thesis tries to provide a better client-server live video streaming solution using the abilities of web and database servers.

Keywords: Streaming, Live Video, Broadcast, Relational Database, Client-Server

ÖZ

NESNE TABANLI VERİTABANI KULLANILARAK CANLI VIDEO AKIŞI İÇİN BİR İSTEMCİ-SUNUCU MİMARİSİ

ÖZDEMİR, Serkan

Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez Yöneticisi: Assist. Prof. Dr. Murat SARAN

Haziran 2013, 99 Sayfa

Bu tez canlı video akışı üstüne yoğunlaşarak ve ilişkisel veritabanı aracılığı ile bir istemci-sunucu mimarisi önermekte ve aynı zamanda klasik canlı video akış yaklaşımlarını inceleyerek, bu yaklaşımların performans, kalite gibi sorunlarını incelemektedir. Tezin diğer bir amacı ise ilişkisel veritabanını kullanarak verinin daha hızlı depolanması ve çekilmesi, daha iyi veri indirme süresi sunması ve gereksiz veri başlıklarını engellemesi noktasında bir istemci-sunucu mimarisi oluşturmaktır. Bu mimari sayesinde gömülü sistemler ve İnternet tabanlı akıllı TV uygulamaları gibi farklı konumlardan veriye erişim mümkün olmaktadır. Bu çalışma, iki farklı canlı video akış yaklaşımını test eden bir Windows masaüstü uygulaması da içermektedir. Uygulama, TCP soketleri kullanılarak gerçekleştirilen klasik canlı video akışı ile ilişkisel veritabanı kullanarak yapılan istemci-sunucu tabanlı canlı video akışını karşılaştırmaktadır. İlişkisel veritabanı kullanılan yaklaşımda MySQL veritabanı yönetim sistemi ve Apache web sunucusu kullanılmıştır. Tez uygulaması farklı sayıdaki kullanıcılarla ve çerçeve oranı, depolama boyutu, resim kalitesi gibi çeşitli parametrelerle test edilmiştir. Test sonuçları, test ortam verileri ile birlikte sunulmuştur. Özetle bu tez, web ve veritabanı sunucusunu kullanarak daha iyi bir istemci-sunucu canlı video akış çalışması üretmeyi amaçlamaktadır.

Anahtar Kelimeler: Video, Canlı Akış, İlişkisel Veritabanı, İstemci-Sunucu

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Live video streaming simply refers to delivery of multimedia that is received by an end user from a provider [1]. This transfer is achieved constantly in this manner. Live video streaming is the most important topic in telecommunication systems (radio, television, etc.) since the need of video-based broadcasts is increasing rapidly. The first “streaming” was used by IP Networks for video on demand (VoD) in 1990s [2] [3].

Live video streaming needs a source (such as a camera), an encoder to make data recognizable, a publisher and a distribution network to deliver the content. Traditional live video streaming is based on this architecture to send single resource to multiple clients.

However, there are some challenges to overcome such as performance, download time and metadata problems in order to access data from the side of client. Because traditional live video streaming is mainly accomplished by point-to-point connections and requires buffering, compression and decompression of data in a traditional streaming approach.

User Datagram Protocol (UDP) is very popular to deliver video and audio in real time because of its lower latency and faster delivery. In spite of this trend, HTTP based live video streaming is being considered by many researchers and content providers. Transmission Control Protocol (TCP) based solutions reuses the network infrastructures. So, amount of outbound traffic is reduced via widely deployed caches [4]. This also prevents scalability problems of servers. Real-time Transport Protocol (RTP) and UDP based solutions have problems with firewalls and NATs when the data traverse. HTTP streaming is easier than UDP in this manner, since the HTTP based solutions can easily use the abilities of typical web servers to deal with various media files.

Video on demand offers an approach called progressive download, which does not download the whole video at once, but downloads it in small parts and plays it immediately.

Peer to Peer (P2P) streaming is a new paradigm to deliver content to a large number of clients at the same time with low cost. There are many applications that offer this solution. But the key point is to analyze its performance with various parameters to provide a good solution from the client's point of view [5]. But, generally, P2P solution is better if the number of clients is extremely large. Another increasing demand for P2P is hybrid P2P which focuses on Internet Protocol Television (IPTV) channels. Hybrid peer to peer approach deals with the cost of peer to peer solutions while the content is hosted.

This study aims to implement a client-server architecture to gain performance and provide a faster retrieval and storing time, better download time with minimum metadata by using relational database. This architecture also provides a multiple access on different domains like embedded devices, Internet based smart TVs etc. Relational database for streaming which allows querying in a semantic way will provide a solution to data access problems.

Implementation uses relational database in server side to store and retrieve video in a fragmented format to share stream with multiple client at the same time without any loss. As a result, we expect a more reliable, secure, and faster live video streaming via relational database.

CHAPTER 1

VIDEO STREAMING

1.1 Background

Delivery of user generated and live videos are very popular topics nowadays with the increase of bandwidth capacities and interest in video sharing communities. Digital contents such as mp3 music and videos became popular in the early 1990s and the need for sharing these kinds of content increased rapidly and video player devices replaced with digital media players by the increase of hardware capacities.

Live video streaming began to play a significant role with the increase in bandwidth capacities in Internet. Now, live video streaming consists of various elements like video source, receiver and a transmitter. Currently live video streaming is used in wide range of categories. Most popular concepts are e-learning, IPTV, radio, video sharing sites like Ustream [6] and areas such as security, and medical. The most common video streaming architecture is shown in Figure 1 below. Basically, it consists of a video source, destination, media publisher and transmission line.

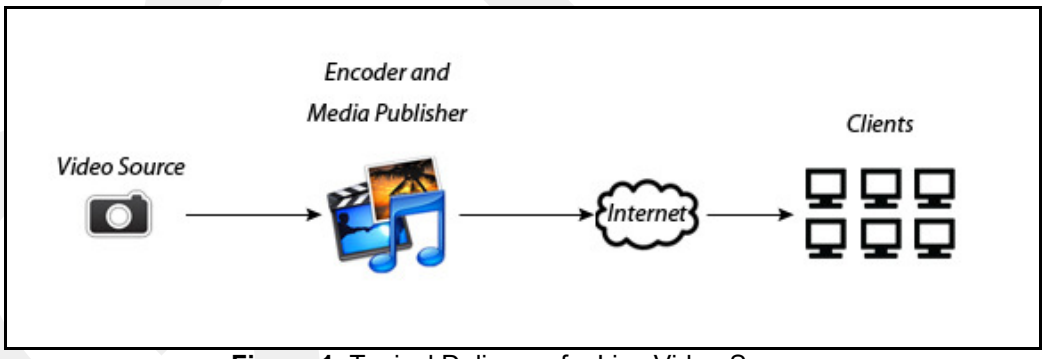

Figure 1. Typical Delivery of a Live Video Source

Live video streaming was supported with many standards, and researchers focused on the best video experience for the users. The most common protocols are listed below [7]. Each of the protocols below has both advantages and disadvantages depending on the usage, amount of clients and transmission line.

1. UDP, User Datagram Protocol
2. RTP, Real-time Transport Protocol
3. RTSP, Real-time Streaming Protocol

4. RTCP, Real-time Transport Control Protocol
5. SCTP, Stream Control Transmission Protocol
6. TCP, Transmission Control Protocol
7. P2P, Peer to Peer

P2P based live video streaming became very popular since the ability of P2P is very strong. If the number of receiver clients is too large, P2P based solutions should be considered. We will focus on these protocols deeply on the next topics.

There are also 3 most common delivery methods. These methods are listed below:

1. Unicasting
2. Multicasting
3. Splitting

1.2 Historical Development

First transmission of signals was granted in the early 1920s by George SQUIER over electrical lines [8]. This invention was the basic of live streaming which provides continuous music and video to clients. But Real Networks [9] is the first company which delivered a live video on 1995. They streamed a baseball match over the Internet. After 1995, many companies started to use live streaming actively in the streaming media market.

1.3 Protocols

1.3.1 User Datagram Protocol

User Datagram Protocol (UDP) is the famous member of streaming protocol suite. UDP was designed by David Reed and defined in RFC 768 [10]. UDP send messages called datagrams. It does not verify the message if it is sent or not. In this manner, it uses a simple architecture with minimum mechanism. So there is no guarantee of delivery and the stream between source and destination is extremely unreliable.

If error checking and quality of service does not have a priority, UDP would be a better solution. Otherwise TCP, Transfer Control Protocol which provides handshaking facilities, is suitable for the design.

Here are the most common advantages of UDP:

1. Transfer is straight forward.
2. Datagrams are open to development in order to model other protocols.
3. It provides simple implementation and suitable scalability.
4. It is ideal for real time transmission.
5. It is free of retransmission and it makes it suitable for VoIP and online gaming.

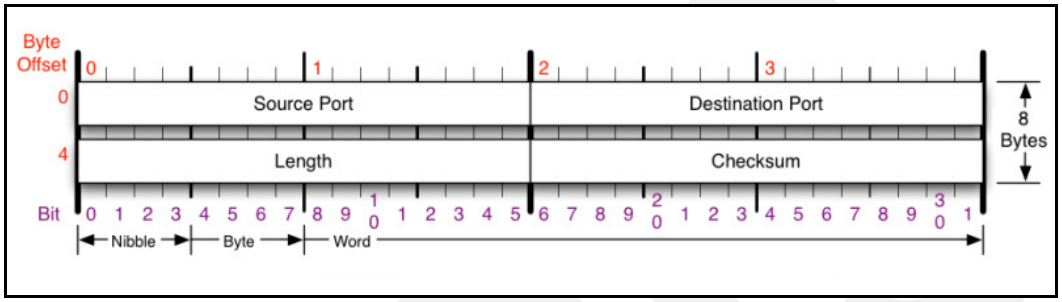

Figure 2. User Datagram Header Format

Datagram sockets are used to grant delivery streaming. As seen in Figure 2 above, socket needs to start from a source port and a destination port. Port field in the "Datagram Header Format" is 16 bits. So, port value must be between 0 and 65535. Ports between 49152 and 65535 can be used for any purpose since the ports between 0 and 49151 are occupied by various services and standards.

UDP messages are carried on IP and header contains 4 fields. Each field is 16 bits and source port field is optional in IPv6 version.

Here are the details of UDP fields [10]:

1. Source Port: It refers to sender's port and must be null if it is not used.
2. Destination Port: It is a required field and refers to receiver's port.
3. Length: This should be the total byte of header and datagram. It should be at least 8 bytes since the header is 8 bytes. IPv4 limits the maximum size of datagram to 65535 bytes + header. But IPv6 offers a better size of space for datagram.
4. Checksum: It is used to detect problems and optional in IPv4. Checksum is computed via summing all 16 bits using one's complement.

Official RFC 768 declares IP interface as follows:

> *"The UDP module must be able to determine the source and destination Internet addresses and the protocol field from the Internet header. One possible UDP/IP interface would return the whole Internet datagram including all of the Internet header in response to a receive operation. Such an interface would also allow the UDP to pass a full Internet datagram complete with header to the IP to send. The IP would verify certain fields for consistency and compute the Internet header checksum."* [10]

UDP uses a simple structure and may be a good solution if the confirmation is not required by the sender. For example, a VoIP user may receive the sound with a small delay or jitter. But the client does not need to verify the message if the problem is tolerable by the receiver.

UDP does not have any timeout, retransmission or acknowledgment mechanism. UDP is completely unreliable in this manner. So, if the streaming design is based on security feature, TCP should be preferred for the streaming design. Figure 3 shows the jitter frame in UDP due to lack of acknowledgement mechanism.

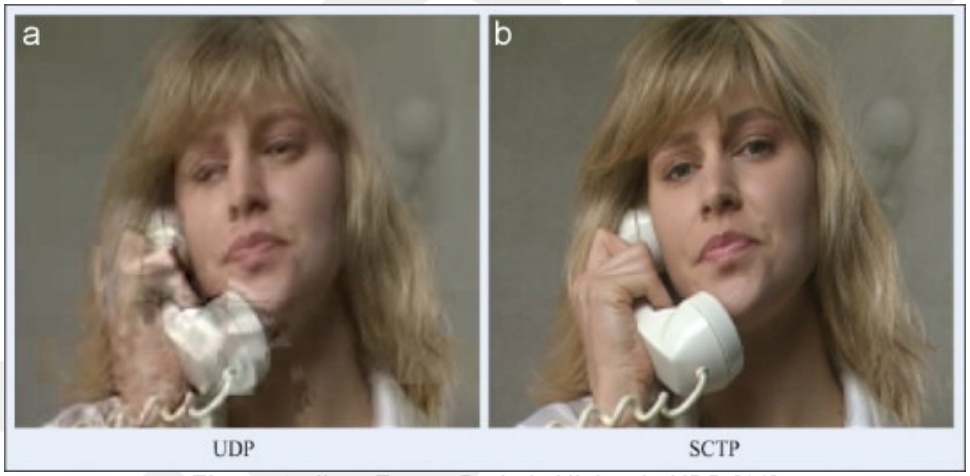

Figure 3. Jitter Frame Ratio is Higher in UDP [11]

UDP send messages one by one and it is possible for one message to reach to destination after its successor. UDP does not have any mechanism to control order problem. UDP is extremely lightweight because of not having control mechanism for error checking or retransmission etc. So it is carried on top of IP easily.

1.3.2 Real-time Transport Protocol

Real-time Transport Protocol (RTP) is an application layer protocol and especially used for streaming audio and video. It transmits real time data over packets and it has header and information sections that include retransmission and reordering control for any frames that are out of order. It also has identification section for encoding of media [12].

In Real-time Transfer Protocol, TCP and UDP can be used. Handshaking is optional and this feature is an advantage for RTP. RTP is usually combined with RTCP which includes the information about quality of service and statistics. If they will run together, RTP port should be an even number. RTP usually uses 1024 and 65535 ports.

RTP is designed for multimedia data to be streamed from one peer to end peer in real time and has capabilities to prevent jitter and adjust the problems. It support multicast functions like transferring media to multiple destinations. RTP is not efficient on audio delivery since the correction of lost packets consumes intolerable time. So, correction of one packet usually results in new packet lost. For this reason, RTP is usually based on UDP.

RTP has two sub-protocols:

1. Data Transfer Protocol
2. Control Protocol

Besides the protocols above, there are two more optional components:

1. Signaling Protocol
2. Media Description Protocol

RTP has a session mechanism and it grants a separate session for each multimedia stream. RTP handles different sessions for audio and video where each session has IP address and port numbers for RTP.

bit offset	0-1	2	3	4-7	8	9-15	16-31
0	Version	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC identifier						
96	CSRC identifiers ...						
96+32×CC	Profile-specific extension header ID						Extension header length
128+32×CC	Extension header ...						

Figure 4. RTP Packet Header [13]

RTP is based on application layer and suitable for integration to new formats such as H.264, MJPEG. RTP allows it in packet header and encoding information can be placed within RTP profile. RTP header has many optional fields after size of 12 bytes. In Figure 4, all fields are shown with offset values. Below are the fields of RTP header:

1. Version: It refers to protocol version.
2. P: Padding.
3. X: Extension.
4. CC: Contributing source count. It contains the identifiers.
5. M: Marker.
6. PT: Payload Type.
7. Sequence Number.
8. Timestamp.
9. SSRC: Identifier of synchronization source.
10. CSRC: Identifier of contributing source.
11. Extension Header. This field is optional.

RTP was initially designed for multimedia conferences for the multi-participants. But it also provides other functions like storing continuous data, distributed simulation and control applications [13].

RTP has not any control activity to make sure that delivery is successful or not. RTP itself does not guarantee the delivery and out of order problems. So, RTCP is used to monitor the quality of service and prevent the out of order problems [13]. RTP offers a new generation of application level protocol to be integrated with any kind of encoder and protocol.

1.3.3 Real Time Streaming Protocol

Real Time Streaming Protocol (RTSP) is a control protocol to deal with networking applications. It is widely used in TV media and communication systems. Protocol manages the transfer between end peers. It offers new generation actions such as pause, seek etc [17]. Once the stream is delivered to client, both server and client

have ability to obtain resource description, session link and control play via switching RTSP message [17].

There are two types of messages in the protocol. Request and response messages usually include state messages and responses to related requests. There are several types of request messages in RTSP. Figure 5 includes an example of client-server communication:

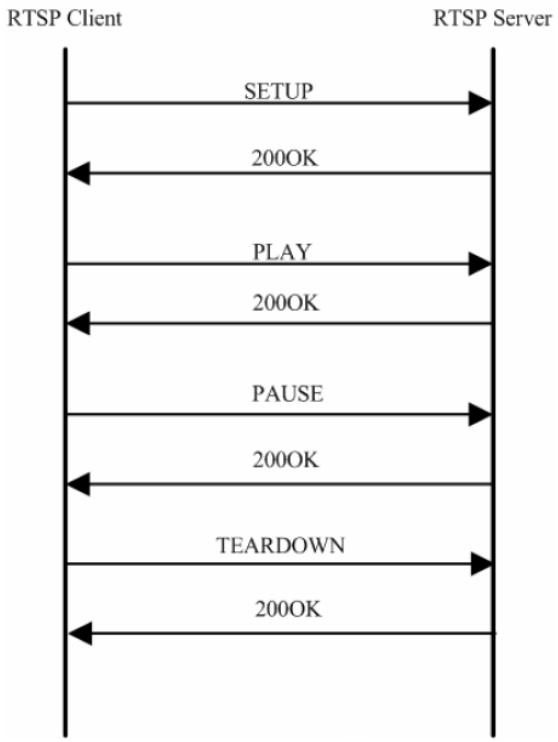

Figure 5. A Typical RTSP Client-Server Communication [17]

1. SETUP: Initiates the session and queries the server for any available stream resource.
2. PLAY: Simply delivers the media.
3. PAUSE: Stops stream but it does not break the session.
4. TEARDOWN: Breaks the session.
5. SET_PARAMETER: Sets parameters for the session.
6. GET_PARAMETER: Gets parameters from the session.

After the link of session, RTSP itself does not operate the control of streaming media. Most of the RTSP media servers use the Real-time Transport Protocol and Real-time Control Protocol to deliver the stream. On the other hand, some custom protocols can be used instead of RTP. For example, Real Network uses Real Data Transport (RDP) [18]. However it is not widely used in streaming market.

RTSP allows custom directions to manage streams. RTSP has ability to deal with concurrent sessions and control on state identifier. RTSP operates with TCP to establish connection between end points.

RTSP uses 554 as port number in transport layer and has OPTIONS feature like in HTTP requests. RTSP control messages are bidirectional. They can be sent from client to the server and from server to the client. For the available request types, OPTIONS request is sent from client to server.

Client to server:

```
1  OPTIONS rtsp://example.com/media.mp4 RTSP/1.0
2  CSeq: 1
3  Require: implicit-play
4  Proxy-Require: gzipped-messages
```

Response from server to client:

```
1  RTSP/1.0 200 OK
2  CSeq: 1
3  Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

All other requests must start with “rtsp://” and client should be able to understand the response format. RTSP is being widely used by many companies and applications. From the server side, there is a number of implementations. Some of them are:

1. FFmpeg [19]
2. Quicktime Streaming Server
3. VideoLAN
4. Windows Media Server
5. Youtube

Client side implementations are also being widely used. Some of them are:

1. cURL
2. Quicktime
3. RealPlayer
4. Skype
5. VLC Media Player

Real time Streaming Protocol is very popular in daily life and it is being used for streaming in set up boxes and media players as de facto standard.

1.3.4 Real Time Control Protocol

Real Time Control Protocol (RTCP) is designed for a service for RTP. RTP itself is not a reliable delivery protocol. For this reason, RTCP monitors the quality of service for RTP.

It has four components [14]:

1. Feedback mechanism for data distribution quality.
2. Granting an ID for each participant.
3. Scaling the control packet mechanism.
4. Session control information.

RTCP has ability to deal with multicast and unicast designs. Unicast designs provide many abilities including media delivery control and rate control for synchronization. RTCP uses the same address with the RTP, and packets are delivered via single session to all clients.

There are several types of RTCP packets [15]:

1. Source description
2. Sender report
3. Receiver report
4. BYE packets

RTCP also provides additional packets, application specific RTCP packets. For a particular RTP session, if the number of client grows, then RTCP traffic increases linearly. In this manner, bandwidth of RTCP is considerable. As seen in Figure 6

below, RTCP feedback messages are sent to each connected node separately. In this manner, if the number of receivers is too large, RTCP report delays can occur up to several minutes.

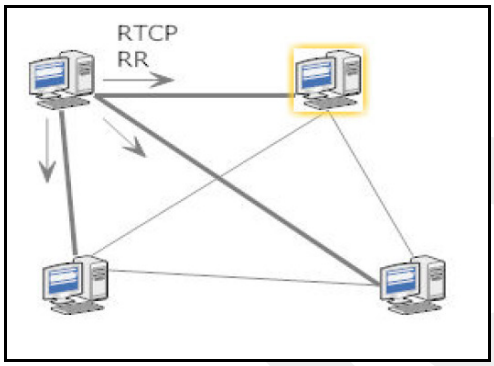

Figure 6. Feedback Architecture in RTCP [16]

1.3.5 Stream Control Transmission Protocol

Stream Control Transmission Protocol (SCTP) is a transport layer protocol and it offers advanced features unlike TCP over the IP based designs. It has some common features of UDP and TCP. For example, SCTP uses messages like UDP. However it provides some reliability features like TCP.

SCTP is a young protocol. However it was handled by RFC and standardized properly with the aid of IETF [20] [21]. SCTP offers reliable delivery service and it makes sure that packets are sent to end points without an error. SCTP provides session based delivery like TCP and the session is kept until the whole transmission is completed successfully. SCTP also provides advanced features for audio delivery and audio signaling.

Here are the main features of the protocol [21]:

1. Unicast: SCTP is oriented toward Unicast protocol. It supports the relation between two peers.
2. Reliable: SCTP can easily detect and correct the problems such as corruption, out of order sequence and duplication.
3. Message oriented: SCTP is designed message oriented like UDP. However it grants an implicit structure.
4. Rate adaptive: Similar to TCP, it controls the bandwidth and check the scalability of transfer.

5. Multi-stream: This feature divides the stream into small sub-streams to prevent data loss.

6. Multihoming: If the client has more than one IP address, it enables the transparent paths to the network. Figure 7 shows the typical structure of SCTP.

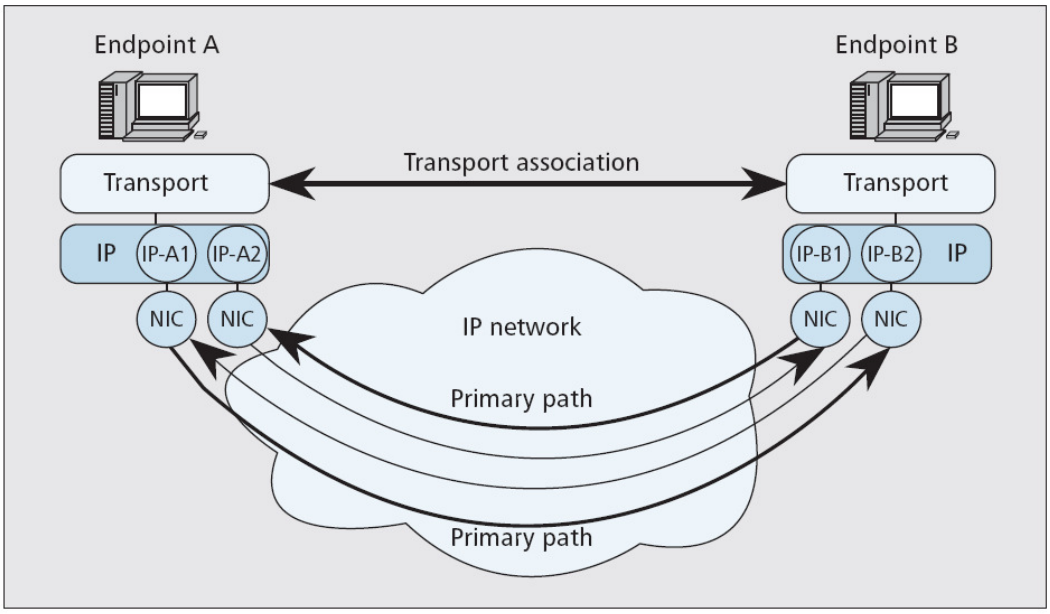

Figure 7. Typical Multihoming in SCTP [20]

SCTP is being used in almost all Linux distributions. The protocol is being used for signaling the networks and IP based signaling for UMTS networks. SCTP delivers the data to transport layer via messages and divides the messages into small chunks. Each chunk is identified by a header. Then each chunk is bundled to SCTP packets in order to submit them to Internet Protocol.

SCTP provides a simple packet structure. Each packet has two sections. Figure 8 includes the packet structure with offset bits:

1. Common header

2. Data chunks

Bits	0–7	8–15	16–23	24–31
+0	Source port		Destination port	
32	Verification tag			
64	Checksum			
96	Chunk 1 type	Chunk 1 flags	Chunk 1 length	
128	Chunk 1 data			
...	...			
...	Chunk N type	Chunk N flags	Chunk N length	
...	Chunk N data			

Figure 8. Packet Structure of SCTP

1.3.6 Transmission Control Protocol

Transmission Control Protocol (TCP) is a core protocol for Internet Protocol. Unlike SCTP, it supports byte stream and handles the delivery control with byte numbers. For this reason, it is called TCP/IP. It provides error checking mechanisms and consumes more time than UDP.

TCP has a reliable delivery mechanism and it guarantees the delivery of stream to the end point. It uses positive acknowledgements to deal with the retransmission of lost data over the network. As seen on the Figure 9, TCP contains two sections: Header and data where header itself consists of 10 fields.

Offsets Octet		0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port														
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																Urgent pointer (if URG set)														
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

Figure 9. TCP Header [22]

Source and destination ports are 16 bits which identify the sender and receiver ports. Sequence number is 32 bits. This field controls the order of packets and helps to fix out of order problems. Acknowledgement number is 32 bits which controls the reliability between receiver and sender. Data offset is 4 bits and reserved section is 3 bits. This field is planned to be used for the future. There are 9 flags and each flag is 1 bit. Window size, urgent pointer and checksum are 16 bits.

TCP has three types of operations [22]:

1. Establish a connection: Protocol uses handshake operation to establish a connection. Before a connection request, a server must listen to a port. After that, client can make a request. First, client sends SYN to the server. Server responses with SYN-ACK with a sequence identifier. Finally, client sends ACK message to the server including the sequence identifier that is received via SYN-ACK.

2. Terminate a connection: TCP uses 4 way handshaking to close the connection. FIN message is sent to terminate the connection. Then the other side responses with ACK and FIN messages. Finally, client sends ACK message to the other side and terminates the connection. Figure 10 shows the communication of termination between peers.

3. Protocol operations: TCP operations are handled by operating systems and it uses states to manage the operations [22]:

    a. LISTEN: Server listens to a particular port for connection request.

    b. SYN-SENT: Refers to a waiting state, after the connection request.

    c. SYN-RECEIVED: Refers to a confirmation state after the connection request.

    d. ESTABLISHED: Server or client is ready to send data.

    e. FIN-WAIT-1: Termination request for client and server.

    f. FIN-WAIT-2: Termination request without acknowledgement.

    g. CLOSE-WAIT: Waiting for termination.

    h. CLOSING: Waiting for an acknowledgement for termination.

    i. LAST-ACK: Waiting for the last approval for termination.

    j. TIME-WAIT: Refers to time in order to make sure that request is received.

    k. CLOSED: No connection exists.

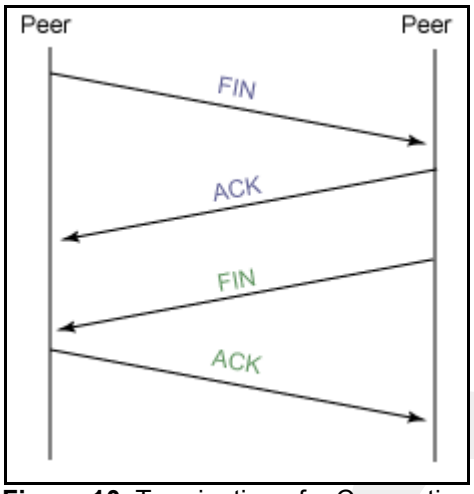

Figure 10. Termination of a Connection

TCP has many advantages compared with the UDP:

1. Sequence is kept ordered.
2. Retransmission is possible.
3. Congestion control.
4. Delivery control.

TCP identifies the byte data with a particular number. For example, bytes are labeled with 10, 11, 12, 13, 14 and 15. Once the 10 is sent, receiver responses ACK including byte number 10. After the delivery of 10, same procedure is followed by 11, 12, 13, 14 and 15. This mechanism provides error free data transfer.

1.3.7 Peer to peer

Peer to Peer (P2P) video streaming offers many new abilities for traditional protocols such as UDP and TCP. If scalability is an important issue for the stream, then P2P would be the best solution. In P2P streaming environment, each node can act as a stream source. So, a node can be both receiver and sender. In this manner, no dedicated infrastructure is required [23].

Even though it has many advantages, there are numerous challenges also. For example, packets are delivered to long haul of distance on an unreliable path between pairs. This causes delays and decrease in video quality.

In P2P systems, each node is equal and has their own tasks. So each node can deter the resources by itself. It can adjust the bandwidth, disk storage or processing power to other nodes. P2P systems are usually used at application layer and with an abstract overlay network. Indexing and searching peers are handled at this network.

P2P transfers and network are not defined with the terms of traditional client-server architecture. In client-server mode, client makes a request and server responses due to the request. P2P nodes are equal and they act as both client and server.

Each node is linked to each other. If a node knows the location of other node, then it creates a direct link to it.

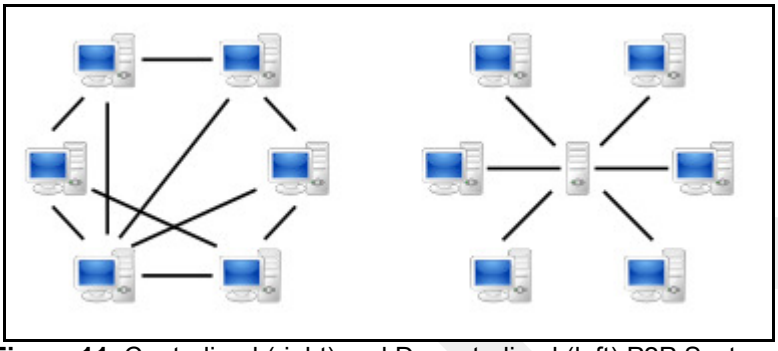

Figure 11. Centralized (right) and Decentralized (left) P2P Systems

There are two different networks based on linking two nodes [24]. Figure 11 includes the centralized and decentralized P2P systems in topological format:

1. Structured P2P Networks: Peers are linked with a particular algorithm and criteria. Distribution hash table is used for indexing. Structured networks provide high scalability and performance.
2. Unstructured P2P Networks: Simply, it does not have any central node and rule for linking. Nodes are connected via ad-hoc based network. There are three types of unstructured P2P networks:
    a. Pure P2P: Consists of one delivery layer and there is no any infrastructure network.
    b. Centralized P2P: A central node is used to index and it functions to bootstrap.
    c. Hybrid P2P: There are super nodes to handle infrastructure within closer nodes.

In decentralized P2P systems, distribution hash tables are used like a lookup service. Hash table includes key and hash value of key. Figure 12 illustrates the hashing table over distributed network:

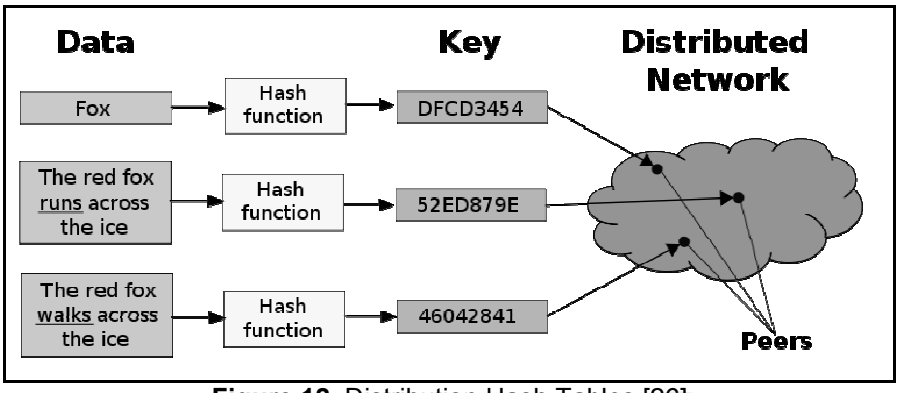

Figure 12. Distribution Hash Tables [26]

If a node looks for “Fox”, it gets the key for data “Fox”. Each node is responsible for mapping table. This feature provides more scalability and prevents the new node arrivals and fails.

Video streaming which is based on P2P system generally requires server, controller and peers. If a user wants to join a network, it sends a JOIN message to controller. Controller responses with an ACK message including network ID and connection is closed. After a successful log on, a node looks around for the object index. If it cannot find any index, it submits an UPDATE message to controller. Then, controller updates its index and sends an ACK response to the peer. The connection is closed [27].

There are numerous advantages of Peer-to-peer systems over classical client-server architecture:

1. Configuration and participation of nodes are easy to implement.
2. Each node shares its own resources. In client-server model, only server shares the data.
3. Reliable: If one node fails, system continues to work.
4. Each node is responsible for its system maintenance and a system administrator is not required for the system.
5. Cost of network is affordable.

There are also disadvantages as well as advantages:

1. Administration is difficult. Because the system is decentralized and peers cannot be controlled directly.
2. Security problems. Unwanted data can be transmitted over this network.

3. Data recovery is difficult since each node is responsible for its own computer and sharing.
4. Legal problems. Copyrighted files can be transmitted through the network.

P2P is extremely used in our daily life and especially content distribution is handled via P2P networks.

Here are some examples of P2P networks:

1. File sharing: Bittorent, G2, eDonkey.
2. P2P CDN: Peer-to-peer content delivery networks are iraffic, Kontiki etc.
3. Software distribution: Linux and some games uses software distribution.
4. Video streaming: P2PTV, PPLive, LiveStation and TVUPlayer.
5. Peercasting: Peercast, FreeCast, Rawflow and IceShare.
6. Communication: Skype uses P2P networks.

P2P Internet traffic occupies the 60% of current Internet traffic [23]. For this reason, some of ISPs limit the usage of P2P due to high bandwidth usage.

1.4 Delivery Methods

1.4.1 Unicasting

Unicasting transfers the data and stream to only single direction addressed with a unique one [28]. Figure 13 shows the network topology of unicasting:

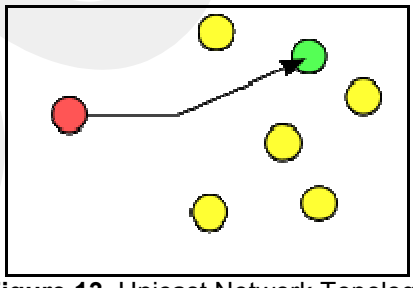

Figure 13. Unicast Network Topology

The most of the client-server traffic in Internet is based on unicasting. Simply, a client makes a request to a particular server via TCP/IP connection and server responses to client address. The path between the client and server is single and data is transmitted through one direction.

Unicasting is not efficient on video streaming applications since the delivery of data is one directional. If the number of receiver client is too large, then unicasting is not a better solution. Because video stream has to be duplicated for each client and it will consume the server's computing resources. Moreover, it consumes the bandwidth and decreases the video quality and performance.

As seen in Figure 14, unicasting uses session based protocols such as Transmission Control Protocol and User Datagram Protocol to deliver stream. Once a user connected to server, it establishes a direct communication with the server. Each client consumes its own dedicated bandwidth. For example, if a client consumes 20 KB per second, then 20 clients consumes 400 KB per second [29].

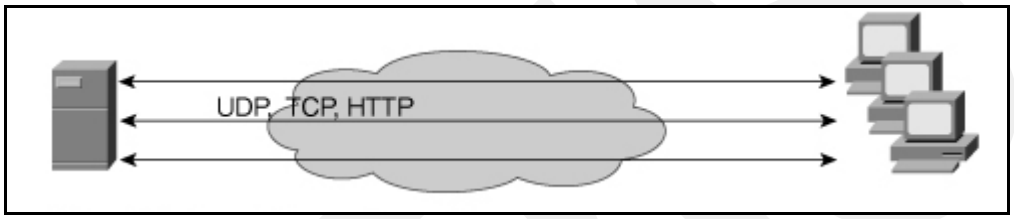

Figure 14. Unicast Protocols and Structure [30]

1.4.2 Splitting

Splitting is a hybrid model of unicasting and multicasting. Simply, stream source makes a connection to a media server and it does not communicate with clients directly. Relation between source and media server is based on unicasting.

A media server shares the stream with multicasting. Stream is delivered to multiple users at the same time. Relation between media server and clients is multicasting. Figure 15 shows the splitting topology which consists of a centralized media server:

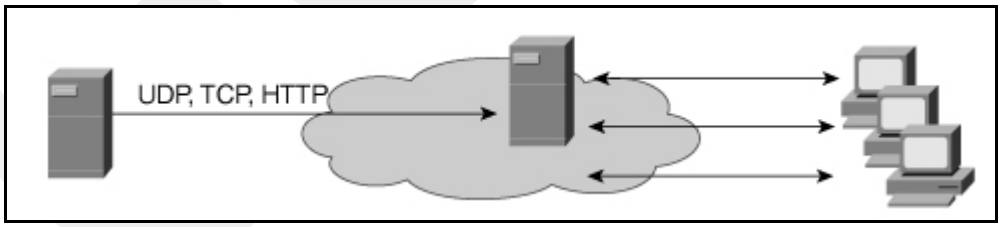

Figure 15. Splitting Topology [30]

1.4.3 Multicasting

Multicasting is a delivery method which is able to carry one message to multiple destinations at the same time within a single transmission line. As seen in Figure 16, a single copy is created from source and other copies are duplicated through network elements like routers and hubs [31].

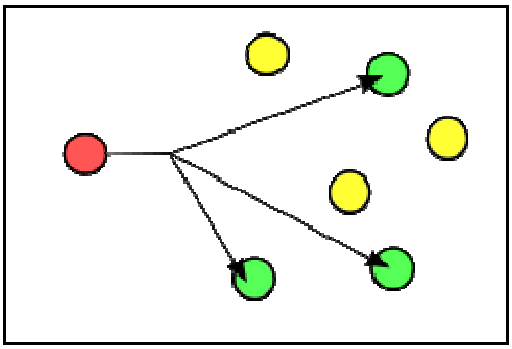

Figure 16. Multicasting Topology

Multicasting usually implemented in Internet Protocol (IP) for Internet TV streaming and media streaming. IP based multicasting is based on IP routing at router level on Data Link Layer. In IP broadcasting, messages are sent to all clients on the network including the clients who do not want to get message. This method is useful for satellite or similar wireless networks.

Routers are the basic players in multicasting since they are responsible for the route and the copy of the messages to clients. For this reason, ISPs must enable multicast features of the routers. Otherwise a multicast delivery is not possible. If a multicast network is planned over Internet or long haul distance, then VPN tunnels must be used to connect two multicast enabled routers.

Multicasting can be used on any kind of network that includes TCP/IP, ATM, Ethernet, frame relay and satellite [31].

Multicasting operates on User Datagram Protocol to deliver the messages. Clients do not send ACK messages to the sender since the nature of UDP is unreliable. So, there is not any handshaking protocol and delivery is not secure. Sender is not able to verify the delivery. At this point, UDP reduces the network traffic and gains the performance of overall data delivery to end points. Moreover, it provides more scalability to the network and stream can reach to more users.

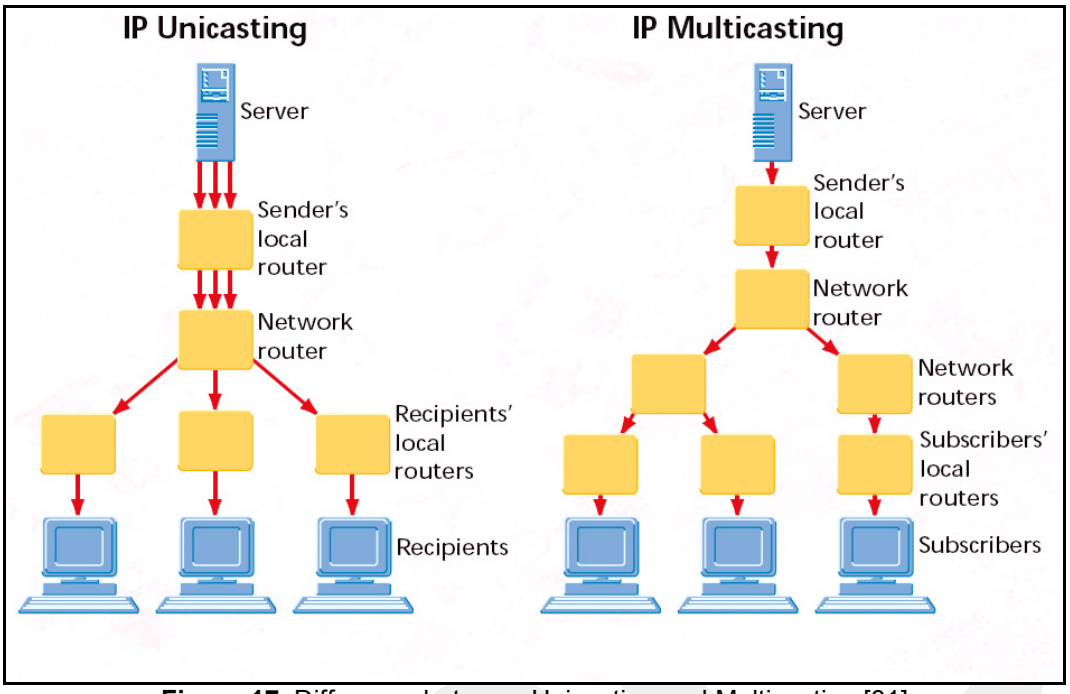

Figure 17. Difference between Unicasting and Multicasting [31]

As seen in Figure 17 above, stream is duplicated for each user in the source of data and delivered on the same transmission line. This method is not efficient if the number of client is too large due to lack of bandwidth.

However, in multicasting, stream is created once and delivered to the clients on the same transmission line. Copies are created on the network elements such as routers. This method is more efficient when compared with unicasting method if the number of receiver clients is increasing. Besides significant advantages of multicasting, there are some challenges at installation point.

Multicast is usually supported by many protocols such as User Datagram Protocol and applications. But in complex networks, maintaining a reliable, lossless delivery line is a serious problem. For example; constant data streams, reliable delivery, and managing the general communication require complicated configurations. However, there are still unsolved questions and they are waiting for researchers.

1.5 Streaming Codec

Codec part of the application is critical due to bandwidth problems while transferring stream to multiple clients. Another major problem with live video streaming is the quality of service. Frame rates and screen resolution of streaming video directly affect the satisfaction of the client and cause server to consume less system resources.

According to a research [41], H.264 codec was provided the best performance compared to H.263, XviD and Real Video. Figure 18 shows the results of research:

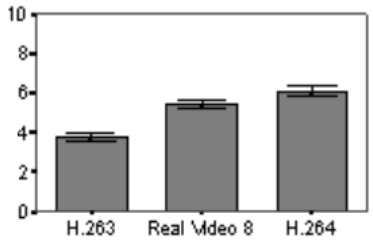

Figure 18. Quality Scores of Video Codecs [41]

According to another research [42], H.264 gives the best mean opinion score compared to Divx 6.0, XviD 1.1.0 and WMV 9.0. Figure 19 includes the scores of this research based on mean opinion:

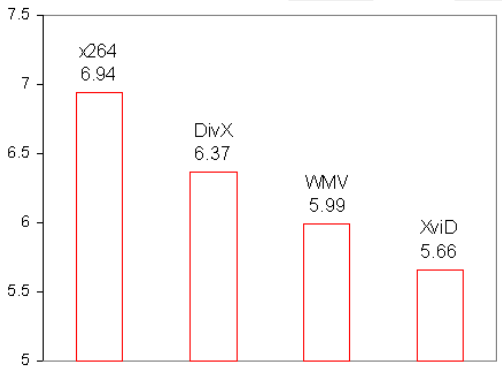

Figure 19. Average Mean Opinion Score for Codecs [42]

Due to performance results of two separate tests, H.264, DivX, WMV, XviD and Real Video codecs are reasonable for integrating live video streaming. Especially H.264 provides some additional quality of service advantages over live video streaming as well as bandwidth performance.

1.5.1 H.264/MPEG-4 AVC

H.264/MPEG-4 Advanced Video Coding also refers to H.264/MPEG-4 Part 10. This compression standard is being used widely for delivering high quality video and it is available since 2003 [43].

H.264 is a block oriented codec and widely used by many popular Internet sources such as Vimeo, YouTube. This codec is also being used in satellite based HDTVs and Blu-ray discs.

Here are some basic features of H.264 [44]:

1. Picture prediction based on block sizes 16x16, 8x8 and 4x4 with the aid of variable block-size motion compensation (VBSMC).
2. Prediction of edges between blocks.
3. Lossless coding.
4. Interlaced-scan video coding.
5. Transform design features within multiple block sizes.
6. Quantization design with step size control.
7. In-loop deblocking filter.
8. Entropy coding with CABAC and CAVLC.
9. Loss resilience features with network abstraction layer, flexible macroblock ordering, data partitioning, redundant slices and frame numbering.

Based on the features above, H.264 provides different profiles for application level purpose. Each profile activates the different features of H.264 due to purpose of application. Beside profiles, H.264 also uses levels to determine the performance of decoder for a profile. A level consists of following features:

1. Maximum decoding speed.
2. Maximum frame size.
3. Maximum bit rate.
4. Video resolution.

Maximum resolution is 4096x2304 with 56.3 frame rate.

1.5.2 DivX 6.0

DivX 6.0 is formerly known as MPEG-4 Part 2 and developed by MPEG. Similar to H.264, it provides some features such as profiles and levels. Each profile is created for different purposes and domains.

“MPEG-4 part 2” offers three basic profiles [45]:

1. Simple Profile: This profile is used where the resolution and bit rate is not prior to applications. VoIP is a typical example of Simple Profile.

2. Advanced Simple Profile: It provides more advanced features similar to H.263 including quantization, interlaced support, B-frames support, Qpel motion compensation and global motion compensation (GMC).

3. Simple Studio Profile: It consists of six levels. Each level includes separate bit depth, resolution, frame rate and data rate.

Despite the significant advantages, this codec is being criticized by many authorities including Michael NIEDERMAYER, who is a maintainer of FFmpeg [46].

1.5.3 Real Video

Real Video is a compression codec developed by Real Networks. They usually combine this codec with Real Audio and pack them as Real Media (.rm) container. Real Networks was the pioneer of the streaming market with the first live video streaming using Real Media format.

Real Media can be streamed over Internet via Real Time Streaming Protocol (RTSP). However, they control the sessions with RTSP and actual delivery is based on Real Data Transport (RDT). This caused integration problems by video players. For example, a user has to install Real Player to play Real Media [47].

Almost each Real Video releases use different compression formats. Here are the versions and compression formats [47]:

1. rv10 and rv13: H.263.
2. rv20: Real Video G2 based on H.263.
3. rv30: Real Video 8.
4. rv40: Real Video 9
5. rv40: Real Video 10.

1.5.4 Windows Media Video

Windows Media Video (WMV) is a codec developed by Microsoft. This codec was designed to stream video over Internet as competitor to Real Video. Currently, Windows Media Video released the 9th version [73].

Microsoft packed WMV as Advanced Systems Format (ASF) container to define streaming properties [48]. ASF consists of some properties related to media file such as copyright management and cryptography keys.

Older versions such as WMV 7 and WMV 8 codecs are competitors of MPEG-4 ASP. However, WMV 9 raised new properties and can be compared with H.264. WMV 9 has non-square integer transform feature that H.264 has not. Nevertheless, H.264 has additional advantages over WMV 9, such as low bit rates, reference B-frames, and in-loop filtering.

WMV codes are playable on almost all players. WMV codec became native in Windows operating systems since the Windows has a platform advantage. However, WMV can be played on all kind of platforms using third party encoders.

CHAPTER 2

CLIENT-SERVER ARCHITECTURE

Client-server architecture is a network topology that arranges the relation between nodes. It has been developed during 1970s and now it is one of the prominent member of computer networks [33].

Most of the services on the Internet are based on client-server architecture. For example, email services, web sites, game servers etc. are based on client-server architecture. Client-server model offers two types of nodes:

1. Server: A computer system that shares resources.
2. Client: An application or computer that requests a particular data from.

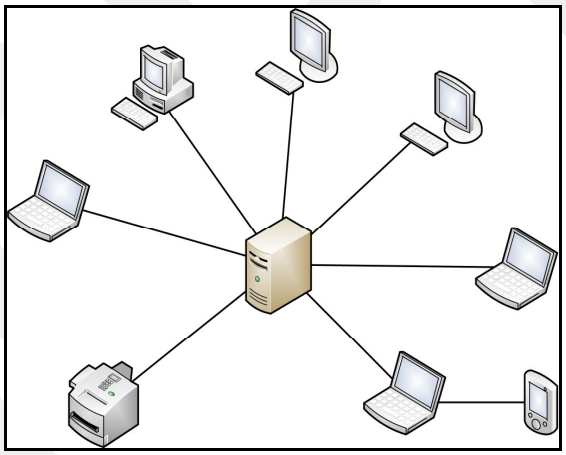

Figure 20. Client-Server Architecture

A server can share all resources such as data, storage, CPU and others. Figure 20 illustrates the usage capacity of client-server architecture. Client-server architecture is usually based on time sharing which allows multiple accesses to a single server at the same time. Server must serve to clients equally. Actually, servers can response to a single request at the same time. Time sharing makes it possible to handle clients.

Client-server architecture is designed in application layer and various protocols are used to maintain a valid communication between client and server. These protocols also refer to common language.

Both client and server can exist in a single computer. Since the client-server architecture requires application layer, they can exist and communicate within a computer. Moreover, a computer can consist of more than one server such as web server and database server. When compared to P2P networks, client-server architecture provides some additional advantages:

1. Accessibility: A server in client-server architecture is always ready to serve resources to client. In P2P, peers must be online to obtain particular data. Otherwise data is not accessible.
2. Cooperative: In client-server architecture, multiple users can access to a single copy of data within a central, common platform. Update and support control is easier.
3. Centralized: Server facilities such as backup and maintenance are easier and cheaper in terms of cost.
4. Security: Client-server architecture provides more security opportunities compared to P2P. Since the configuration and permissions are single within a server, it is not as complicated as P2P systems.
5. Performance: Path between client and server is single. So, the transmission between server and client is straight forward. Messages do not have to traverse around routers and nodes.
6. Reliability: There are no third party nodes between server and client. So the data is more secure.

In this thesis, client-server architecture was preferred for accessibility, centralized features and performance features to provide a better live video streaming experience to the clients. Because quality of service and performance are the most important keys for media streaming that satisfy the client's video experience.

CHAPTER 3

RELATIONAL DATABASE

Relational database is a set of data organized in a predefined structure. Relational model is used to create a relational database. The most popular database management systems are based on relational database [34]. A relational database consists of tables and a table contains records called “tuple”. Tables also have attribute that refers to column.

MySQL is a popular relational database management system and has some terms which have equivalent in relational database terminology. In relational database, each row has the same attributes and the relation is defined within a table that consists of rows and columns. Relation is provided via operation commands. Insert, delete and update commands modify the relation. A primary key is defined within a table to distinguish all of the rows. Each primary key must be unique [35].

A foreign key is the primary key of another table that is used in an attribute of the table. Foreign keys do not need to be unique value. Foreign keys are essential for querying multiple tables [35].

Relational database systems come with data redundancy problems. Briefly, data redundancy occurs if a field appears more than once in a database. In this case, different values of redundant data can be fetched through multiple database operations. For instance, different names of a customer can be obtained for the multiple orders of customer. However, programmers are able to overcome this problem using foreign keys properly.

Indexes are the major players of relational database. They provide a faster search within high number of rows. Indexes use B+ trees and decrease the lookup in table. It gains the performance dramatically. There are four mathematical set operations to query relational database using relational calculus and algebra [35]:

1. UNION: Combines rows and removes duplicates.
2. INTERSECTION: Produces the differences of two relations.

3. DIFFERENCE: Produces the tuples that exist in first relation but not exist in second relation.

4. CARTESIAN PRODUCT: Combines all elements of first relation with the elements of second relation.

According to Techtarget.com [36], Oracle is the leading company that uses commercial relational database. On the other hand, MySQL is the leading DBMS in open source league. According to another market share report, Oracle is deployed %70 and SQL server is deployed %68. Figure 21 shows the market share of relational database management systems [37].

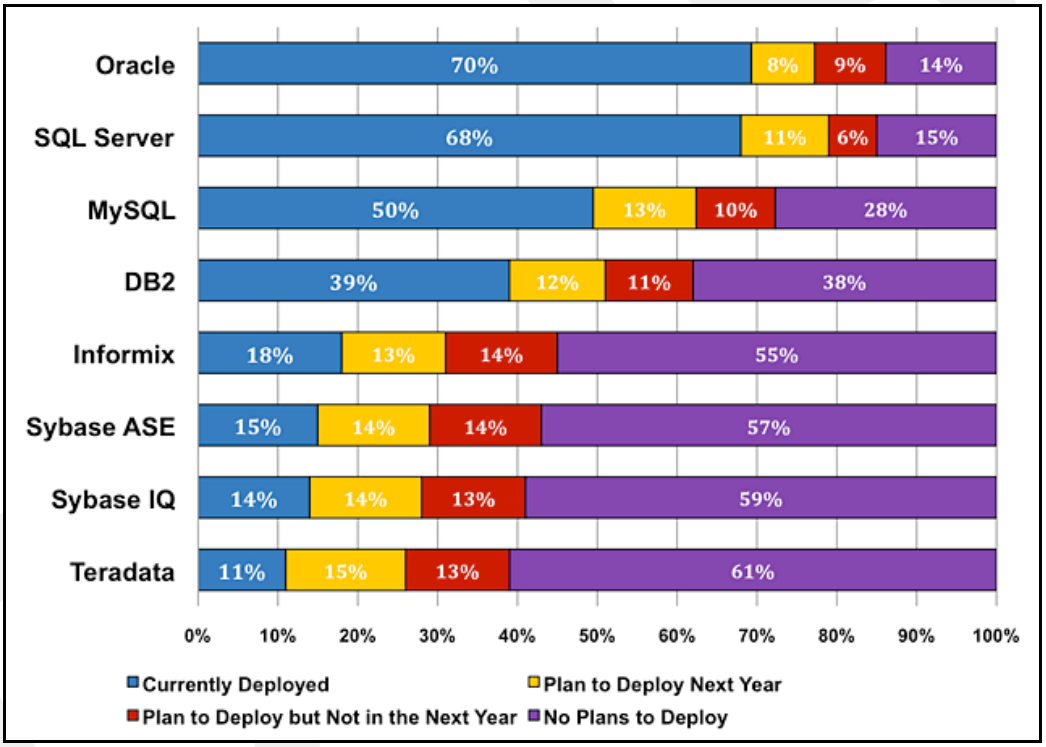

Figure 21. Market Share of Relational Database Management Systems [37]

In this study, MySQL is used to handle relational database queries since it provides free, fast and quick solution.

CHAPTER 4

PREVIOUS ACADEMIC WORKS ON LIVE VIDEO STREAMING

There are many active research topics on live video streaming since the live streaming and IP based TVs are popular and commercial nowadays.

The idea of streaming live video over database itself is a new topic in live video streaming market. However, multicasting based solutions are the major topics in research areas since they are more commercial due to scope of streaming. Unicast streams are extremely limited with the processing resources of server such as CPU and RAM. In other words, hardware limits the service capacity of the server. On the other hand, bandwidth between video source and client also limits the stream since the transmission line is single because of the nature of client-server architecture.

Approach of delivering video over database is a typical unicasting streaming. For this reason, this study can be compared with only unicast based streaming approaches such as TCP socket oriented streaming. Researches on live video streaming focuses on two different delivery concepts: Unicasting and multicasting.

4.1 Multicasting

Multicast solutions are the most popular topic of video streaming market because of new generation IPTVs and sharing applications. Although they have many advantages, they usually need to have a special configuration on network or a client application to handle P2P stream.

The latest active researches are listed below:

1. “Feature Research on Unstructured P2P Multicast Video Streaming” by Yang YUEXIANG, Liu CHAOBIN, Huang GAOPING [52].

   This article defines the security problems and offers four solutions to distinguish P2P multicast video streaming from P2P file streaming. As a result, they provide detection and identification of P2P multicast video streaming with %87 accuracy rate.

2. “Optimized Channel Rate Allocation for H.264/AVC Scalable Video Multicast Streaming over Heterogeneous Networks” by Bin ZHANG, Xiang LI, M. WIEN, J. OHM [53].

   This study offers an allocation for bitrate to different kind of network abstraction level based on H.264/AVC. As a result, paper offers a new and fast algorithm that provides a suitable protection approach on the network abstraction level.

3. “P2P Multicasting Network Design Problem — Heuristic approach” by Krzysztof WALKOWIAK [50].

   The paper focuses on P2P multicasting design problems due to rapid deployment and low costs. This study aims to reduce overlay network cost via a heuristic algorithm. Therefore, tests were done for various amounts of nodes and results were not computed in larger networks. Nevertheless, cost of a network can be determined through the experiments.

4. “Survivability of P2P Multicasting” by Krzysztof WALKOWIAK [51].

   This paper was published in 2009 and it describes the key points on guaranteeing the delivery to end point. In some cases such as delivering stock data, security updates, etc., content of delivery material is very important. For this reason, paper offers particular solutions to overcome this problem.

This study is not directly comparable with multicasting solutions since it is based on unicasting. For this reason, challenges of multicasting solutions are not the problems of this study. The reason is related to three key topics of video streaming: First, this study offers a simple network topology while P2P needs a complicated and well-structured network design. So, a P2P network must be designed carefully to provide a fast and reliable delivery. Second, P2P solutions are not able to provide a reasonable delay compared with unicast solutions. Because, video frames traverse through multiple nodes over complicated network. The last problem is security. This study provides a full control over the stream. Each step of the stream can be programmable and controllable.

4.2 Unicasting

Unicasting solutions offer small video streaming scope compared with multicasting solutions. However, unicast based streaming solutions are more suitable for

businesses and individuals if they offer a low scope video streaming due to rapid deployment and compatibility.

Unicast based video streams can be fetched in any kind of network topology without an additional network configuration. Moreover, unicast video streams can be controlled and specialized for any unique video streaming design. For these reasons, if the scope of video streaming is limited, then unicasting would be a better solution to provide a fast, reliable and stable delivery.

Here are the most recent works on unicasting:

1. “Maximizing Video Quality for Several Unicast Streams in a Multipath Overlay Network” by S. BOUDKO, W. LEISTER, C. GRIWODZ, P. HALVORSEN [54].

   Video streams that are based on overlay networks for multipath need to manage available bandwidth for all clients. At this point some decisions are required. This study offers a scenario with a benchmarking system to determine the optimal solution for each delivery path.

2. “Adaptive Unicast Video Streaming With Rateless Codes and Feedback” by S. AHMAD, R. HAMZAOUI, M. AL-AKAIDI [55].

   This paper offers an adaptive error correction to keep video stream undamaged. According to paper, channel code rate is determined in advance with an estimated pocket loss rate. However, health of the network is not stable and predictable.

3. “Advanced Rate Adaption for Unicast Streaming of Scalable Video” by C. LIU, I. BOUAZIZI, M. GABBOUJ [56].

   A paper from 2010 provides a mechanism called “Multiple Virtual Client Buffer Feedback”. It includes various information about sub-streams in scalable media streaming. This mechanism is proposed as an alternative to Packet-Switched Streaming Service (PSS).

4. “Real Time Video Streaming over Heterogeneous Networks” by M. QADEER, R. AHMAD, M. KHAN, T. AHMAD [57].

   Article defines the key features of delivering video stream over heterogeneous networks. Bluetooth, Wi-Fi and GPRS-EDGE networks are

used with MP4 simple profile and IETF protocols RTSP, RTP, and RTCP to overcome problems such as error correction, bit rate and bandwidth.

Live video streaming that is based on unicasting provides fast and reliable solution. However, service capacity of unicasting is extremely limited. Because video source is only available in server, and stream must be duplicated for each additional stream request. On the other hand, bandwidth limits quality of video due to single transmission line. This study offers a new approach to reach more clients compared with traditional unicasting solutions using relational database and web server.

CHAPTER 5

PROPOSED SOLUTION

Live video streaming is a prominent topic of data delivery in computer networks. Especially, IPTV and digital media streaming are the popular topics in the world since the traditional TV replaced with digital streaming. Moreover, live video streaming is being used in wide range of categories from military to medical projects.

High bandwidth capacities enabled streaming to be used in many areas and will be the hot topic of communication systems. Besides the growing need of live video streaming, there are serious challenges:

1. Performance: If a single stream has to arrive to thousands of people, there exists delay and quality of service problems. For this reason, a client may receive the video or TV broadcast after a minute with a low resolution. On the other hand, lost frames or packets are occurred due to the distance between source and destination.
2. Download time: If the stream is on a limited bandwidth capacity, then users have to wait for frames to be loaded. A better download time satisfies the user experiences. So, stream source must interact with the end user and adjust the resolution to fill the bandwidth of transmission line between video stream source and destination.
3. Response time and delay: Delay and response time increase proportional to number of connected clients. In other words, stream must buffer the stream for all kinds of network types. For this reason, response time is one of the challenging problems.

In this manner, desktop application implements two unicast streaming architectures to make a proper comparison. Here are the architectures that are implemented within the desktop application part of the study:

1. Live video streaming based on database and client-server architecture: It consists of two tiers. First one is implemented via desktop application and other one is via PHP to handle stream over database and web server.

2. Streaming via TCP sockets: This approach was implemented in Microsoft Visual Studio 2008 with C# programming language and it listens to a particular port within the server.

Streaming is usually operated under P2P or multicast based connections. In spite of reasonable features of P2P and multicast, there still exist problems that are waiting for researchers [38].

This study aims to provide client-server architecture to solve performance problems with a location aware approach. Location awareness allows this structure to be accessed from multiple domains like embedded devices and Internet based TVs. In other words, stream can be accessible over web server where the Internet connection is available.

Desktop and web application use relational database management system to store and retrieve stream in a fragmented format to share it to multiple users. Relational database allows querying in a semantic way to solve data access problem.

As a result, we expect a more reliable, secure, and faster live video streaming using relational database.

5.1 Goals

In this section, advantages of the study are going to be discussed. Besides significant advantages of P2P streaming and multicast, they have still problems on several issues. These problems occur usually in gaining the number of connected client or delay and security problems.

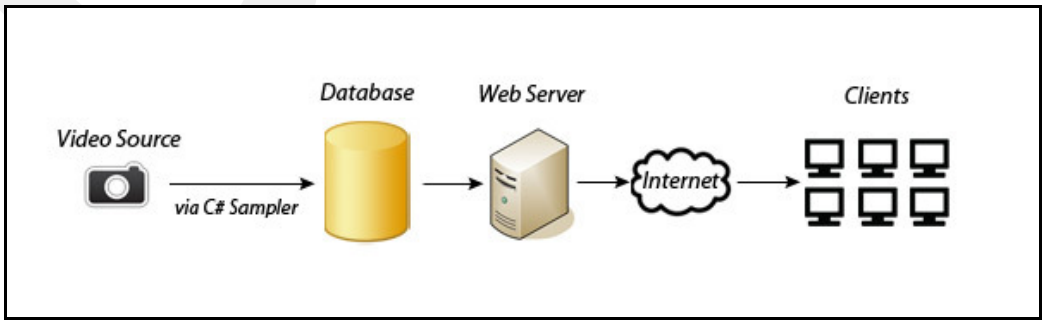

Figure 22. Flow Diagram of Proposed Solution

As seen in Figure 22 above, video is fragmented by desktop application and each fragment is delivered to relational database and it is stored in a separate row. Then it is served to clients over “Web Server”. Architecture offers an easy installation and access to data over the Internet.

Here are the goals of “A Client-Server Architecture for Live Video Streaming Using Object Relational Database”:

1. Client-server architecture: Client-server approach provides more security and reliability options to streaming design. Moreover, transmission line can be under control and streaming parameters can be tweaked easily. For example, if the bandwidth of transmission line is not sufficient, video resolution can be decreased easily. Compared with other approaches, it is more reliable and secure especially for military and medical streaming designs.
2. Download time: P2P and multicast based solutions are lack of download time due to number of subscribers and bandwidth capacity. This study aims to minimize the download time via setting buffers in relational database.
3. Response time: Using the abilities of client-server architecture, a better response time may be obtained even the number of clients is too large. While video streaming deals with multiple clients, web and database server can handle the link between source and destination.
4. Scalability: Scalability of live video streaming design depends on the scalability of web server. With today’s technology, client-server based solutions can be easily scaled.
5. Location aware: In client-server based live video streaming architecture, stream is reachable where the Internet connection exists. Moreover, stream is delivered via a single URL. It is easily fetched by embedded devices or within a software flow. On the other hand, stream can be queried via database server and delivered in a custom form.
6. Easy data access: In client-server streaming design, stream is always ready to go with a single SQL command. Stream is available on multiple domains at the same time.
7. Lossless delivery: Client-server based streaming allows a full control over the delivery. Tolerance depends on the tweak settings of streamer application.

For this reason, if there is no bandwidth problem between source and destination, delivery goes lossless.

8. Secure and reliable: This study offers streaming over HTTP connection. Thus, stream is delivered via single URL. If a secure connection is maintained via SSL certificate, a secure and reliable communication will exist between source and destination.

9. Web and DB server support: Web servers and database servers are specialized for the best delivery experience between client and server. So, an integrated solution gains streaming and provides some additional advantages. For example, web server has a great ability to host multiple clients at the same time. It opens slots for new connections and waits for the best performance. This feature provides faster response and minimum delay. Similarly, database server makes it possible to deliver data on the fly.

5.2 Challenges

Client-server based live video streaming design has some challenges even though it has many advantages.

There are three basic challenges as follows:

5.2.1 Bandwidth

Video delivery consumes more bandwidth resources compared with MPEG based delivery since MJPEG was used as video codec in the study. Especially MP4 is very bandwidth friendly.

"International Cablemakers Federation" publishes in a bandwidth requirement report as follows:

> *"The currently low penetration figures for IPTV and HDTV do not mean that these technologies have failed in the market. Rather, they mean that the growth is yet to come, and the bandwidth requirements for digital video may encounter a strong surge as more households adopt these technologies in three to five years. Transmission speeds needed to support HDTV depend on the compression technology. With MPEG-2 compression, an HDTV signal will require 15 Mbps to 20 Mbps. As of early 2007, MPEG-4 was not generally available, but estimates for streaming HDTV over digital channels ranged from 5 Mbps to 10 Mbps – about five times more than standard-definition TV."* [39]

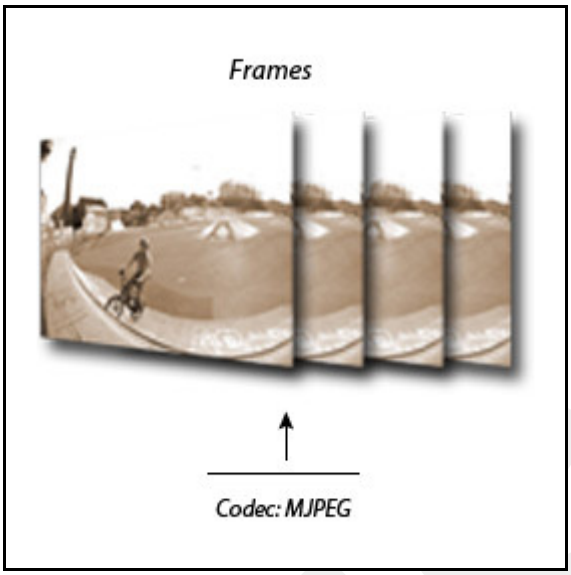

Figure 23. MJPEG Delivery

MP4 is a bandwidth friendly codec since it uses a special algorithm in video decoding [40]. In MP4 decoding, each frame contains the difference of previous frame. For example, if the video streams white frames within a time interval, bandwidth requirement dramatically decreases. Because the difference is almost zero. MP4 is ideal for IPTV solutions.

As seen on the Figure 23, MJPEG deliver the stream as compressed JPEG images one by one to client. It does not connect frames with a kind of relation. For this reason, MJPEG needs more bandwidth resources compared with MP4 solutions.

However MJPEG provides easy installation for web scripting languages like PHP. The relation of frames in MP4 makes it harder to decode within a live video streaming since the header part of MP4 like metadata, relation variables and overheads occupies a serious space in MP4 codec.

5.2.2 Hardware cost

Client-server based architecture with relational database needs some extra services and system resources. In addition to streaming costs such as bandwidth; maintenance of web server and database server are required.

Scalability is another key for cost. If the number of clients increases either new hardware or new server must be added to current network. Unlike P2P and multicast streaming design, server costs are prominent problem of client-server architecture since the stream source server.

5.2.3 Processing cost

Beside hardware costs, web server and database server need more processing resources such as RAM and CPUs. Because, there are more overhead that are transmitted within server. Each movement of data requires an additional processing cost. For this reason, client-server architecture with services requires more processing power. But each additional client can compensate the processing resources due to efficiency of web and database server.

5.3 Application Flow

There are three parallel flows in desktop and web application:

1. Web application: A starter PHP file initiates the client-server live video streaming over relational database.
2. Desktop application, client-server streamer: This section fetches video samples from the input and stores them in database.
3. Desktop application, TCP based streamer: This section listens to a particular port of host based on TCP sockets.

During the streaming, some common variables such as capture rate and buffer size coordinate the whole system. Here are the global, project wide variables that are operated by whole system. Figure 24 includes the screenshot of test platform which consists of common variables.

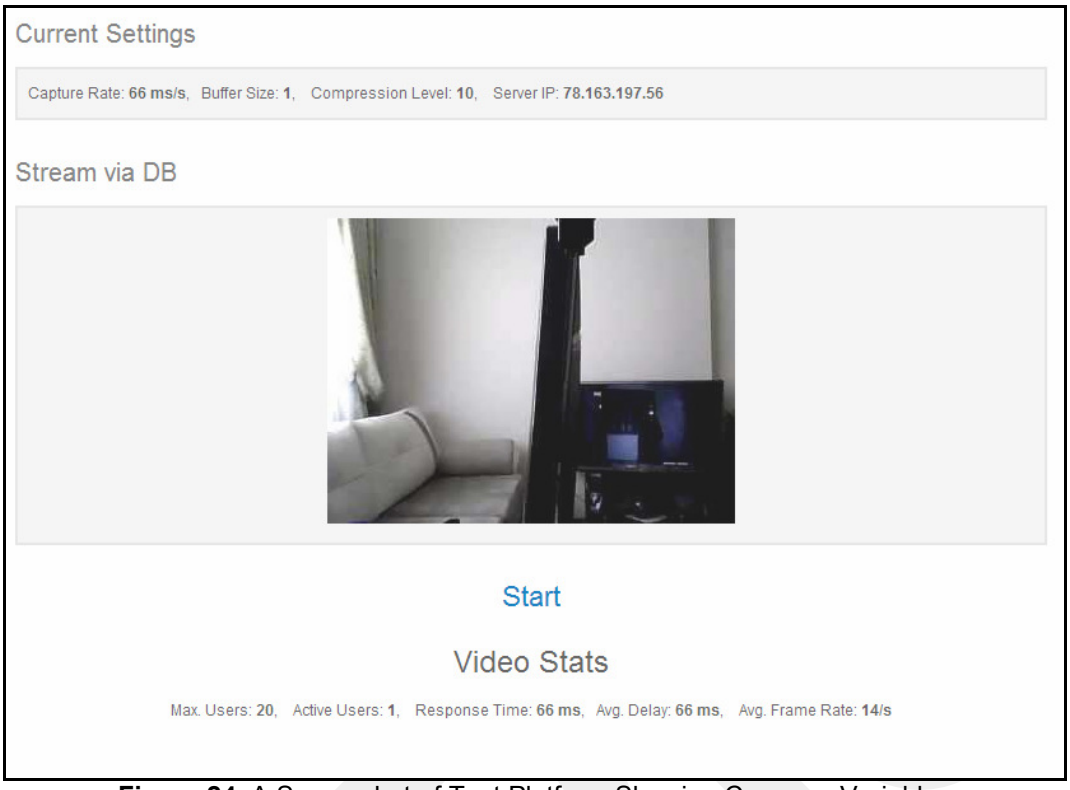

Figure 24. A Screenshot of Test Platform Showing Common Variables

5.3.1 Capture rate

Capture rate is the time between two frames. Usually it is defined as total frames/second. Default value is set to 15 frames per second. This value is almost equal to 66 milliseconds between two frames. Since there are two tiers in the study, coordination must be granted between tiers. For example, desktop application produces the samples using video input with predefined capture rate. However, web server must know the same capture rate of the stream to show a proper video.

5.3.2 Buffer size

Simply, buffer size is equal to row number of a table in relational database. It holds the frames which are sampled via desktop application. Simply, it provides efficiency to streaming even though it increases the delay and response time. More buffer size provides more clients due to flexibility of time.

It is a “First In, Last Out” buffer. In other words, when a frame stored to table, oldest one is removed from the table.

5.3.3 Compression level

Compression level refers to JPEG compression rate. When a sample was grabbed from video input source, it is converted to JPEG, compressed and delivered to buffer

in database. Figure 25 shows the screen with different compression levels. Quality of picture goes worse while compression level decreases.

Compression level is an integer value between 0 and 100. If the value takes 100, it means there is no compression. If it takes 0, it provides maximum compression. Compression is required to minimize the storage requirements of each frame. For example, leaving compression level at 100 will decrease the bandwidth performance. Default value is assigned to 10 where it satisfies the quality of service.

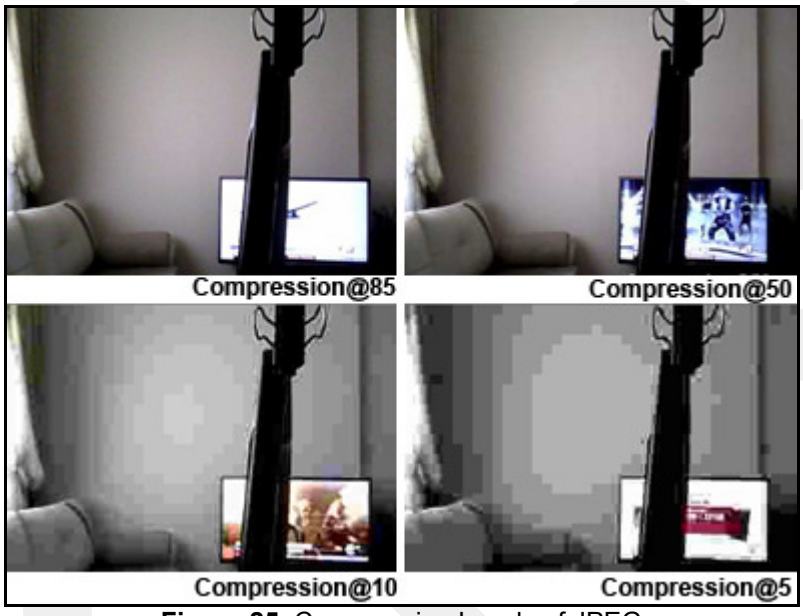

Figure 25. Compression Levels of JPEG

5.3.4 Current user number

Current user number keeps the amount of active users within streaming system. This parameter is used to control the maximum user amount in order to measure test metrics between TCP and database oriented live streaming application.

If the user number is equal or more then maximum user number, requests of new clients are refused and dropped. Otherwise, it is accepted and user number is incremented. At the end of the session, user number is decremented.

5.3.5 Maximum allowed user number

Maximum user number is a variable of test phase. It keeps the user number under maximum user number. Thus, it makes testing easier if the test is oriented around user number. For example, keeping all the parameters constant, a test scenario is handled by changing the user number.

5.3.6 Maximum loss frame rate

Maximum loss frame rate is a parameter to allow streamer in web server to skip frames. This parameter is the exact value of maximum failed frame number as a block. In other words, if the rate is 20, streamer must skip 20 frames at once to raise the flag and stream stops for a particular client.

Client-server architecture over HTTP does not have a built-in tolerance for non-existent content. For example, if a HTML file does not exist in a web server, a server response with “404” message is returned and connection is terminated. This situation is not an appropriate state for a video streaming design since the connection between server and client must be alive and active; so it must not be terminated. This parameter provides a continuous stream over HTTP.

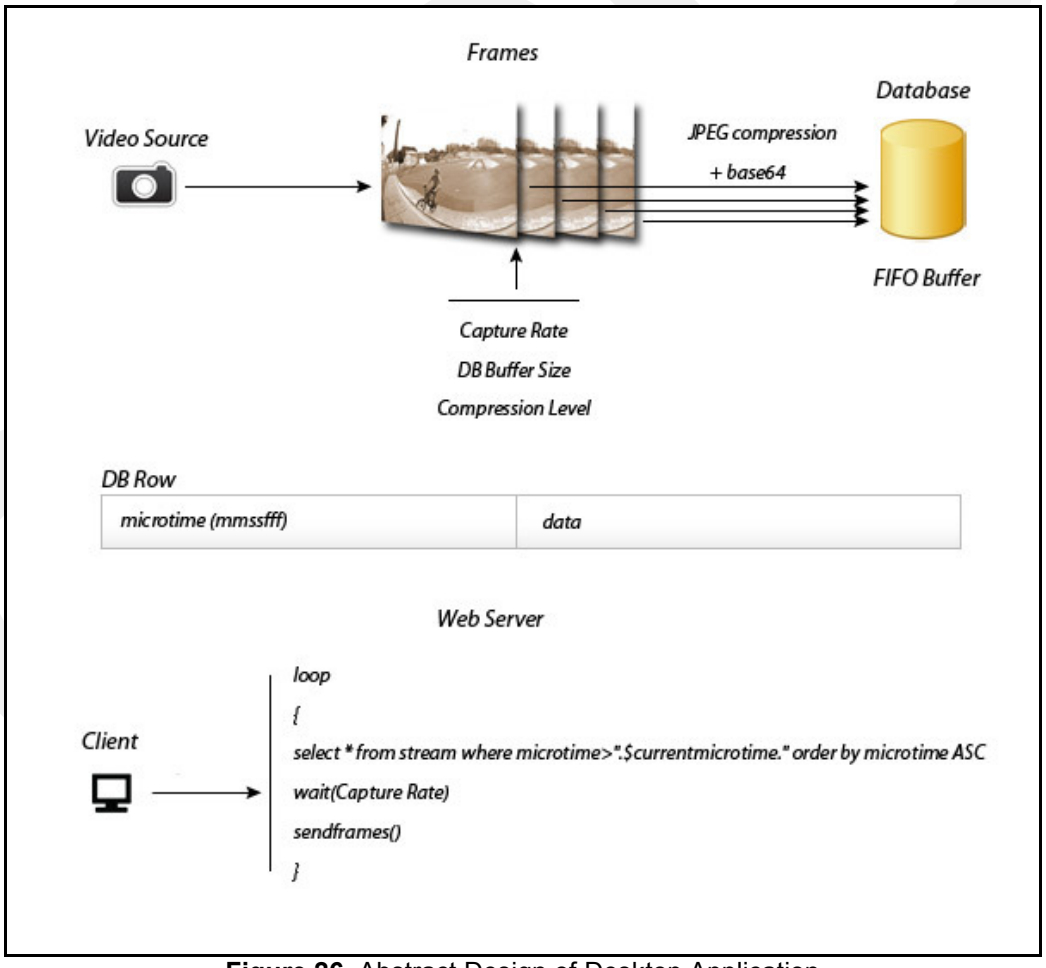

Figure 26. Abstract Design of Desktop Application

5.3.7 Web Server

If a client requests a stream, it should send it through an URL. Following address is a typical address to obtain stream. Port number 80 must be forwarded to local computer properly: http://78.162.239.45/stream.mjpeg. Once the request arrives to web server, “stream.php” initiates the stream with the steps below:

1. Constants are defined to prevent corruptions in the streaming time. PHP requires setting memory limit and error reporting manually. Moreover, a timeout must be set manually to deal with unexpected termination of connection between client and server.
2. Database connection is defined in a separate file and the file is included in stream.php.
3. Parameters are gathered from database. These parameters are current user number, maximum allowed user number, capture rate, buffer size, compression rate, maximum frame loss rate.
4. Header parameters are created. Each header text is included with header () function of PHP. Following header values are set respectively:
    a. Connection: close
    b. Cache-Control: no-store, no-cache, must-revalidate, pre-check=0, post-check=0, max-age=0
    c. Cache-Control: private
    d. Pragma: no-cache
    e. Expires: -1
    f. Content-type: multipart/x-mixed-replace; boundary=$myboundary

    Value of “$myboundary” is a delimiter between frames of MJPEG. In this manner, it can be any unique text value.
5. New generation browsers usually support output compression. In other words, server sends the content such as images, HTML files in a compressed format. For this reason GZIP compression is disabled and output is flushed to client immediately.

6. In a while loop, frames are scanned through a SQL command with the aid of micro time variable. Micro time consists of a set of time variables: Minute, second, and millisecond respectively. For example, 1514215 is extracted as 15 as minute, 14 as seconds and 215 as milliseconds.

    a. If the flag of “Maximum frame rate loss” rises, session is terminated and connection is closed.

    b. Frame is captured from database as raw data and decoded with base64 function.

    c. JPEG data and boundary value are printed to client respectively.

    d. Statistic values are updated. Response time, delay and frame rate are recorded for test environment.

7. At the end of the video stream, user number is decremented and connection is terminated.

5.3.8 Client-server streamer of desktop application

In order to get stream via web server, another application must sample the video source and store frames in database. This part handles this phase of the application. Initial parameters are set through the desktop application. Without stream sampler, web server does not deliver the stream to the clients since there will be no rows in the database. Once the camera source selected and started, stream can be started via a button called “Start DB”.

Here are the steps of client-server streamer:

1. Common parameters are gathered. These are capture rate, db buffer size, compression rate, frame loss rate and maximum allowed users. These parameters are set on the GUI of desktop application.

2. Public IP address of the server is obtained using dyndns.org [49]. Simply, web page including IP address of dyndns.org is scrapped and IP address is extracted from the HTML.

3. Common parameters in Step 1 are stored in database for the use of web server. Thus, web server can obtain settings through database.

4. A new thread is created for the rest of the job. A thread is required to deal with sleep problem in C# programming language. In C#, sleep function

makes the whole GUI frozen until the end of the sleep period. Sleep functions are used to grab video samples in time interval. Due to interval, sampler part of the application must sleep in terms of capture rate to provide a seamless video stream.

5. Redundant data rows are removed.
6. While loop initiates the sampling of video source;
    a. A video sample is grabbed and encoded with base64.
    b. Micro time is assigned as “mmssfff” where “mm” is minute, “ss” is second and “fff” is millisecond.
    c. Raw data is inserted to database with micro time.
    d. If database buffer exceeds the limit, then old rows are deleted from database.

5.3.9 TCP based streamer of desktop application

TCP works with sockets and must seed a particular port on the server. Similar to client-server based solution, frames are sampled and delivered to TCP socket. This phase of the application was implemented for comparison to client-server based solution. Since the both solutions are unicast, it is easier to compare two approaches.

Following steps define the key processes of TCP based solution:

1. Common variables are gathered from GUI of desktop application.
2. Public IP address is fetched from dyndns.org.
3. Common parameters are stored in database.
4. A thread is created for sampler. As discussed in client-server streamer, thread is required to prevent lock of GUI.
5. An “HttpListener” is created for a particular IP address and port such as http://192.168.2.105:8080. Thus, listener waits for a request on this port number. For each request a new callback function is called to initiate the video stream over TCP socket.
    a. When a TCP request is received, another procedure begins.

b. All common variables such as capture rate and maximum user number are fetched.

c. Stopwatch parameters are set to obtain statistics values.

d. A new listener is set for a new possible client over particular IP and port number.

e. User number is incremented.

f. If the maximum user number is not exceeded, stream starts with a while loop.

g. Sample image is converted to byte data.

h. Image data is sent to client with MJPEG encoding.

CHAPTER 6

THE APPLICATION

The applications were developed with C# programming language, Microsoft Visual Studio 2008, PHP, MySQL with the aid of MJPEG codec and Apache Web Server. All the files of desktop and web application can be downloaded directly from http://www.serkanozdemir.com/ms-thesis.zip.

6.1 Architecture and Requirements

Web server part of the application is based on PHP 5.3.13, Apache 2.2.22 and MySQL 5.5.24. All required modules above were installed with WampServer 2.2 as well as phpMyAdmin [58]. WampServer is a Windows based development environment that allows creating applications with PHP, MySQL and Apache. As seen in Figure 27 below, WampServer provides a system tray toolbar that allows configuring PHP, Apache and MySQL in an easy way.

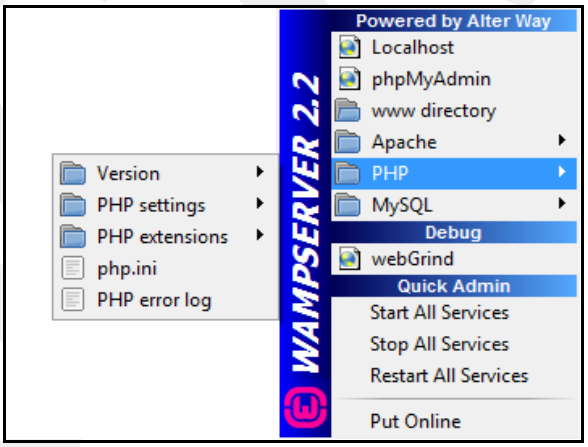

Figure 27. WampServer Panel in System Tray

In default mode, PHP uses output buffering. This feature prevents video streaming since the video is not able to be transmitted instantly. For this reason, output buffering feature must be disabled and implicit flush feature must be turned on by editing php.ini configuration file. On the other hand, “memory_limit” is set to 128 MB to be able to deal with various amounts of video frames on the fly without any corruption.

Desktop part of the application is responsible for video sampling and TCP based live video streaming. Application was developed on “Microsoft Visual Studio 2008 Version 9.0.21022.8 RTM” with “Microsoft .NET Framework Version 3.5 SP1”.

MJPEG was chosen as video delivery codec due to easy implementation and built-in support by major Internet browsers such as Firefox and Google Chrome.

All test cases were executed with Google Chrome since it has native support for MJPEG video codec. In other words, when using Google Chrome, an additional codec installation is not required.

Apache needs a module called “rewrite_module” that is essential for URL rewriting. For example, clients request stream with “127.0.0.1/stream.mjpeg” which is actually “127.0.0.1/stream.php”. This module is not a must. But it provides a better and user friendly interface for the clients.

6.2 Motion JPEG

Motion JPEG (MJPEG) is a video codec which compresses each video frame with JPG compression algorithm. So, each frame is sent one by one as JPG image individually. MJPEG is being used widely by digital cameras, IP cameras and non-linear video systems since the implementation is easy.

MJPEG is supported natively by major Internet browsers such as Safari, Firefox and Google Chrome. For this reason, MJPEG is suitable for a proper test among different approaches. MJPEG was chosen as a video delivery codec for the following reasons:

1. It has many built-in libraries on multiple platforms such as Microsoft Visual Studio and PHP. These platforms allow programmers to deal with JPG easily. For this reason, implementation is easier compared with modern codecs such as H.264/MPEG-4 AVC.
2. If the video content changes rapidly, clients can meet with quality problems due to high compression rate. Transition between frames can cause a significant quality loss. MJPEG is able to overcome this issue since it delivers the frames individually to clients.
3. Due to light structure and maturity of MJPEG, it has a common support by many platforms. For example, most Internet browsers support the MJPEG as a native delivery codec.

Although it has many advantages, it has disadvantages too. At this point, the most important topic is efficiency. MJPEG sends video frames individually without any advanced prediction algorithms. Modern competitors of MJPEG are extremely skilled on bandwidth problem since they have a prediction algorithm and better compression rate.

6.3 MySQL Configuration and Database Structure

This study offers a new live video streaming approach using relational database. For this reason, MySQL part is a basic but a simpler part of the study.

All tables of the database use MyISAM as a storage engine since it is more powerful when the amount of records is not decent. MyISAM keeps the records in order as they come. For this reason, fetching data is faster than InnoDB [59].

On the other hand, database must be still readable while other process is trying to insert new row. MySQL defines the ability of MyISAM in its official site.

> *"MyISAM supports concurrent inserts: If a table has no free blocks in the middle of the data file, you can INSERT new rows into it at the same time that other threads are reading from the table. A free block can occur as a result of deleting rows or an update of a dynamic length row with more data than its current contents. When all free blocks are used up (filled in), future inserts become concurrent again."* [59]

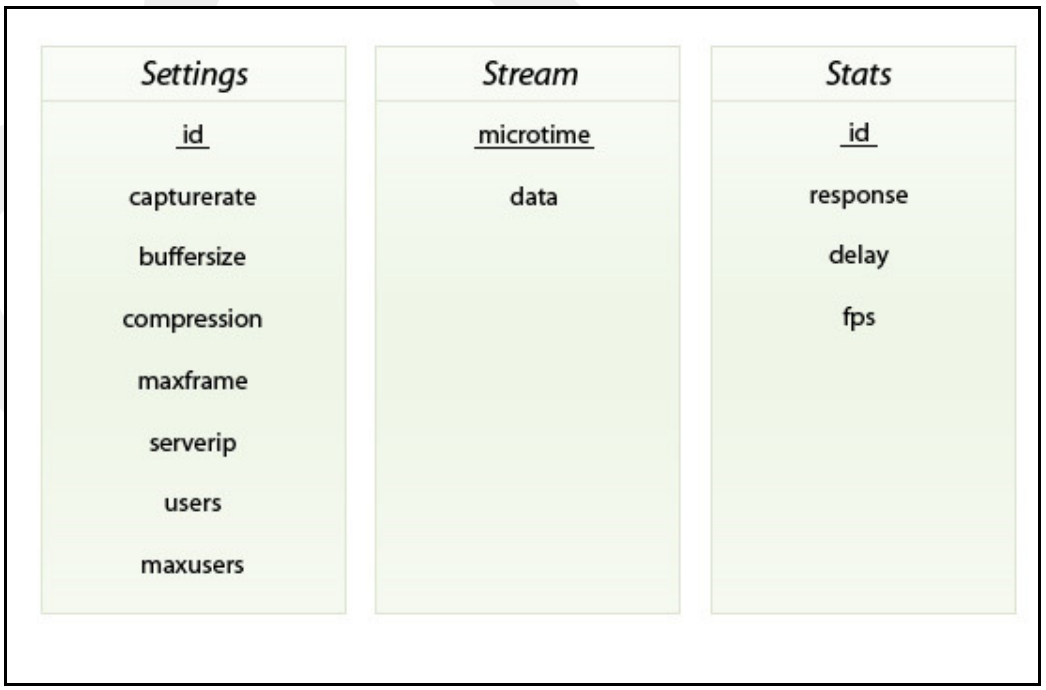

Figure 28. Tables of Database "stream"

The name of the application's database is "stream" and it has 3 tables. Figure 28 shows the tables and fields of them. "Settings" table is used to store basic settings

prior to beginning of the live video streaming. Video sampler gets all values from its GUI and stores it in database for the use of web server. Here are the definitions of fields:

1. id (int): Primary key of the table. Default record id is 1 which keeps the current configuration.
2. capturerate (int): It keeps the value of capture rate in terms of milliseconds between two video frames.
3. buffersize (int): Buffer size stores the amount of rows that hold the video frames.
4. compression (int): It is the JPEG compression rate out of 100. Higher value means less compression.
5. maxframe (int): This value is the maximum amount of lost frames at once. If the frame loss does not occur at once, it does not trigger the video streaming system.
6. serverip (varchar): Public IP address of the Internet gateway. It provides easier navigation on client side.
7. users (int): It holds the current number of connected and active users.
8. maxusers (int): Maximum allowed users at the same time.

“Stats” table is used for statistics, only to provide a proper test environment. Stats table is updated during the streaming by one of clients. In other words, only one client updates this table due to performance problems. Here are the fields of table:

1. id (int): Primary key. Default value is 1 similar to “Settings” table.
2. response (int): Keeps the response time in terms of milliseconds.
3. delay (int): Average delay in terms of milliseconds.
4. fps (int): Average frame per second.

“Stream” is the backbone of the video streaming design. But its structure is quite simple. It has two fields:

1. microtime (big int 16): It stores the exact time of frame inserted to row. This field is not a primary key but used as a key. For example, “4257304” is extracted as follows:

a. Minute: 42

b. Second: 57

c. Millisecond: 304

2. data (text): This field stores the decoded and compressed JPG image.

6.4 Desktop Application

Desktop application is implemented with C# programming language to deal with camera source. In other words, it grabs video input and stores the samples in the database. Beside this, it also includes a TCP socket based streaming solution to make a proper test.

However, Microsoft Visual Studio 2008 is not sufficient with its built-in functions. For this reason, some additional references and libraries are required. These are MySQL Connector and Touchless SDK.

6.4.1 MySQL connector

Microsoft Visual Studio 2008 does not have a built-in support for MySQL database management system. MySQL offers a connector for ADO.NET to fix this problem [60]. Latest available version “Connector/Net 6.7.2” was installed to server.

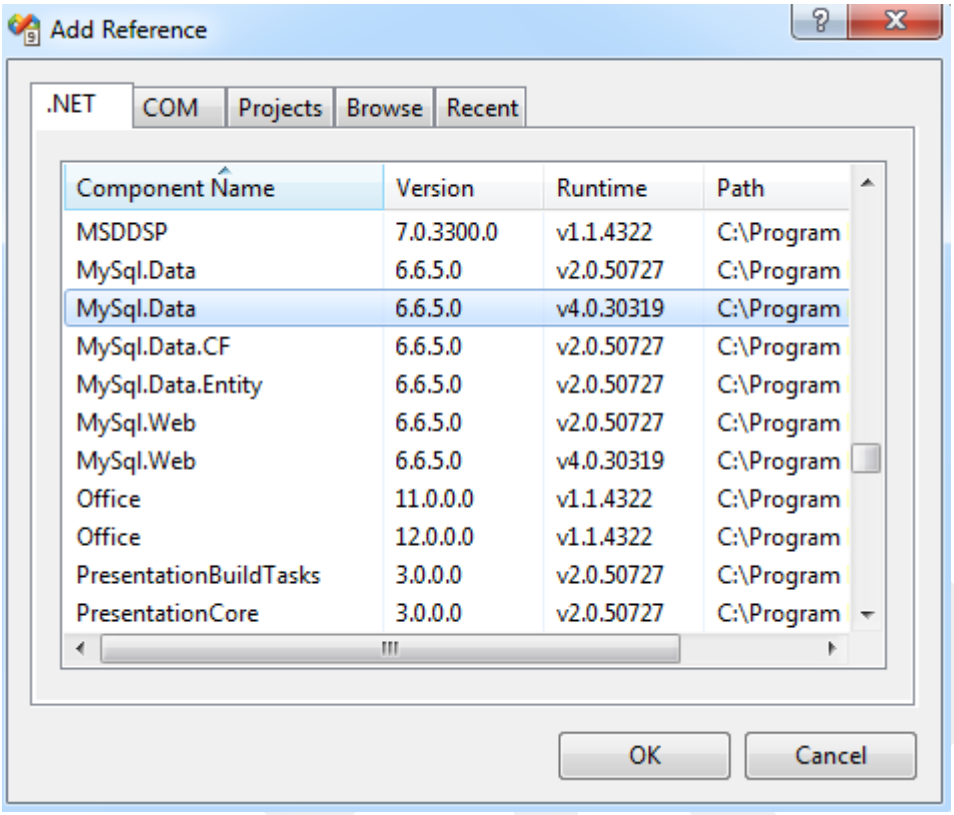

Figure 29. Installation of MySQL Component

As seen on the Figure 29, “MySql.Data” component must be added to application via solution explorer.

6.4.2 Touchless SDK

Touchless SDK is a project to deal with web cameras in an easy way [61]. SDK is open source and can be downloaded from its site for free. Touchless SDK uses DirectShow to query web camera and return the results back to caller function. Touchless SDK also offers library for C# programming language. Therefore, it makes easier to query web cameras.

6.4.3 Components

Basically, desktop application can be considered as three sections:

1. Common Section: It consists of GUI, constants and program manifest. These components work for both live video sampler and TCP socket based video stream.

2. Video Sampler: This part is responsible for video sampling. It deals with video source and generates samples within a particular interval. Then samples are compressed and delivered to database.

3. TCP Socket Solution: TCP based solution is implemented for test purposes. Simply, it listens to a given port and IP address. If a request is made, it initiates the video stream.

6.4.4 Common section

Common section includes many features used by both video sampler and TCP based solutions.

6.4.4.1 Software manifest

Software manifest includes meta information for the software itself. For example, software version, checking new versions and running as administrator features can be set through application manifest [62]. Manifest file must be in XML format. Thus, only allowed tags must be used to describe settings.

In our case, we need to run software as administrator. To achieve this, a simple code snippet is added to manifest. As seen in Figure 30, administration right is obtained via a statement in “requestedPrivileges”.

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1" xmlns:as
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
</asmv1:assembly>
```

Figure 30. Administration Privilege was Granted in Manifest File

6.4.4.2 Libraries

Application was built with additional libraries even though C# programming language has a built-in support. C# is extremely complicated for dealing with web camera and sampling it. For this reason, we preferred a more programmer friendly solution with Touchless SDK.

Touchless SDK offers an agent library to communicate with camera. This library is called as “WebCamLib”. It comes with a “cpp” and header file. When the streaming system needs to interact with camera, functions of this agent library must be used. Figure 31 includes the references and external libraries as well as application files:

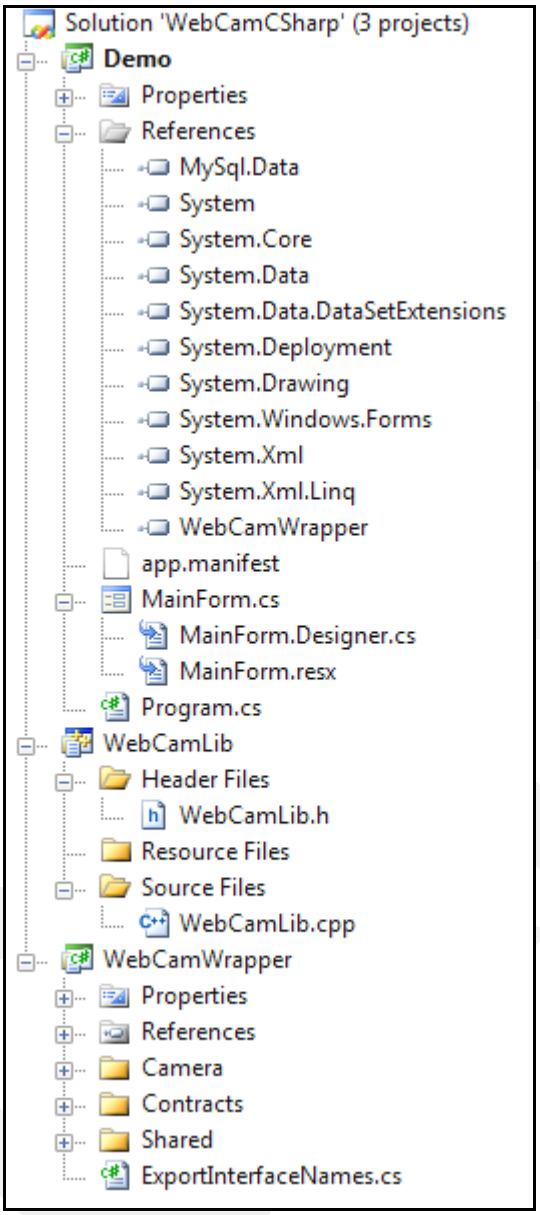

Figure 31. Solution Tree of the Desktop Application

"WebCamLib" offers following functionalities:

1. Initiates a callback mechanism with DirectShow.
2. Grabs all the information about cameras installed to computer.
3. Obtains data about a particular camera selected before.
4. Runs or stops a camera.
5. Displays the properties box of selected camera.

6.4.4.3 Constants

Constant values are required to access some common variables from anywhere. Due to the scope problem in object oriented programming, a variable can be accessed within its scope [63].

Software starts with constant values. As described before, particular variables such as current user number must be defined as a constant to access anywhere of the software. Moreover, these variables are updated during video streaming and stored in database for the use of web server.

```
public class stream
{
    public static int stop = 0;
    public static int comlev = 10;
    public static int videowidth = 320;
    public static int videoheight = 240;
    public static int maxusers = 20;
    public static int usernum = 1;
    public static byte[] pic = null;
    public static ImageCodecInfo jpegCodec = GetEncoderInfo("image/jpeg");
    public static ImageCodecInfo GetEncoderInfo(string mimeType)...
}
```

Figure 32. Constants of Desktop Application

The class “stream”, which is shown in Figure 32, is defined in “Program.cs” and accessible anywhere in the software. Here are the brief definitions of constants:

1. stop: In GUI of software, a button triggers the stop event which stops the sampling. Simply, “Stop” button changes “stop” variable to 1.
2. comlev: It refers to compression level. Default value is set to 10 out of 100 in terms of JPG quality.
3. videowidth and videoheight: They define the width and height of video output.
4. maxusers: It keeps the value of maximum allowed users connected to video streaming system.
5. usernum: Current user number. If a new request occurs, it is incremented.
6. pic: It holds the image data in byte format. Since the software is multithreaded, sampled image must be accessible anywhere in the software.
7. jpegCodec: It includes the codec information of JPG. It is required for JPG compression.

6.4.4.4 Namespaces

Namespaces are used in C# programming language to make programming easier for programmers. Most used functions are declared with a directive on the top of file [64].

Following namespaces are used in desktop application:

```
using System;
using System.IO;
using System.Net.Sockets;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using System.Drawing.Imaging;
using Touchless.Vision.Camera;
using System.Diagnostics;
```

Figure 33. Namespaces of the Desktop Application

As seen in Figure 33, some of the directives are essential for C# programming language. Following directives are specific to this application:

1. System.Net.Sockets: This namespace allows developers to manage Windows Sockets [65]. So, it is an essential namespace to deal with TCP sockets and TCP based streaming.

2. System.Threading: It provides classes for multi-threaded programming as well as data access classes such as Interlocked and Mutex. Multi-threaded is required to put “sleep” gaps between video frames. In single thread mode, sleep commands could be used, but it leads the freeze of GUI. For this reason, each stream must be forwarded to a new thread [66].

3. Mysql.Data.MySqlClient: This namespace must be installed separately to the project. Because it is not a built-in namespace in Microsoft Visual Studio 2008. Simply, it provides connection to MySQL server and query within Mysql.Data classes.

4. Touchless.Vision.Camera: This is an open source library that is outsourced from Touchless SDK. It provides ready to go classes to play with camera easily. This library is based on DirectShow [61].

5. System.Diagnostics: It provides classes to implement performance counters. These classes are used in test cases to measure metrics such as average delay and response time.

6.4.4.5 Camera settings and initialization

Camera settings and initialization is a totally common phase of the desktop application. Because, both our approach and TCP socket based streaming use the same video source.

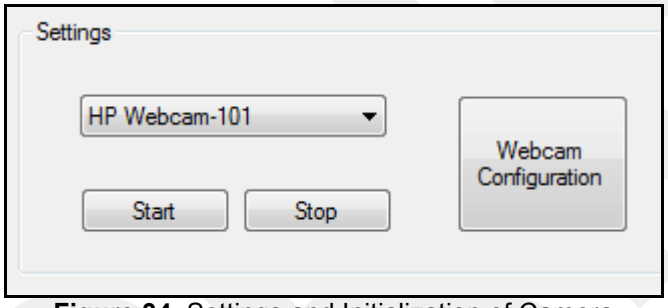

Figure 34. Settings and Initialization of Camera

Once the GUI form is loaded, cameras on the operating system are enumerated with Touchless SDK. Figure 34 show the user interface part of camera settings and initialization.

```
comboBoxCameras.Items.Clear();
foreach (Camera cam in CameraService.AvailableCameras)
    comboBoxCameras.Items.Add(cam);

if (comboBoxCameras.Items.Count > 0)
    comboBoxCameras.SelectedIndex = 0;
```

Figure 35. Querying Available Cameras

As seen in Figure 35, available cameras are inserted to a combo box via “foreach” function. Available cameras are obtained from operating system. For this reason, camera must be installed properly to the operating system in order to operate Touchless SDK and streaming system.

Initialization starts with “Start” button. “Onclick” event leads to “thrashOldCamera” and “startCapturing” functions respectively. “thrashOldCamera” function trashes the old camera resources and dispose all camera links that were created before.

Once the “thrashOldCamera” is completed, “startCapturing” function begins for initialization. Simply, it gets various parameters to initiate camera such as capture width and height.

```
private void startCapturing()
{
    try
    {
        Camera c = (Camera)comboBoxCameras.SelectedItem;
        setFrameSource(new CameraFrameSource(c));
        _frameSource.Camera.CaptureWidth = 320;
        _frameSource.Camera.CaptureHeight = 240;
        _frameSource.Camera.Fps = 30;
        _frameSource.NewFrame += OnImageCaptured;
        pictureBoxDisplay.Paint += new PaintEventHandler(drawLatestImage);
        _frameSource.StartFrameCapture();
    }
    catch (Exception ex)
    {
        comboBoxCameras.Text = "Select A Camera";
        MessageBox.Show(ex.Message);
    }
}
```

Figure 36. “startCapturing” Function for Initialization

To be more specific on the procedure in Figure 36, capture width and height are given with integer values. Note that capture width and height is completely different than video output. Nevertheless, output and input resolutions should be the same to avoid video quality problems. Frame per second value for capturing is given as 30. But this assignment does not mean that actual FPS will be 30. Probably, camera produces less than 30 frames in a second due to poor light and hardware conditions.

After assignments of constants, an event handler is defined to print captured image to GUI immediately. If all the processes above fail, then an exception will be thrown with “Select A Camera” message. “Stop” button initiates the release of video camera. Simply, it calls the “thrashOldCamera” function and terminates the video stream if exists.

Another key function is “drawLatestImage”. This function deals with the video frames on the fly. It captures the latest frame from video source and writes it to a variable on the memory.

```
private void drawLatestImage(object sender, PaintEventArgs e)
{
    if (_latestFrame != null)
    {
        // Draw the latest image from the active camera
        e.Graphics.DrawImage(_latestFrame, 0, 0,
            _latestFrame.Width, _latestFrame.Height);
        MemoryStream fs = new MemoryStream();
        SaveJpeg(fs, _latestFrame, stream.comlev,
            stream.jpegCodec);
        stream.pic = fs.ToArray();
        fs.Close();
    }
}
```

Figure 37. "drawLatestImage" Function

Figure 37 shows that "_latestFrame" is assigned to "stream.pic" via "MemoryStream". Obviously, "_latestFrame" includes the latest frame of video source that was grabbed via Touchless SDK. For this reason, we are not interested deeply on how the frames are grabbed from video source.

"MemoryStream" class is defined in "System.IO" and it creates streams in memory instead of disk. It keeps the data as "unsigned byte array" [67]. For this reason, variable of "fs" is converted to byte array in order to keep in "MemoryStream" format.

"SaveJpeg" function is responsible for the conversion of frame to JPG and keeping the data in memory stream. In this function, "stream.comlev" and "stream.jpegCodec" constants are used. As described before, constant of "stream.comlev" includes the compression rate of JPEG within a scale of 0 and 100. Similarly, "stream.jpegCodec" keeps the codec information of JPEG to make a proper compression.

Therefore, the frame is processed to be ready to deliver. "stream.pic" is a global variable that holds the latest frame. In both TCP socket based streaming and DB based streaming, "stream.pic" is the single reference for the video source. In other words, streamer part of the software only deals with "stream.pic". Otherwise, especially for TCP based streaming, all active clients had to receive the frame directly from video source. This causes to performance loss as well as implementation problems. On the other hand, "DrawImage" does not allow to multiple access at the same time. So, that is why "stream.pic" is essential for the video streaming.

6.4.4.6 Stream settings

Stream starts with some essential parameters. Simply, these parameters can be determined via GUI of the software.

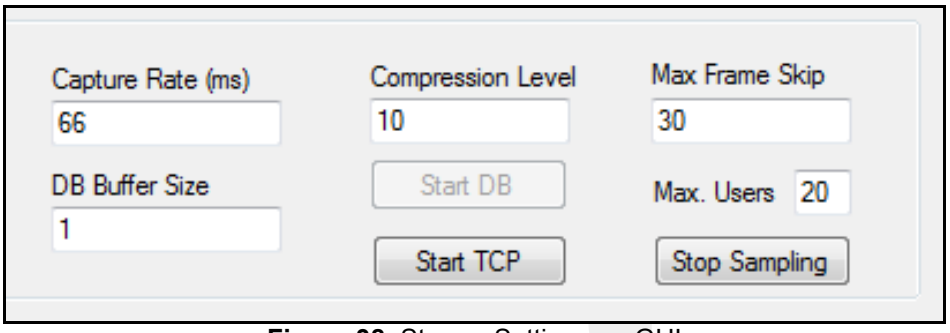

Figure 38. Stream Settings on GUI

As seen in Figure 38, there are five edit boxes and three buttons on the form. As described before, capture rate, compression level, frame skip rate, DB buffer size and maximum users are determined via edit boxes. These parameters are usable for both DB oriented streaming and TCP socket based streaming.

On the other hand, there are three buttons on the form which are going to be described later. Briefly, “Start DB” initiates the video stream based on relational database and “Start TCP” initiates the TCP socket based video streaming.

“Stop Sampling” button terminates the video stream regardless of the type of video streaming.

6.4.5 Video sampler

Video sampler is the starting point of live video streaming via relational database. Before starting to video sampling, we have to make sure that camera is initiated and web, database server must be running. If these conditions are met, then video sampler is ready to go. Video sampler starts with “Start DB” button in GUI. Initially, constant parameters are fetched from GUI. These parameters are listed below:

1. Capture rate.
2. DB buffer size.
3. Compression rate.
4. Maximum loss frame.
5. Maximum users.

As described before, these constants are common variables and they are also used in web server to manage client side tweaks and test cases. Once the parameters are stored to related variables, public IP address is determined by scrapping the data on dyndns.org [49]. To do this, “WebRequest” and “WebResponse” classes, which are defined in “System.Net”, are used.

Finally, a thread is created for the rest of the work. Since the sampling process needs sleep actions and GUI is frozen due to sleep actions, threading is a must implementation for the desktop application. Briefly, threading supports concurrent execution in C# programming language. In other words, threads do not affect each other unless they do not try to access to same resource [68]. For this reason, threading fixes the problem generated by sleep actions.

```
Thread MyThread = new Thread(new ThreadStart(fonksiyonumuz));
MyThread.Start();
```

Figure 39. Creation of Thread

The code in Figure 39 shows a typical creation of a thread. “ThreadStart” function takes another function named “functiondb” as a parameter. This means that thread is going to continue its life with “functiondb” function. In other words, “functiondb” function is going to handle the rest of video sampling job.

Meanwhile, all the parameters, which are gathered at the top of the function, are stored to database. MySql provides many classes to manage a database. In our case, following classes are used [69];

1. MySqlConnectionStringBuilder: It creates connection strings.
2. MySqlConnection: It links MySQL connection to a valid database.
3. MySqlCommand: It creates MySQL command string.

```
MySqlConnectionStringBuilder bag =
    new MySqlConnectionStringBuilder();
bag.Server = "localhost";
bag.UserID = "root";
bag.Password = "3292115";
bag.Database = "stream";
MySqlConnection baglanti = new MySqlConnection(bag.ToString());
baglanti.Open();
```

Figure 40. MySQL Database Connection

As seen on the code in Figure 40, a connection to a relational database is established via “MySql.Data” classes.

6.4.5.1 Function of “functiondb”

After the successful creation of thread, function of “functiondb” will be executed in the thread. First job of the function is to check stream tables and delete recorded frames if exist. To do this, a MySQL command is adequate. Figure 41 includes the commands of MySQL delete:

```
MySqlCommand cmd;
cmd = baglanti.CreateCommand();
cmd.CommandText = "delete from stream where 1";
cmd.ExecuteNonQuery();
```

Figure 41. MySQL Command to Delete Frames

While loop starts after the removal of old frames. Loop progresses while the “stream.stop” is not 1. In other words, if the stop button is not clicked, stream keeps running since the stop button triggers the assignment of “stream.stop” to 1. Figure 42 shows the flow chart of loop section:

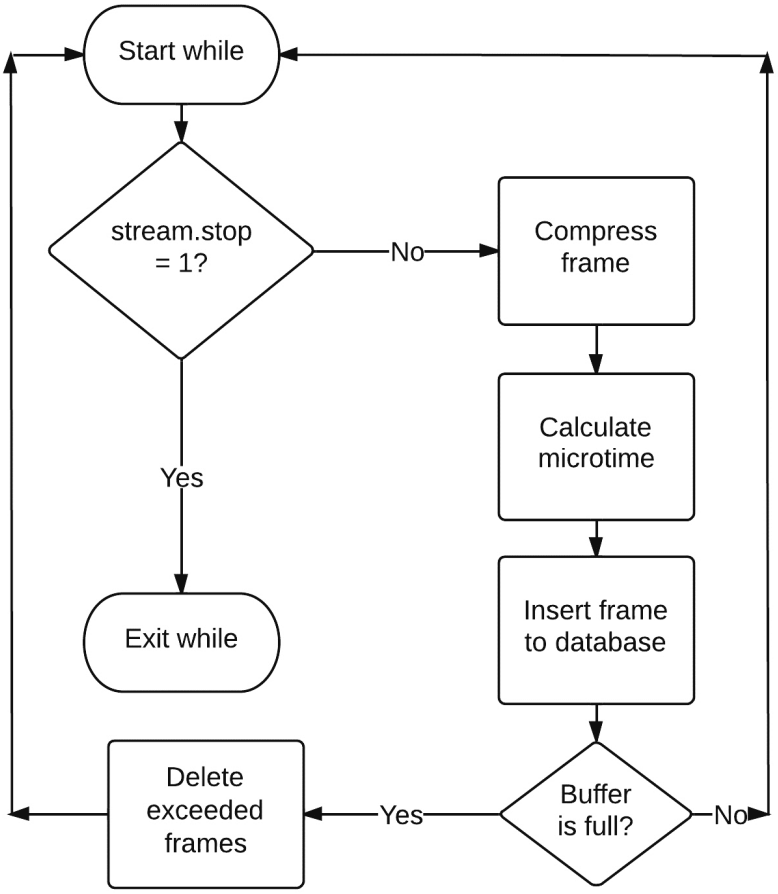

Figure 42. Flowchart of Video Sampler Loop

Sampler must use sleep actions between frames. For example, if the FPS rate is determined as 15 FPS, then sleeping duration must be 66 milliseconds in order to

achieve this rate. Simply, thread is sent to sleep with a threading class with the aid of capture rate which is obtained from GUI.

Next step is submitting of data and “microtime” to be sent to database. As described before, “stream.pic” includes the latest frame that is grabbed from video camera. For this reason, it is used for the latest video reference.

“stream.pic” holds the data as byte arrays. Since the text fields in MySQL are not able to store byte arrays, it is decoded by Base64. One more advantage of Base64 is wide range of support by multiple platforms. For example, both Microsoft Visual Studio 2008 and PHP have built-in support for Base64 [70].

```
String s = Convert.ToBase64String(stream.pic);
int datee = Convert.ToInt32(DateTime.Now.ToString("mmssfff"));
cmd.CommandText = "insert into stream values(" + datee + ",'" + s + "')";
cmd.ExecuteNonQuery();
```

Figure 43. Delivery of Frames to Database

Micro time is directly fetched via built-in date function. As seen in the Figure 43 above, micro time value comes with a string input: “mmssfff”. “mm” refers to current minute as two digits, “ss” refers to current second as two digits and “fff” refers to current milliseconds as three digits.

Once the data and date are ready to go, they are submitted to database with a MySQL query. Meanwhile, preview frame in GUI is updated with the latest frame. After that, buffer size is checked in database if any overflow is exists. If there are more frames then buffer size, first inserted frame will be deleted from buffer.

6.4.6 TCP socket solution

TCP based solution is implemented to compare two unicast approaches. For this reason, same procedures and features are used. For example, MJPEG was chosen as delivery codec to make a proper comparison between TCP socket based streaming and relational DB based streaming.

Similar to database approach, this implementation gets the parameters like capture rate and compression rate. All setting parameters are gathered via GUI. Initialization of TCP based solution starts with gathering parameters. Finally these parameters are stored in database and a thread is created for the rest of the job.

New thread takes the name of “functiontcp” function as a parameter. In other words, new thread is going to start with the process of “functiontcp”.

6.4.6.1 Function of “functiontcp”

Function only sets a listener for port number 8080 using local IP that uses “HttpListener”. Simply, “HttpListener” listens to the port number for a possible request. If a new request comes, it is forwarded to callback function [71].

```
public void functiontcp()
{
    HttpListener listener = new HttpListener();
    listener.Prefixes.Add("http://192.168.2.105:8080/");
    listener.Start();
    listener.BeginGetContext
        (new AsyncCallback(OnRequestReceive), listener);

}
```

Figure 44. Function of “functiontcp”

Figure 44 above shows the code of “functiontcp”. It listens to 8080 and forwards new clients to “OnRequestReceive” callback function. On the other hand, “HttpListener” is able to host only one client. If another client makes a request, delivery is going to fail. To deal with this problem, a new “HttpListener” code is placed to “OnRequestReceive”. As a result, for each new client, “OnRequestReceive” is going to be called in a recursive manner.

6.4.6.2 Function of “OnRequestReceive”

Unlike database approach, this function is fully responsible for the delivery of stream and test operations. So, it has more codes inside compared with function of “functiondb”. Initially, current user number is incremented. This means that, for each client, a global variable of “stream.usernum” is incremented to be able to track the user activity.

Common parameters are gathered from GUI and variables are defined at the top of the function. For the test cases, a “Stopwatch” variable is defined. “Stopwatch” class is defined in “System.Diagnostics”. As described before, “Stopwatch” is essential to measure processing metrics such as response time and delay.

Meanwhile, another “HttpListener” is set for a new client. If a new request arrives, this function is called one more time. Just before streaming, headers and separators are sent to client.

```
ctx.Response.StatusCode = 200;
ctx.Response.ContentType = "multipart/x-mixed-replace; boundary=--myboundary";
```

Figure 45. Headers of TCP Streaming

As seen in Figure 45, success code of 200 is sent to client as well as header information. In header data, boundary refers to delimiter that will be used between two frames. In this case, we must sent “--myboundary” message after each frame. Otherwise, stream would not be meaningful to client.

Similar to database approach, streaming is implemented within a while loop. Loop checks “stream.stop”. If it is 1, then loop is terminated. As implemented in database model, we must use sleep actions between frames to have proper FPS rate. For this reason, thread sleeps as capture rate.

```
byte[] imageData = stream.pic;
string header = "--myboundary\r\n" + "Content-Type:image/jpeg\r\n"
    + "Content-Length:" + imageData.Length.ToString() + "\r\n\r\n";
writer.Write(header);
writer.Write(imageData);
```

Figure 46. Delivery of Data in TCP Approach

Frame is directly sent to client as byte data where “--myboundary” is the delimiter of frames. Since the frame is sent as byte data, browsers can easily identify the JPEG media type. Figure 46 shows the delivery of frames to clients in TCP socket approach.

TCP socket based streaming is efficient if several clients are targeted. Otherwise, performance of streaming dramatically decreases due to lack of processing resources of server.

6.5 Web Server Application

In relational database model, clients make request over web server. For this reason and due to high processing ability, a PHP script welcomes the clients. This agent file in the web application is “stream.php”.

A PHP script must be modified carefully to be able to deliver a video stream. The basic modifications are listed below:

1. Memory limit: It must be at least 128 MB.
2. Error reporting must be turned off.
3. Header information must be proper; including no-cache, no-store.
4. GZIP and output compression must be turned off.

5. Implicit flush feature is essential for video streaming. Because, printed data should be sent to client immediately [72].

All modifications above can be implemented in PHP easily. As a result, “stream.php” file starts with the modifications above. All parameters sent by video sampler are gathered from database settings table such as capture rate, buffer size and compression rate.

Streaming is implemented within a while loop. At the beginning of the loop, a MySQL query searches for available frames on the database.

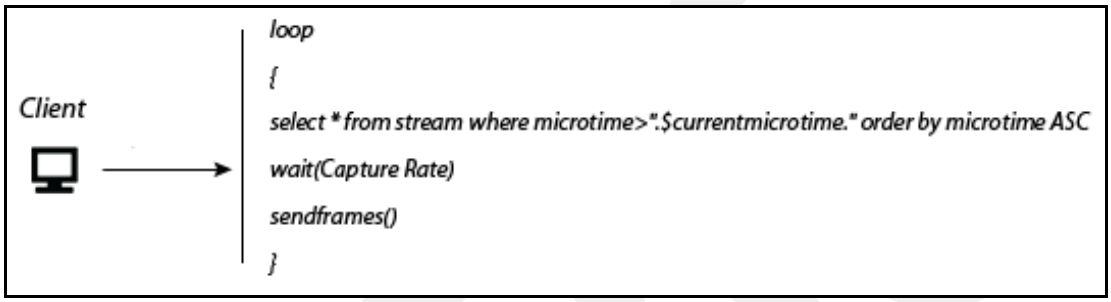

Figure 47. Streaming Implementation of “Stream.php”

As seen in Figure 47, loop checks the database continuously for available frames. This way is extremely efficient if the buffer size is over a hundred. Because, a single MySQL query can obtain a hundred frames at once. However, performance of delay and response time dramatically decrease.

```
$currentmicrotime = $r['microtime'];
print "--$boundary\n";
$predata = imagecreatefromstring(base64_decode($r['data']));
print "Content-type: image/jpeg\n\n";
imagejpeg($predata, NULL, 50);
usleep($capturerate*1000); // 50000 = 50 ms
print "--$boundary\n";
imagedestroy($predata);
```

Figure 48. Printing Frames in PHP

Micro time and frame data are extracted from database. Since the data was encoded in video sampler, this script must decode back. Figure 48 includes the PHP codes which print frames to clients. Unlike Microsoft Visual Studio 2008, PHP provides an efficient and easier sleep function with “usleep”.

6.6 Screenshots and Manual

Applications are designed as user friendly and capable to make multiple tests. Most common parameters are placed into GUI.

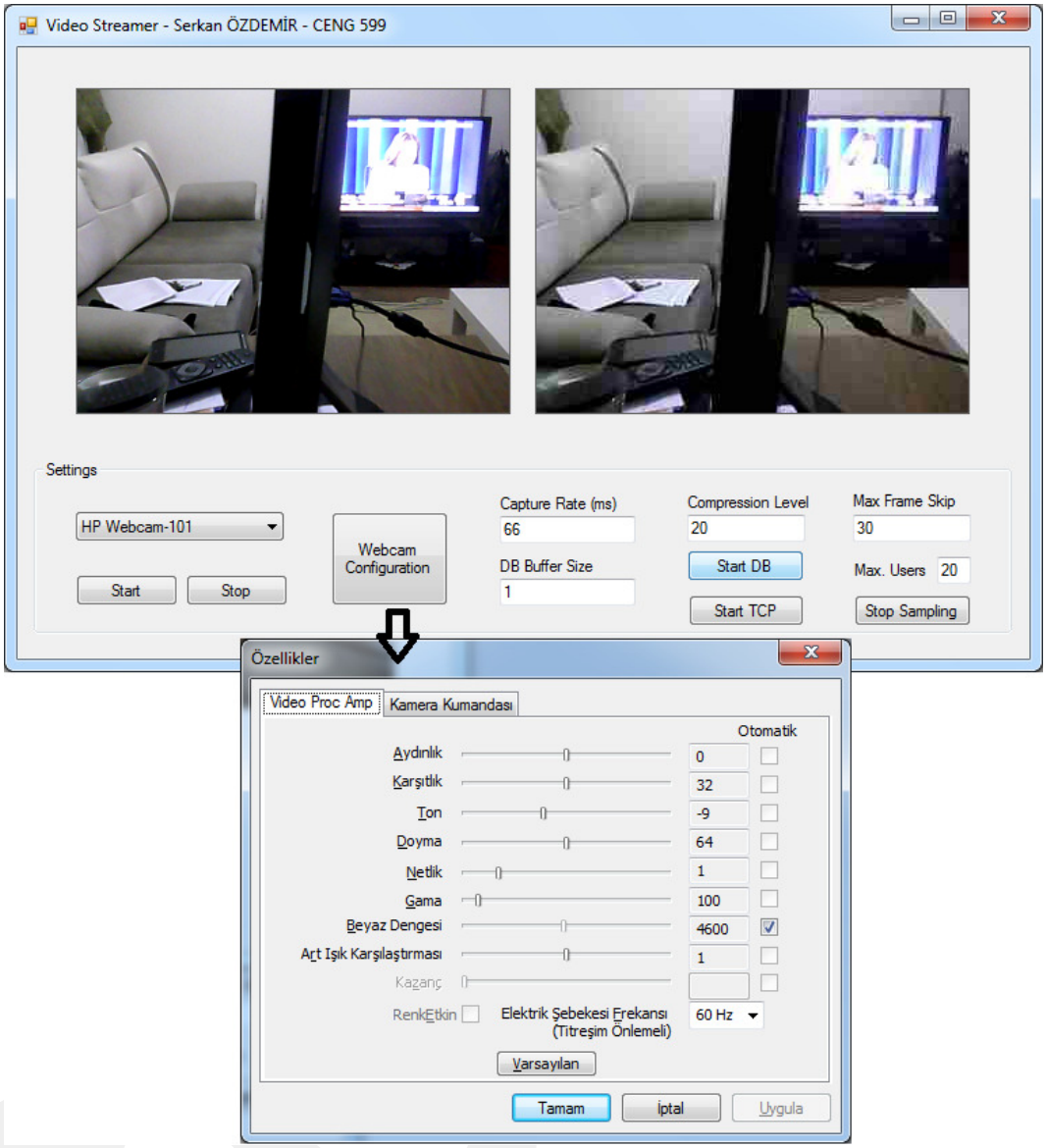

Figure 49. User Interface of Desktop Application

Application has two preview frames as seen in Figure 49. First frame on the left displays the video resource regardless of capture rate and compression level. Frame on the right displays the sampled video which is affected from capture rate and compression level. Webcam Configuration button opens a new window and allows making advanced settings of video camera.

Figure 50. User Interface of Web Application

Client can access to web interface at http://127.0.0.1 or via public IP address. In case of public IP usage, port forwarding may be required. Port 80 and 8080 must be forwarded to local ports.

Figure 50 shows the user interface of web application and it offers two preview pages. “Streaming via DB” includes streaming window with statistics which is proposed by this study. Other link, “Streaming via TCP”, provides a traditional approach based on TCP sockets.

Simply, “Start” link initiates the streaming process. Stream does not stop unless stop button of browser is clicked or browser is closed. Web interface provides settings information and statistics output as well as video streaming frames. Capture rate, buffer size, compression level and public IP address are displayed above the video frame.

Video stats are updated continuously and displayed under video frame using Ajax technology of PHP.

CHAPTER 7

TEST RESULTS

Testing phase took place within a single server. Server has Intel Core i5 2,67 GHz processor, 4 GB RAM and 32 bit Windows 7 operating system.

Streaming test cases were tested in local server. For this reason, time between departure and arrival to remote computer is ignored. In other words, test cases include the results of processing and efficiency performance of streaming approaches. Bandwidth of network is assumed as unlimited since the local server is used for test phase.

Each test case examined 5 times and average values were gathered to make final decision on values. Compression rate for test cases is 10. This value is out of 100 in terms of JPEG image quality. Frame size is 320x240 pixels. For this reason, required bandwidth for a single video stream session is around 35 KB/s. If 10 users are connected to server, then 350 KB/s required to deliver stream without any corruption.

Frame rate for test cases is 14 FPS and maximum lost frame rate is 30. Thus, a client is able to be connected to server even it skips 30 frames at once. In other words, video stream will be considered successful even it skips frames less than 30 at once. This feature fills the gap of client-server connectivity problem.

In client-server model, there are five different metrics: DB buffer size, users, response time, average delay and average frame rate. In TCP model, all metrics of client-server model, except DB buffer size, are used.

Stopwatches were used in C# programming language to calculate execution and delivery time. In PHP, it was easier to calculate the delivery time since the frames in database has a column includes micro time in format “mmssfff” where “mm” is minute, “ss” is second and “fff” is millisecond.

As seen on Table 1, Table 2, Table 3 and Table 4, some test results were marked as “n/a” which is expanded as “not available”. Briefly, if the application did not respond or client could not get more than five frames per second, we marked the

case as “n/a”. The main reason for the performance loss is the lack of either processing power or bandwidth. For example, this term was used for the insufficient processing power for the client-server streaming model. Since the client-server approach has more overheads due to traverse of frames between services, CPU is extremely busy to process frames. When we watch the CPU load and memory usage during the examination of test cases, we can observe the high load of CPU even though the memory usage is low.

On the other hand, the term of “n/a” refers to processing power and transport layer insufficiency in TCP socket based streaming. In TCP model, CPU is busy to duplicate the video source for each new client. Additionally, TCP model sends the stream over the same TCP socket. For this reason, it gains high load at transport layer. As a result, TCP model requires more processing power within a low internal bandwidth. In other words, stream traverses between video source and transport layer. This situation is better for a single client. However, the solution is not suitable if you serve to multiple clients due to duplication of the stream.

7.1 Response Time

Response time is the duration between first creation of stream and the arrival of first frame to client. Response time is a similar metric to delay. However it consists of additional initialization time. Figure 51 illustrates the calculation of response time:

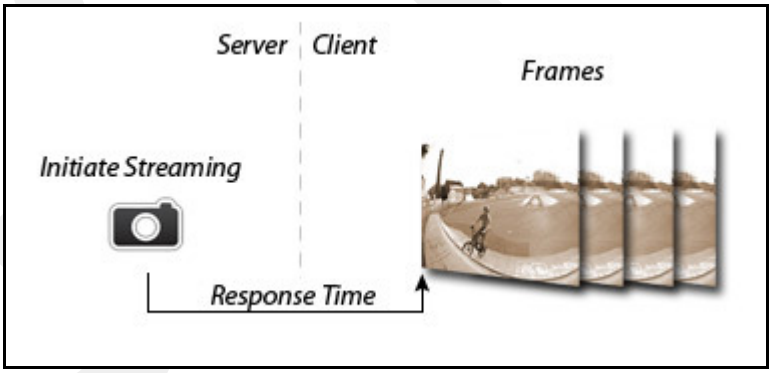

Figure 51. Response Time

7.2 Average Delay

Delay is a similar metric to response time. But it does not include the initialization time. For this reason, it is expected that delay must be shorter than response time for a single client. If the number of client increases, delay increases too.

Delay is obtained via micro time variable. Since the frames in database come with micro time value which is the creation time, time interval is calculated at the arrival

of frame and added to sum of delay. Sum of delay is divided by the number of total frame and average delay is obtained. As seen in Figure 52, average delay is derived from the delay values and frame numbers.

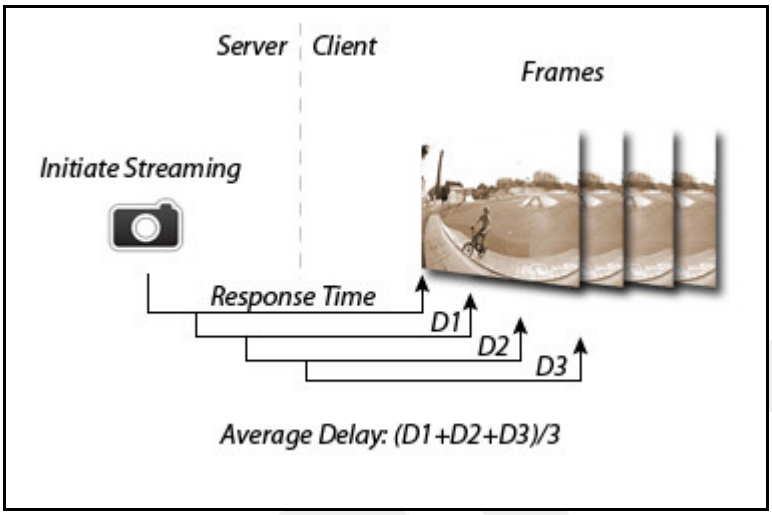

Figure 52. Average Delay Metric

7.3 Average Frame Rate

As seen in Figure 53, frames per second (FPS) is determined by counting the frames for each second. Then average frame rate is obtained by overall frame rate of each second. For example, if 100 frames are received within 5 seconds, then average frame rate would be 20 FPS. On the other hand, frames that are not received are not counted, and average frame rate only consists of received frames.

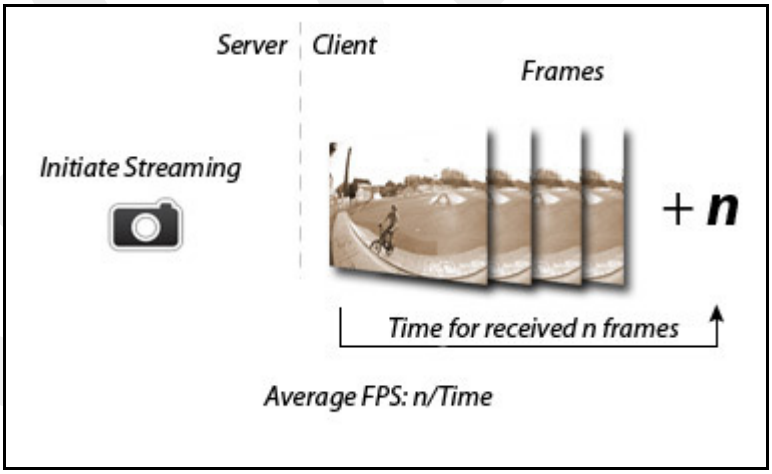

Figure 53. Determination of “Average FPS”

7.4 Buffer

Buffer is used for database model to keep system more stable. If the number of clients increases, more buffer rows are required to provide a seamless video

stream. On the other hand, buffer leads to delay increases due to temporary hold of frames.

Each element of buffer is a row of database table. In other words, table rows are used as LIFO queue. Rows are fetched from database via SQL commands with the aid of micro time. Rows are ordered by the time they are inserted.

7.5 Test Results of Database Model

Here are the test results based on buffer size. “n/a” shows that stream was unsuccessful due to limited resources of server. In other words, unsuccessful metrics in test results are limited with the processing capacity of the computer. For example, maximum users in buffer size 1 may be 20 instead of 10 within a more powerful computer.

Table 1. Test Results where DB Buffer Size is 1

Users	Response Time	Average Delay	Average FPS
1	120 ms	115 ms	14 FPS
5	145 ms	120 ms	13 FPS
10	n/a	n/a	n/a

DB buffer size has a key role if the stream will be served to more clients. As seen in Table 1, stream is corrupted once the user number is 10 or more. In this case, system deals with overheads more than actual data delivery since the buffer size is 1. Web server and database server are extremely busy in order to keep the buffer updated and deliver frames in buffer to multiple clients.

As a result, buffer size 1 will be suitable if the number of clients is around 5. In this case, clients receive the live video stream with a better response time and minimum average delay.

Table 2. Test Results where DB Buffer Size is 5

Users	Response Time	Average Delay	Average FPS
1	357 ms	430 ms	14 FPS
5	378 ms	432 ms	13 FPS
10	401 ms	435 ms	13 FPS
20	447 ms	550 ms	9 FPS
50	n/a	n/a	n/a

If the “DB Buffer” is set to 5, then live video streaming is able to reach more clients at the same time. In this mode, web and database server deal with less overhead compared with previous test. In other words, video streamer puts frames to database and clients receive the frames in bulk.

As seen in Table 2, stream begins to loose frames after reaching user number to 20. For user number 50, streaming does not work properly. In this case, maximum rate for loss frames is 30. Test shows that DB oriented streaming design is not able to server 30 frames at once.

Delay and response time increase proportional to DB buffer size. For example, 20 clients see the frames after 550 ms of creation at the same time. Compared with TCP based solution, delay rate is not competitive. This mode will be successful if the number of targeted clients is less than 20. Otherwise frames are not received by clients properly.

Table 3. Test Results where DB Buffer Size is 20

Users	Response Time	Average Delay	Average FPS
1	1435 ms	1515 ms	13 FPS
5	1440 ms	1530 ms	13 FPS
10	1442 ms	1545 ms	13 FPS
20	1642 ms	1782 ms	11 FPS
50	3796 ms	5457 ms	6 FPS

Test case above uses 20 as DB buffer size. As claimed in earlier sections of the thesis, more buffer size provides more users, but limited to processing resources of the server computer.

This mode uses overheads in minimum in order to serve video stream frames to maximum number of clients.

As seen in Table 3 above, this configuration is capable to host maximum 50 users at the same time due to lack of processing resources. Moreover, average FPS decreases to 6 FPS even though the frames are ready to go in database rows. For this reason, a more powerful computer can host more users with a better FPS value.

This mode is more suitable to gain maximum benefit from the server's processing resources.

Although it is able to host more clients at the same time, delay and response time increase extremely. For user number 50, delay is around 5 seconds. In other words, frames arrive to client after 5 seconds of creation.

7.6 Test Results for TCP Model

TCP test cases are handled within desktop application. During the delivery of video stream, statistics variables are also stored.

TCP model uses all the parameters of DB model except buffer size. As explained in previous pages, buffer is implemented on relational database to deliver video stream over buffer itself to clients.

Test cases are based on user number. For each user amount, response time, average delay and average FPS variables are determined.

Table 4. Test Results Based on TCP Sockets

Users	Response Time	Average Delay	Average FPS
1	72 ms	69 ms	14 FPS
5	78 ms	71 ms	13 FPS
10	80 ms	75 ms	13 FPS
15	85 ms	82 ms	7 FPS
20	n/a	n/a	n/a

Table 4 includes the test results of TCP model which are slightly different compared with DB model. TCP model is more successful on delay and response time values.

TCP model is not able to serve video stream to 20 users at the same time. TCP model was implemented in multithreaded mode and based on TCP socket listeners. For this reason, each additional user triggers the listener and creates a new thread. In this point of view, 20 users run 20 separate threads at the same time. Each thread makes a copy of video stream and delivers to the client. So, capacity of TCP model is limited by the processing capability of the server.

However, TCP model provides a better response time and average delay regardless of number of the users.

Database model uses redundant services such as web and database server. These two services will produce huge amount of overhead if the buffer size is closed to 1. On the other hand, TCP model directly deals with client and deliver the video stream to the clients without any agent services. For this reason, TCP model prevents the additional traverse of data within server.

Although TCP model has significant advantages on delay and response time, it is not able to gain all the processing resources of the server. TCP multiplies the video source and lock the transfer of video data on a narrow road.

CHAPTER 8

CONCLUSIONS AND DISCUSSIONS

This study offers an alternative and cheap solution since the live streaming is essential for some markets such as security, military and medicine.

In this study, a new unicast, live video streaming approach offered due to performance and data access problems in traditional unicast solutions. Therefore, a client-server architecture for live video streaming using object relational database was implemented.

A TCP socket based video streaming was also implemented to make proper and fair tests between these two approaches. As mentioned before, TCP socket based streaming needs more processing power if the number of clients is too large. Although it provides a direct path between video source and transport layer, each new client requires the duplication of the stream which makes the stream inefficient. Simply, we placed “n/a” value, which is “not available”, when we did not get respond or we received less than 5 frames per second. This way provides a better view for proper comparison between these two approaches.

Although this study had claimed a better bandwidth and response time, test cases showed more complicated results.

Conclusions of the study are listed below:

1. Bandwidth performance was dramatically decreased since the delivery codec was MJPEG. Although MJPEG uses a compression algorithm, it has many disadvantages compared with modern competitors such as MPEG-4. There are two main disadvantages on MJPEG. JPEG has a decent compression algorithm but it is not efficient. For example, H.264 video codec is more successful on compression [44]. The other disadvantage is related to video efficiency. New video encoders link frames to each other. In other words, current frame is not sent out completely. Only different part from the previous frame is delivered. In this way, new generation codecs use the bandwidth with more efficiency. But, if a client does not care about bandwidth and does

not have a tolerance to unstable, unacceptable quality loss, then MJPEG would be a better solution.

2. This study offers a solution with database and web server. For this reason, additional cost of installation and maintenance of these services are required.

3. Test results showed that TCP socket based solution performed a better response time and average delay. Response time values of TCP tests are around 80 milliseconds while our approach offers 115 milliseconds. Main reason of difference is related to overheads. In relational database model, frames moves between multiple services such as web server and database server. Thus, response time performance decreases compared with TCP model.

4. Although TCP model has advantages on response time, it failed when the user number was above 15 due to lack of processing power of server. DB model was able to host 50 users with buffer size 20. Briefly, DB model is able to access more people within the same processing environment.

5. DB model uses the abilities of web and database servers. To be more specific, Apache and MySQL servers are extremely specialized on using the all processing resources of computer. However, traditional unicast solutions gain only network layer and video source.

6. DB model provides multiple access to frames. Requested frames are obtained easily with a single SQL command. Since many programming platforms allow SQL querying, video frames can be accessed from any domain.

7. DB model provides a more secure delivery. Since the flow of data is fully under control, frames can be reformed or reprocessed to provide a secure delivery. It is also possible to deliver frames over secure HTTP.

8.1 Recommendations for Future Works

MJPEG is an old video delivery codec compared with modern siblings such as H.264. For a better bandwidth performance, H.264 can be used instead of MJPEG. However, H.264 requires additional programming skills on video processing. For this reason, converting a live video stream to H.264 stream may be a topic of future works. Bandwidth problems trigger many other problems such as quality loss, frame

loss. If the network type and number of clients are not controllable, MJPEG would not be a sufficient solution for delivery. For reference purposes, FFMPEG and similar providers offer solutions to convert one media format to another format [19]. But dealing with video structure requires an additional proficiency on video processing and programming. Another recommendation is to merge all services in one. Microsoft Visual Studio allows many built-in complicated operations. In this manner, web server and database server can be embedded into a single application. This way provides a more commercial and profitable product on the market.

REFERENCES

[1] **T. SILVA**, **J. M. ALMEIDA**, **D. GUEDES**, Live streaming of user generated videos: Workload characterization and content delivery architectures, 2011.

[2] **GELMAN**, **A. D. BELLCORE**, **MORRISTOWN**, On buffer requirements for store-and-forward video on demand service circuits, 1991.

[3] Wikipedia, Streaming Media article, retrieved January 21, 2013, from http://en.wikipedia.org/wiki/Streaming_media.

[4] **Y. SANCHEZ**, **T. SCHIERL**, **C. HELLGE**, **T. WIEGAND**, **D. HONG**, **D. DEVLEESCHAUWER**, **W. VANLEEKWIJCK**, **Y. LELOUEDEC**, Efficient HTTP-based streaming using Scalable Video Coding, 2011.

[5] **N. RAMZAN**, **H. PARK**, **E. IZQUIERDO**, Video streaming over P2P networks: Challenges and opportunities, 2012.

[6] Ustream, retrieved January 28, 2013, from http://www.ustream.tv.

[7] **CH. Z. PATRIKAKIS**, **N. PAPAOULAKIS**, **CH. STEFANOUDAKI**, **M. S. NUNES**, "Streaming content wars: Download and play strikes back" presented at the Personalization in Media Delivery Platforms Workshop, [218 – 226], Venice, Italy, 2009.

[8] Patent by **G. O. SQUIEX**, retrieved February 11, 2013, from http://www.google.com/patents?id=5pV5AAAAEBAJ&dq=1641608.

[9] Real Networks, retrieved February 11, 2013, from http://eu.real.com/.

[10] User Datagram Protocol, retrieved February 11, 2013, from http://tools.ietf.org/html/rfc768.

[11] **F. LEU**, A novel network mobility handoff scheme using SIP and SCTP for multimedia applications, 2009.

[12] **J. AVESTRO**, **R. FERIA**, Adaptive RTP-Compatible Audio Streaming for Handheld Clients (ARCASH), 2006.

[13] Real-time Transport Protocol, retrieved February 12, 2013, from http://tools.ietf.org/html/rfc3550.

[14] **A. BASSO**, **G. L. CASH**, **M.R. CIVANLAR**, Real-time MPEG-2 delivery based on RTP: Implementation issues, 1999.

[15] **H. SCHULZRINNE**, **S. CASNER**, **R. FREDERICK**, **V. JACOBSON**, RTP: a transport protocol for real-time applications, RFC 3550.

[16] **S. SHAHBAZI**, **K. JUMARI**, **M. ISMAIL**, A new design for improvement of scalable-RTCP, 2009.

[17] **Y. LIU**, The research of streaming media mutual digest authentication model based on RTSP protocol, 2008.

[18] Real Network, RDP, retrieved February 19, 2013, from http://service.real.com/help/library/guides/production/htmfiles/whatsnew.htm.

[19] FFMPEG, retrieved February 22, 2013, from http://ffmpeg.org/ffmpeg.html#rtsp.

[20] **T. DREIBHOLZ**, **E. RATHGEB**, Stream control transmission protocol: Past, current, and future standardization activities, 2011.

[21] Stream Control Transport Protocol, retrieved February 22, 2013, from http://tools.ietf.org/html/rfc3286.

[22] Transmission Control Protocol, retrieved March 11, 2013, from http://tools.ietf.org/html/rfc793.

[23] **E. SETTON**, **P. BACCICHET**, **B. GIROD**, Peer-to-Peer Live Multicast: A Video Perspective, 2008.

[24] **X. SHEN**, **H. YU**, **J. BUFORD**, **M. AKON**, Handbook of Peer-to-Peer Networking (1st edition). New York: Springer. p. 118, 2009.

[25] **B. YANG**, **H. MOLINA**, Designing a super-peer network, Proceedings of the 19th International Conference on Data Engineering, 2003.

[26] Wikipedia, Peer-to-peer article, retrieved March 12, 2013, from http://en.wikipedia.org/wiki/Peer-to-peer.

[27] **M. YEUNG**, **C. CHUNG**, **F. HARTANTO**, Peer-to-peer video distribution over the Internet, 2003.

[28] A Unicast address, retrieved March 12, 2013, from http://www.science.uva.nl/research/air/projects/old_projects/ipv6/IPv6_uni.htm.

[29] Differences Between Multicast and Unicast, retrieved March 15, 2013, from http://support.microsoft.com/kb/291786/en-us.

[30] **M. ARREGOCES**, **M. PORTOLANI**, Data Center Fundamentals, 2003, pp. 962-963.

[31] **G. LAWTON**, Multicasting: will it transform the Internet? 1998.

[32] IP Multicast Applications: Challenges and Solutions, retrieved March 15, 2013, from http://tools.ietf.org/html/rfc3170.

[33] **T. DUONG-BA**, **T. NGUYEN**, Distributed client-server assignment, 2012.

[34] **D. A. GRIER**, The Relational Database and the Concept of the Information System, 2012.

[35] **E. CODD**, A Relational Model of Data for Large Shared Data Banks, 1970, pp. 377-387.

[36] Oracle the clear leader in $24 billion RDBMS market, retrieved March 18, 2013, from http://itknowledgeexchange.techtarget.com/eye-on-oracle/oracle-the-clear-leader-in-24-billion-rdbms-market/.

[37] Market Share, retrieved March 18, 2013, from http://www.mysql.com/why-mysql/marketshare/.

[38] **B. LI**, **H. YIN**, Peer-to-peer live video streaming on the Internet: issues, existing approaches, and challenges, 2007.

[39] International Cablemakers Federation, How much Bandwidth? retrieved March 21, 2013, from http://www.icf.at/en/6000/how_much_bandwidth.html.

[40] MPEG4 Visual, retrieved April 1, 2013, from http://mpeg.chiariglione.org/standards/mpeg-4/video.

[41] **S. PYKKÖ**, **J. HAEKKINEN**, Evaluation of Subjective Video Quality of Mobile Devices, 2005.

[42] **D. VATOLIN**, MSU Subjective Comparison of Modern Video Codecs, retrieved April 1, 2013, from http://compression.ru/video/codec_comparison/subjective_codecs_comparison_en.html.

[43] **S. WENGER**, RTP Payload Format for H.264 Video, retrieved April 1, 2013, from http://tools.ietf.org/html/rfc3984#page-2.

[44] **G. SULLIVAN**, **P. TOPIWALA**, **A. LUTHRA**, The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions, 2003.

[45] ISO, ISO/IEC 14496-2:2004 - Information technology -- Coding of audio-visual objects -- Part 2: Visual.

[46] **M. NIEDERMAYER**, 15 reasons why MPEG4 sucks, retrieved April 5, 2013, from http://guru.multimedia.cx/15-reasons-why-mpeg4-sucks/.

[47] Helix Media Delivery Platform, retrieved April 10, 2013, from http://www.realnetworks.com/helix/streaming-media-products/.

[48] The Difference Between ASF and WMV/WMA Files, retrieved April 11, 2013, from http://support.microsoft.com/kb/284094/en-us.

[49] Check IP address, retrieved April 11, 2013, from http://checkip.dyndns.org.

[50] **K. WALKOWIAK**, P2P multicasting network design problem — Heuristic approach, 2010.

[51] **K. WALKOWIAK**, Survivability of P2P multicasting, 2009.

[52] **Y. YUEXIANG**, **L. CHAOBIN**, **H. GAOPING**, Feature research on unstructured P2P multicast video streaming, 2009.

[53] **B. ZHANG**, **X. LI**, **M. WIEN**, **J. OHM**, Optimized channel rate allocation for H.264/AVC scalable video multicast streaming over heterogeneous networks, 2010.

[54] **S. BOUDKO**, **W. LEISTER**, **C. GRIWODZ**, **P. HALVORSEN**, Maximizing video quality for several unicast streams in a multipath overlay network, 2010.

[55] **S. AHMAD**, **R. HAMZAOUI**, **M. AL-AKAIDI**, Adaptive Unicast Video Streaming With Rateless Codes and Feedback, 2010.

[56] **C. LIU**, **I. BOUAZIZI**, **M. GABBOUJ**, Advanced Rate Adaption for Unicast Streaming of Scalable Video, 2010.

[57] **M. QADEER**, **R. AHMAD**, **M. KHAN**, **T. AHMAD**, Real time video streaming over heterogeneous networks, 2009.

[58] Wamp Server, retrieved April 18, 2013, from http://www.wampserver.com/en/.

[59] MyISAM Storage Engine, retrieved April 18, 2013, from http://dev.mysql.com/doc/refman/5.1/en/myisam-storage-engine.html.

[60] MySQL Connector/Net, retrieved April 21, 2013, from http://dev.mysql.com/downloads/connector/net/.

[61] Touchless SDK, retrieved May 5, 2013, from http://touchless.codeplex.com/.

[62] MSDN, Create and Embed an Application Manifest, retrieved May 5, 2013, from http://msdn.microsoft.com/en-us/library/bb756929.aspx.

[63] MSDN, Constants (C# Programming Guide), retrieved May 6, 2013, from http://msdn.microsoft.com/en-us/library/ms173119.aspx.

[64] MSDN, Using Namespaces, retrieved May 6, 2013, from http://msdn.microsoft.com/en-us/library/dfb3cx8s.aspx.

[65] MSDN, System.Net.Sockets Namespace, retrieved May 7, 2013, from http://msdn.microsoft.com/tr-tr/library/system.net.sockets(v=vs.71).aspx.

[66] MSDN, System.Threading Namespace, retrieved May 16, 2013, from http://msdn.microsoft.com/en-us/library/system.threading.aspx.

[67] MSDN, MemoryStream Class, retrieved May 16, 2013, from http://msdn.microsoft.com/en-us/library/system.io.memorystream(v=vs.71).aspx.

[68] **A. SINGH**, Threading in C# with Example, retrieved May 18, 2013, from http://www.mindstick.com/Articles/13cd0056-e8ea-4637-b7c8-0146ea71fe4a/.

[69] Devart.Data.MySql Namespace, retrieved May 21, 2013, from http://www.devart.com/dotconnect/mysql/docs/Devart.Data.MySql~Devart.Data.MySql_namespace.html.

[70] The Base16, Base32, and Base64 Data Encodings, retrieved May 21, 2013, from https://tools.ietf.org/html/rfc4648.

[71] MSDN, HttpListener Class, retrieved May 21, 2013, from http://msdn.microsoft.com/en-us/library/system.net.httplistener.aspx.

[72] PHP Manual, ob_implicit_flush, retrieved May 21, 2013, from http://php.net/manual/en/function.ob-implicit-flush.php.

[73] MSDN, Working with Video, retrieved May 31, 2013, from http://msdn.microsoft.com/en-us/library/windows/desktop/ff819519(v=vs.85).aspx.

CURRICULUM VITAE

Personal Information

Surname, Name: ÖZDEMİR, Serkan

Nationality: Turkish (TC)

Date and Place of Birth: 20 August 1983, Kayseri

Marital Status: Married

Phone: 0505 620 53 35

Email: bilgi@serkanozdemir.com

Education

Degree	Institution	Year of Graduation
MS	Çankaya Univ. Computer Engineering	2013
BS	Ankara Univ. Computer Engineering	2009
High School	Talas FKT Anatolian High School	2002

Work Experience

Year	Place	Enrollment
2007-2011	Elitek Ltd. Sti.	Software Developer

Foreign Languages

Advanced English, Elementary German, Beginner Spanish

Hobbies

Playing musical instruments and traveling around the world.