VIRTUAL PRIVATE MULTIMEDIA NETWORK PUBLISHED AS SaaS (SOFTWARE AS A SERVICE) IN CLOUD COMPUTING ENVIRONMENT

BULUT BİLİŞİM ORTAMINDA SANAL ÖZEL ÇOKLUORTAM AĞLARININ "YAZILIM SERVİSİ" OLARAK YAYINLANMASI

NAEEM THJEEL YOUSIR

SUPERVISOR: Assoc. Prof. Dr. EBRU AKÇAPINAR SEZER

Submitted to Graduate School of Science and Engineering of Hacettepe University as a Partial Fulfillment to the Requirementsfor the Award of the Degree of Doctor of Philosophy in Computer Engineering

2014

This work named "**VIRTUAL PRIVATE MULTIMEDIA NETWORK PUBLISHED AS SaaS (SOFTWARE AS A SERVICE) IN CLOUD COMPUTING ENVIRONMENT**" by **NAEEM TH. YOUSIR** has been approved as a thesis for the Degree of **DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING** by the below mentioned Examining Committee Members.

Prof. Dr. Hayri Sever
Head ……

Assoc.Prof. Dr. Ebru Sezer
Supervisor ……..

Assoc.Prof. Dr. Nizami Gasilov
Member ……

Assist.Prof. Dr. Erhan Mengüşoğlu
Member ……..

Assist.Prof. Dr. Mehmet Serdar Güzel
Member …….

This thesis has been approved as a thesis for the Degree of **DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING** by Board of Directors of Graduate School of Science and Engineering.

Prof. Dr. Fatma SEVİN DÜZ
Graduate School of
Science and Engineering

ETHICS

In this thesis study, prepared in accordance with the spelling rules of Institute of Graduate Studies in Science of Hacettepe University,

I declare that

- all the information and documents have been obtained in the base of the academic rules
- all audio-visual and written information and results have been presented according to the rules of scientific ethics
- in case of using others Works, related studies have been cited in accordance with the scientific standards
- all cited studies have been fully referenced
- I did not do any distortion in the data set
- and any part of this thesis has not been presented as another thesis study at this or any other university.

NAEEM THJEEL YOUSIR

ABSTRACT

VIRTUAL PRIVATE MULTIMEDIA NETWORK PUBLISHED AS SaaS (SOFTWARE AS A SERVICE) IN CLOUD COMPUTING ENVIRONMENT

NAEEM THJEEL YOUSIR

Doctor of philosophy, Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. EBRU AKÇAPINAR SEZER

November 2014, (84) pages

This work is dedicated to the design and implementation of private and interactive multi-media control over the cloud. The design and implementation are accomplished using concepts and tools of latest software technologies (i.e., HTML5 facilities and WebSocket), where WebSocket is considered as the fastest software technology that could be deployed to transfer streams over the internet due to the synchronization of data transfer adopted by this technology.

The ultimate objective of this work is to build private stream player without the need to install more components to the internet browsers (e.g., flash player). This can be done by implementing Real Time Streaming Protocol (RTSP) on facilities provided only by the internet explorer (i.e., HTML5 technologies such as WebWorkers, primitive variables and Media Source API).

To implement RTSP, it requires two transport protocols: UDP (i.e., Asynchronous transport protocol); where AJAX is used to implement Real Time Control Protocol (RTCP) and TCP (i.e., Synchronous transport protocol); where WebSocket is used to implement Real Time Protocol (RTP).

Results are collected and an evaluation is conducted to reveal differences of the flash player embeddable component.

The implemented streaming system has been tested in two environment: one is the laboratory environment where could latency is not involved in the testing infrastructure, the resultant values for the streaming metrics showed a high performance in delivering streams in realtime manner, also the interactivity is maintained for each individual user. the test in phase was done for 4, 8, and 16 users at the same time and every user managed to interact the streaming system independently.

For second environment which is the implementation over the cloud, the system maintained the performance but network latency has been an important parameter for determining the overall performance; here 4 did not face cloud latency but 8 and 16 have been affected.

The cloud environment showed a great potential to host realtime interactive multimedia interaction system which is supported by encryption methodologies to preserve the security of the stream. the performance of the presented secure gateway for multimedia streaming showed a strong proportion to the encryption algorithm especially with higher length for the encryption and decryption keys.

Keyword: Video Streaming, Cloud, Media Source API, Google V8 engine.

ÖZET

Bu çalışmada bulut üzerinde özel ve etkileşimli çoklu-ortam kontrolü tasarımı ve uygulanması gerçekleştirilmiştir. Önerilen tasarım ve uygulama en son yazılım teknolojileri, kavramları ve araçları kullanılarak gerçekleştirilmiştir.

Bu çalışmanın temel amacı, internet tarayıcılarının daha fazla bileşen (örneğin, animasyon oynatıcı) yüklemesine gerek kalmadan özel akış (stream) çalar gerçekleştirimini yapmaktır. Bu amaç, tez kapsamında Internet Explorer tarafından sağlanan olanaklar ile Gerçek Zamanlı Akış Protokolü (RTSP) uygulanarak sağlanmıştır.

RTSP'yi uygulamak için iki aktarım protokolü gerektmektedir: UDP (Asenkron Aktarım Protokolü); AJAX, Gerçek Zaman Kontrol Protokolü (RTCP) ve TCP (Senkron Aktarım Protokolü) uygulamak için kullanılır. Sonuçlar, animasyon oynatıcı gömülebilir bileşeninin farklılıkları ortaya çıkarmak için toplanmış ve değerlendirme yapılmıştır.

Uygulanan akış(stream) sistemi iki farklı ortamda test edilmiştir: ilki laboratuar ortamında olup, bulut gecikme test altyapısı bu ortama dahil edilmemiştir. Elde edilen sonuçlarda, akış matrisi ölçüm değerleri gerçek zamanlı durumda akışların teslim alınmasında yüksek performans elde edilebildiğini göstermiştir. Ayrıca her kullanıcı için performansın korunduğu gözlenmiştir. Bu aşamadaki testlerde, ek olarak 4, 8 ve 16 eş zamanlı kullanıcı için incelenmiştir ve her bir kullanıcı bağımsız olarak kullanıcı etkileşim sisteminin içinde yer almıştır. İkinci test ortamı, bulut üzerinde gerçekleştirilmiştir. Uygulama sistem performansını korumuş, ancak ağ gecikmeleri genel performansı belirlenmesi için önemli bir etken olmuştur. Burada 4 kullanıcı için bulut gecikmesi yaşanmamış, ama 8 ve 16 adet kullanıcı için etkilenme gözlenmiştir.

Akışın güvenliğini korumak için şifreleme metodolojileri tarafından desteklenen bulut ortamı, gerçek zamanlı etkileşimli çokluortam sistemi barındırmak için büyük bir potansiyel göstermiştir. Çokluortam akışı için sunulan güvenli ağ geçidi (getway) performansı özellikle şifreleme ve şifre çözmede uzun anahtarları kullanımı tavsiye edilmektedir.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to Prof. Dr. Hayri Sever Head of Computer Engineering Department for her encouragement and continuous support throughout this study. I would like to thank my supervisor Assoc. Prof. Dr. EBRU AKÇAPINAR SEZER for his constant support, guidance and friendship. I am deeply indebted to my friends Dr. Ali Sadeq, Dr. Ammar Abdel Fattah, Brother and colleague Mohammad Nazim David and all other colleagues, whose assistance, stimulating suggestions and encouragement helped me all the way during my research and thesis writing.

I am deeply grateful to my dear wife, her patience, continuous motivation, support and welcomed presence. Without her encouragement, I would have never had the strength to complete this work.

I am also grateful to my parents, my father and mother, their continuous prayers for me throughout the years of study.

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ARPA	Advanced Research Projects Agency
HTML	Hyper Text Markup Language
NCSA	National Center for Supercomputing Applications
W3C	World Wide Web Consortium
CSS	Cascading Style Sheets
UDDI	Universal Description, Discovery and Integration
OS	Operating System
SaaS	Software as a Service
VOD	Video-On-Demand
MEC	Media-Edge Cloud
VPS	Virtual Private Server
SOAP	Simple Object Access protocol
CGI	Common Gateway Interface
J2EE	Java 2 Platform, Enterprise Edition
WWW	World Wide Web
WEAs	Web-based Enterprise Applications
AVI	Audio Video Interleaved
AIFF	Audio Interchange File Format
WAV	Waveform Audio File Format
ISDN	Integrated Services for Digital Network
URL	Uniform resource locator
AM	Amplitude Modulation
FM	Frequency Modulation
CD	Compact Disk
TV	Television
QoS	Quality of Service
MTU	Maximum transmission unit
SSRC	Synchronization Source
MPEG	Moving Photographic Expert Group
JPEG	Joint Photographic Expert Group
PT	Payload-Type
GPS	Global-Positioning-System
SCTP	Stream-Control-Transmission-Protocol
DSL	Digital subscriber line
IP`	Internet protocol
SDP	Session Description Protocol
IETF	Internet-Engineering- Taskforce
VoIP	Voice over IP
MySQL	My Structured Query Language
HTML	Hyper Text Markup Language
API	Application Program Interface
FIFO	First In First Out
TCP	Transmission Control Protocol

RTCP	Real-time Control Protocol
RTSP	Real-time Streaming Protocol
JSP	JavaServer Pages
EJB	Enterprise JavaBeans
UDP	User Datagram Protocol
HTTP	Hypertext transfer protocol
MVC	Model–view–controller
RTP	Real-time Transport Protocol
RSA	Ron Rivest, Adi Shamir and Leonard Adleman,
AJAX	Asynchronous JavaScript + XML
DOM	Document Object Model
SSL	Secure Sockets Layer
TLS	Transport Layer Security
FTP	File Transfer Protocol
XML	EXtensible Markup Language

1. INTRODUCTION

In the late 1960, the Advanced Research Projects Agency for the Department of Defense (ARPA) sponsored a conference at which several dozen ARPA-funded graduate students were brought together at the University of Illinois at Urbana-Champaign to meet and share ideas. During the conference, ARPA rolled out the blueprints for networking the main computer systems of about a dozen ARPA-funded universities and research institutions. They were to be connected with communications lines operating at a then-stunning 56 Kb [1][2].

One of the primary goals for ARPANET was to allow multiple users to send and receive information simultaneously over the same communications paths (e.g., phone lines). The network operated with a technique called packet switching, in which digital data was sent in small bundles called packets. The packets contained address, error-control and sequencing information. The address information allowed packets to be routed to their destinations. The sequencing information helped in reassembling the packets, which, because of complex routing mechanisms, could actually arrive in out of order-into their original order for presentation to the recipient. Packets from different senders were intermixed on the same lines [1][2][3][4].

As the Internet evolved, organizations worldwide were implementing their own networks for both intra-organization (i.e., within the organization) and inter-organization (i.e., between organizations) communications. A wide variety of networking hardware and software appeared. One challenge was to get these different networks to communicate. ARPA accomplished this with the development of IP-the Internet Protocol, truly creating a "network of networks," the current architecture of the Internet. The combined set of protocols is now commonly called TCP/IP [1][2][5].

The World Wide Web allows computer users to locate and view multimedia-based documents on almost any subject over the Internet. Though the Internet was

developed decades ago, the web is a relatively recent creation. In 1989, Tim Berners-Lee of CERN (the European Organization for Nuclear Research) began to develop a technology for sharing information via hyperlinked text documents. Berners-Lee called his invention the Hyper Text Markup Language (HTML). He also wrote communication protocols to form the backbone of his new information system, which he called the World Wide Web [5][6]. In particular, he wrote the Hyper Transfer Protocol (HTTP)-a communications protocol used to send information over the Web. Web use exploded with the availability in 1993 of the Mosaic browser, which featured a user-friendly graphical interface. Marc Andreessen, whose team at NCSA developed Mosaic, went on to found Netscape, the company the many people credit with initiating the explosive Internet economy of the late 1990s [6][7].

In October 1994, Tim Berners-Lee founded an organization—called the World Wide Web Consortium (W3C)—devoted to developing nonproprietary, interoperable technologies for the World Wide Web. One of the W3C's primary goals is to make the Web universally accessible—regardless of ability, language or culture[6][8].

The W3C is also a standardization organization. Web technologies standardized by the W3C are called Recommendations. W3C Recommendations include the Extensible Hyper Text Markup Language (XHTML), Cascading Style Sheets (CSS), Hyper Text Markup Language (HTML—now considered a "legacy" technology) and the Extensible Markup Language (XML). A Recommendation is not an actual software product, but a document that specifies a technology's role, syntax rules and other specifications[8][9].

With internet bloom software technologies are emerging every single day and others are fading the same quick as new technologies are emerged. Cloud computing is one of the most innovative approaches to build web applications over web. Cloud computing promises web developers to change the way computer and internet are deployed to build applications [7]. Could computing is an environment rather than a tool where programs and data are stored in distributed locations over the internet,

therefore, software developers can access their programs from anywhere around the world once they have access to the internet. By the advent of cloud computing no more desktop-centric applications are in demand anymore and most organizations are in migration process toward the cloud based application[6][7][8].

The kernel block of cloud based application is the web service and access policy where all functions which were encapsulated in binary libraries are now implemented as web services and published in special repositories such as UDDI (Universal Description, Discovery and Integration) which had been exposed to the public for free by giant software companies such as IBM and Amazon[7][8].

Client machines can execute programs using internet explorer shell which is subjected to intensive development recently, for example, chrome OS (Operating System) which is an operating system that its kernels are derived from Linux kernel and has been designed specifically to execute web applications and installed applications. Using chrome OS turns the desktop computer to a web thin client which has the ability to execute, debug, schedule and control the consumable web applications in online and offline fashions. Web applications use HTML5 as the language for implementing its business logic, HTML5 is still under development; this is despite the fact that most web browsers are already supporting it (e.g., Google Chrome, Internet Explorer form ver. 10). HTML5 specification includes new features such as supporting video, audio and search tags; YouTube has already begun supporting HTML5 videos [6] [7][8][9].

Is this dissertation a integral efforts are produced to migrate an attractive application to the cloud which is the multi-media applications. This dissertation aims to promote multi-media streaming applications to be a web service and ready to be used by software developer in their software. Implementing stream application (e.g., video players) is a challenging approach due to different aspects such as the network latency and security issues. Network latency is to be defeated by the coding

approaches where many codec packages are published over the internet to be used before start streaming media such WebM, Ogg, DivX and many others, while security issues are considered in the removing the capabilities of web applications to access kernel resources in the machine that executes the application[7][8].

1.1 Research Objectives

The research objectivity is to design and implement private gateway to access and share multimedia resources over the cloud; this software gate is to be published as web service in cloud computing environment. Along the implementation RTSP and its descendant protocols (RTP and RTCP) are implemented using web socket (TCP based communication channel) technology and HTTP (UDP based polling techniques).

Encryption algorithm is deployed to encrypt streams exchanged with the proposed software gateway; the typical usages of this gateway are secure media content management, media streaming and interactive application development. This proposal is about developing a new model to create private virtual network to stream multimedia contents over the internet between web services and client terminals. The new model will advertise this multimedia private virtual network as SaaS (Software as a Service) to be publicly available over the internet.

1.2 Problem statements

The dissertation is dedicated to investigate multimedia streaming over the cloud environment where huge number of applications in wide spectrum of domains is demanding this technology. In this dissertation a robust proposal has been put together to study and resolve the problem of streaming multimedia streams in the sandbox of internet explorer. Streaming multimedia within the internet explorer has the following statements:

- HTTP based streaming technologies are based on reliable transferring protocol (i.e., UDP) but with less performance. Many applications like broadcasting TV

channels over the internet, playing games over the internet and other applications of the kind, demand better streaming protocols in order to accomplish their tasks.

- Employing Applets to stream multimedia contents is not recommended anymore; this is due to the need for security reasons where applets can easily compromise the security of computer operating system.
- Employing third party to deliver multimedia streams is not recommended due to the demand for installing third party software module within the low level kernels of the internet explorer; this will grant it enough privileges to access underlying resources.

1.3 Motivations

The motivations for submitting the proposal in this dissertation are:

- Most software venders and software development providers migrate into the new platform of cloud where applications stores nowadays contain millions of applications that based on integrating cloud services.
- The need for reliable and high performance multimedia service to be published over the cloud; this is to support cloud based streaming applications
- web based developing environment has been developed and promoted by introducing new software technologies such as the advance operating system for chrome and the possibility to run web programs in multi-threading fashion and establishing communication sessions over TCP transport protocol.
- new software technologies have been introduced in modern web programming environment such as HTLM5 software technologies where many technologies are emerged (i.e., web worker, primitive data types and debugging tools)
- Google has introduced the V8 engine to build high performance web applications and a new server side development environment has been built over that engine; this is node.js where server side javascript is introduced for the first time to build non-stop applications.

1.4 Related works

In dissertation many previous works which have been introduced in analogues domain have been studied intensively to reveal points of intersection and to highlight contribution areas. Anyway most related efforts are presented below:

1- NGINX is presented in [10] to deliver MP4 and FLV video content using progressive download or HTTP pseudo-streaming. The progressive downloads rely on the efficiency of the streaming server to handle disk I/O and concurrent connections, while MP4 or FLV pseudo-streaming capabilities also allow the clients to use a simple “trick-play” technique to seek to the desired position in a video stream without needing to download the entire resource. NGINX plus which the updated version extends this capability to support adaptive streaming for video-on-demand (VOD) applications with Apple HLS and Adobe HDS. This work has announced many benefits like adaptive streaming which allows the video player to select the most appropriate bit-rate in real-time.

2- In [11] a new scheme for identification over the cloud is presented where the problem of lack of a unique identification of the “who” within the network infrastructure is presented and tackled by introducing the concept of a classification mechanism that is fine-grained enough to associate tenants, services and cloud providers to their network streams. The Tenant-ID, Service-ID and Cloud-ID is added as a tag to Layer-3 packets throughout the consumer-to-service communication. In this work, the identification scheme is assessed to be generic enough to be applicable across the whole cloud environment, thereby eliminating current limitations and enabling new network functionality. This identification scheme is controlling cloud based sessions to deliver streams over the internet while in our proposed identification scheme we consider labeling the stream chunk itself; this approach prevent attacking streaming session at the stream level.

3- In [12] an intensive investigation is conducted to describe the pros and cons of using chunked and unchunked videos, a new methodology for fairly comparing

the performance implications of video object granularity at web servers is presented. This work has come up to a result that conventional servers there is little performance difference between the two approaches (i.e., chunked and unchunked). This work has many issues in common with what we accomplished where we did an investigations find out if sending streams in chunks affect the overall performance we have concluded that in a web socket connection which is kept alive, the chunk size did not affect the performance but when reconnection is needed the chunked streams have less performance than unchunked.

4- In [Ramesh] a work is produced to explain the principal concepts of multimedia cloud computing and presents a noval framework, multimedia cloud computing has been addressed from multimedia-aware cloud (media cloud) and cloud-aware multimedia (cloud media) perspectives. In this work, a presentation for multimedia-aware cloud which addresses how a cloud can perform distributed multimedia processing and storage and provide quality of service (QoS) provisioning for multimedia services. This work proposed a media-edge cloud (MEC) architecture to achieve QoS; this MEC storage, central processing unit (CPU), and QoS adaptation for various types of devices are considered. Anyway, this work comes up to a conclusion that using multimedia's fundamental concepts can be used to improve mobile delivery applications. this work addresses the effect of using multimedia streaming the same way we use it in our proposal but with less focus on using TCP based communication sessions like we did in relying on web socket to deliver multimedia streams.

1.5 Requirement and Implementation plan

To implement the proposed system in this dissertation multiple phases are needed as it is shown below:

1.5.1 Phase one: Setup & configuring the environment

- Installing and setup linux based VPS (Virtual Private Server): here a reservation for linux based server is needed and a registration over the cloud also needed.
- Installing and configure linux Apache and TomCat servers: these servers are available in different versions each with different capabilities. TomCat server has to be of version greater than 7.0.41 in order to support required technologies (i.e., web socket)
- Installing Node.js and V8 google engine: this version has to be download through hub gate
- Updating chrome and firefox internet explorers for latest versions to support web socket and multithreading
- Configure Node.js and TomCat to peacefully work together.

1.5.2 Phase two: Determining and examining proper tools

- Two approaches are candidate for the implementation of the private gateway over the web:

  First: Servlet/ JSP applications hosted by Apache http server and TomCat web server 7.0.4 or later.

  Second: Node.js and V8 google engine; this is a new and modern approach to build stream servers

1.5.3 Phase three: Implementing RTSP, RTP and RTCP – Web socket and HTML5: in this phase we have the most intensive work where the protocol is designed and its kernel data structures are declared.

1.5.4 Phase four : Securing the transmitted stream using fast encryption algorithm (elliptic curve)

- Using Curve spline mathematics to select irreducible polynomial to built robust group; this group is used later to encrypt streams transmitted over the web

1.5.5 Phase six: debugging the implementation and analysis results

In this phase the proposed private gateway is implemented in local environment and all processes are debugged for bugs.

The output results are collected and analyzed before deploying in real environments

1.5.6 Phase seven: publishing the implementation as SaaS

- Register for public UDDI and gain publishing key
- use Servlet / SOAP to publish implemented gateway

build Servlet Filter to proxy incoming http traffic

CHAPTER 2

THEORETICAL BACKGROUND FOR THE PROBLEM DOMAIN

2.1 Introduction

In the last few years, the internet has become the most important communication network in the world. Due to its universality and growing popularity [13][14][15], the number of internet-based services increases from day to day. Basic features like e-mail and file-transfer applications are not wholly responsible for this unique growing process. It is also the World-Wide-Web and its web-based applications that share a very important role in the development of different applications over the internet. Due to the low bandwidth granted for private households in the beginning years of the internet [14][16], these web-based applications were very basic and did not have multimedia contents. But now after years of research and development, new broadband techniques have been emerged in the market like DSL and ADSL which have changed the situation dramatically. Today, it is common to stream any kind of multimedia content over the Internet yet with challenges in term of coding and network latency [13][15][17]. It is no longer a big deal to watch the news or a webcam based conferences anywhere in the world on your personal computer. This though is just the beginning of multimedia streaming over the internet, in the next few years the technology will expand drastically. Services like Voice over IP (VoIP) and Video on Demand are only two examples of challenges the internet faces for the future [16][17].

Most multimedia transmissions are accomplished on the Web, even though Hyper Text Transfer Protocol (HTTP) can be used to stream [13][14][16][17][18]. Afterwards, and for better streaming applications, the client's player contacts the multimedia server using a different set of protocols like the Real-Time Streaming Protocol (RTSP) which is an IETF draft standard derived from HTTP[17][18]; this protocol is deployed to communicate with the Real Networks client and server.

RTSP is used massively in commercial streaming media applications. So it is basically a client-server multimedia presentation control protocol, used to deliver multimedia streams over the IP network efficiently. Other techniques are also used to deliver multimedia streams like The Web caching techniques which are considered inappropriate due to the big amount of data encapsulated in multimedia streams. To achieve deployment simplicity, multimedia proxy services, such as prefix caching are to be deployed within standard streaming context; this is without the need of doing changes to the existing client/server software or network mechanisms due to the integration model dedicated for integrating servers to proxies. Although UDP can also be used also to deliver multi-media streams as quantized packets, RTSP typically runs over TCP which is more efficient but with challenges in reliability. RTSP is similar to HTTP in terms of contained structures such as [18][19][20]:

- The OPTIONS which is an attribute for server capabilities (e.g. RTSP version number, supported methods, etc.).
- The DESCRIBE attribute which captures the properties of a particular file (e.g. modified time and session description information, typically using SDP).
- The SETUP option, where Client and server state machines are configured in which transport parameters are also negotiated (e.g. RTP over unicast UDP on a particular port). The client transmit a SETUP message for each stream (e.g., audio and video) in the file.
- In the PLAY method, in this method the server starts streaming the multimedia file. Streams may hold the playback point.
- TEARDOWN method which terminates multimedia streaming session and thus releases all resources in both the server side and the client side. This method does not expect any resume commend for current ended session

2.2Real time Streaming Protocol (RTP)

In order to make the streaming offers new potential in term of building web applications, the Internet-Engineering- Taskforce (IETF) has developed the streaming protocol RTP (Real-Time-Protocol). According to the ISO/OSI layers [16][18], it is located on the Transport-Layer and typically operates overthe User-Datagram-Protocol (UDP). The Real-Time-Protocol, together with its extensions RTSP and RTCP makes it possible to stream multimedia content over IP based networks. Since UDP is an unreliable protocol with no loss detection or error correction mechanisms the delivery is very simple but still efficient. There are though reliable transport protocols such as the Transmission-Control-Protocol (TCP), which guarantee the delivery of each bit in the media stream [6][8][9], but this also means that when there is a loss of data on the network, the stream stalls while the protocol detects the errors and retransmits the missing data.

UDP has no implementation for retransmission, but to avoid a complete stall due to a temporary network error, clients can use a buffer to bridge over the waiting time for new data. Streaming media with real-time requirements over UDP does not work without a parameter that carries the time information of the datagram. Thus, a new protocol is needed: the Real-Time-Protocol (RTP) family. It consists of three different protocols, the RTP, the Real-Time-Control-Protocol (RTCP) and the Real- Time-Streaming-Protocol (RTSP), of which all work together. RTP is responsible for the actual delivery of the real-time data, while RTCP helps with a simple Quality-of-Service (QoS) management [16][19], such as packet loss detection. A retransmission function is not part of the RTP family. RTCP can only detect increasing number of lost packets and informs the sender to react for example by reducing the frame rate. The third member of the family, the Real-Time-Streaming-Protocol (RTSP), is deployed to carry control information such as "play" or "pause". It can be used to tunnel multimedia streams through firewalls where traditional security policy could prevent such streams from reaching their destinations.

Due to the missing reliability of UDP and the need for more efficient protocol that can provide different streams within one connection, a new streaming protocol has been developed: the Stream-Control-Transmission-Protocol (SCTP). It was essentially designed to serve transport of SS7 (Signaling System 7) over IP and to multiplex the data and the control information of a VoIP session, but another use of SCTP has been discovered. SCTP can also be used for mixing multimedia content with data that needs a high level of reliability. This makes SCTP an attractive alternative to conventional transport protocols because it is more efficient, secure and easier to handle [15][19][20].

The different types of streams do not only contain multimedia payload like video, but also sensitive data with a high need of reliability, for example, data gathered from the Global-Positioning-System (GPS).

2.2.1 Disadvantages of RTP

As mentioned above, streaming over RTP/UDP does not include data loss detection or error correction mechanisms. This is not needed in casestreaming multimedia data with real-time requirements. But what if we want to mix real-time data with data in need of a high reliability rate? This could be a detailed image every 2 seconds [15][16], chat-data or the data of the Global-Positioning-System. To realize this scenario we would need a RTP/UDP connection for a video stream and a TCP connection for each reliable stream. This would lead to a massive overhead within the network, making it difficult to configure a firewall for all these connections. The work in [15]shows how it is possible to bundle all the streams into one SCTP connection and define different types of reliability on each logical independent stream [16][19].

2.2.2 The RTP Header

In figure (2.1), the header of RTP packet is presented where most important fields are declared.The most important fields for the guarantee of real-time delivery are the sequence-number and the time stamping fields.

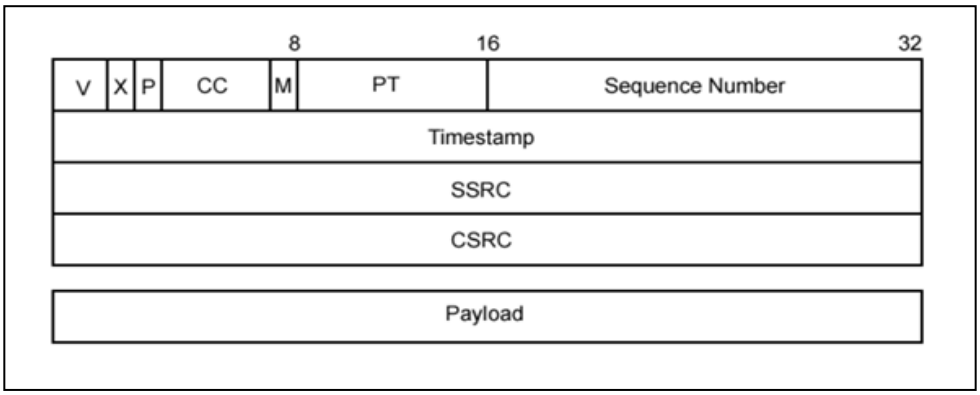

Figure 2.1: RTP Packet Format

As it is presented in figure (2.1), the Payload-Type (PT) is the field that used by the applications to determine how to interpret the payload [15]. A profile stipulates the default static mapping of payload type codes to payload formats. Typical profiles are defined for the sending of JPEG-Files, MPEG-Videos or PCMA-Audio. It is also possible to define dynamic profiles for the use of additional Payload-Type codes. The 32- bit value of the Synchronization Source (SSRC) field is used to identify the sender; this value is a random numeric value which is used to discriminate RTP sessions where no two synchronization sources have the same SSRC [20].

2.2.3 Streams

One of the main concepts and also one of the most significant differences between SCTP and TCP is that SCTP's usecompletely independent data streams within one association. To manage this mapping, a unique stream-ID is used for every data-chunk [16][19][20]. Therefore, it is no longer necessary to guarantee a complete in-order-transmission, the order matters only within one stream. This means, the different streams cannot block each other. With TCP this could only be achieved by

setting up one connection for each stream, which would lead to a reduced bandwidth due to the overhead. One example of overhead is the flow and congestion control. Since we would have a number of different TCP connections, each connection would use its own control mechanism, while SCTP only uses one per association that covers all streams as it is presented in figure (2.3).

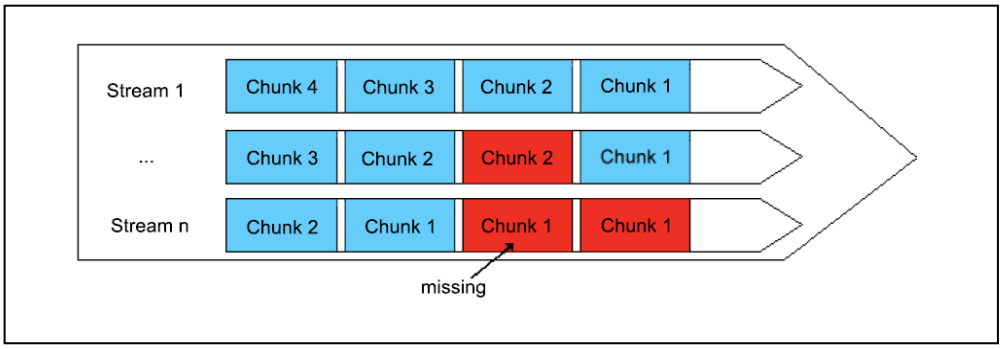

Figure 2.2: Different streams within one association

Figure (2.2) shows the independence of the different streams. Assuming that the server would like to send four chunks on every stream to the client, and that some chunks are lost along the way, the figure shows the data flow within an association. While the chunks in stream 1 go through without any problems, the second chunk in the second stream needs to be retransmitted, the same for the first chunk of the last stream. But since this retransmission does not block the data flow of the other streams, this makes the multi-streaming mechanism one of the most important benefits of using SCTP [16].

2.2.4 SCTP Packets

This section discusses the structure of a SCTP-packet, which forms the payload of an IP-packet (if SCTP runs over IP). A SCTP packet can be divided into a common-header and into chunks (see Figure 2.3). The common-header provides general information such as the source and the destination port of the packet [15][17].

Chunks may contain either control information or user data and can be multiplexed into one packet limited by the Path-MTU size.

2.2.4.1 The Common Header

The common-header of a SCTP packet as shown in Figure (2.3) consists of 12 bytes. It has a source and a destination port number to allow the multiplexing of different associations at the same address. The main concepts of these numbers are the same as the ones used by TCP and UDP. The next 32-bit value is called the verification tag. It prevents the insertion of an out-of-date or false message into the association. This association-specific value is exchanged between the two endpoints at the beginning of an association. The last 4 bytes of the common header, which makes up the checksum, are used to protect a SCTP packet from transmission errors. It is obvious that this 32-bit checksum is more robust than the 16-bit form used by TCP [15][17][18].

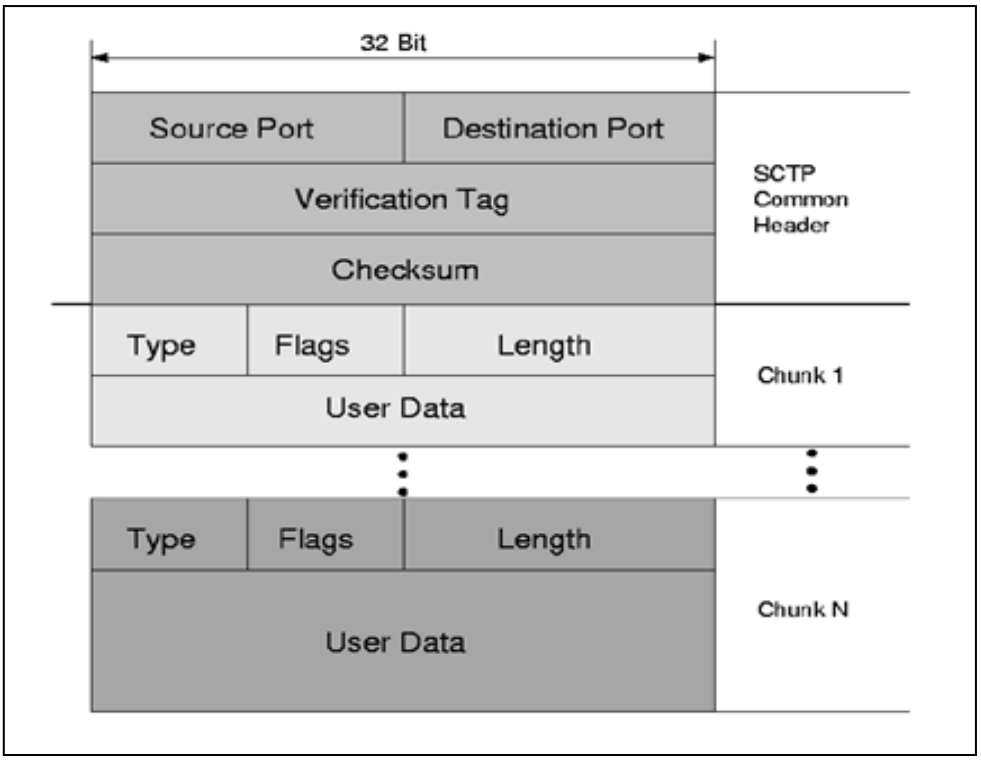

Figure 2.3: The common-header of a SCTP packet

2.3Real-Time Control Protocol (RTCP)

Basically, RTCP is a protocol designed to monitor the delivery of RTP packets along streaming sessions, and to providing feedback as it regards the quality of the distribution of streaming data. RTCP also provides the ability to establish an identification scheme (CNAME) for participants [15][16][18][19]; this is besides scaling the transmission of RTCP packet to fulfill the expanded number of client to prevent the producing excessive feedback traffic in huge multicast groups. It also provides minimum session control statistics. RTCP packets comprise of BYE packets, sender reports, source descriptors, receiver reports and APP packets as a special application function. Receiver reports contains the stream's SSRC, the fraction of RTP packets lost, the sequence number of the last RTP packet received, and the packet inter-arrival jitter. This information is used by senders to update their transmission speed or to shift to different encoder. The sender reports contains the stream's SSRC, the sequence number of the last RTP packet sent, the wall clock time of the last transmission, and the number of RTP packets and bytes sent. The client is able to perform media synchronization using the RTP timestamp and wall clock time information [15][16][20]

2.4 Prefix Caching in RTSP

This section presents a description on how client delay can be reduced using caching protocol information at the proxy [15], and how the RTSP Range header can be utilized to fetch the stream suffix. Figure (2.4) presents the sequence of RTSP messages handling when the proxy has cached the prefix of each stream (e.g. audio and video). The figure also shows the description of the appearance and the options supported by the server.

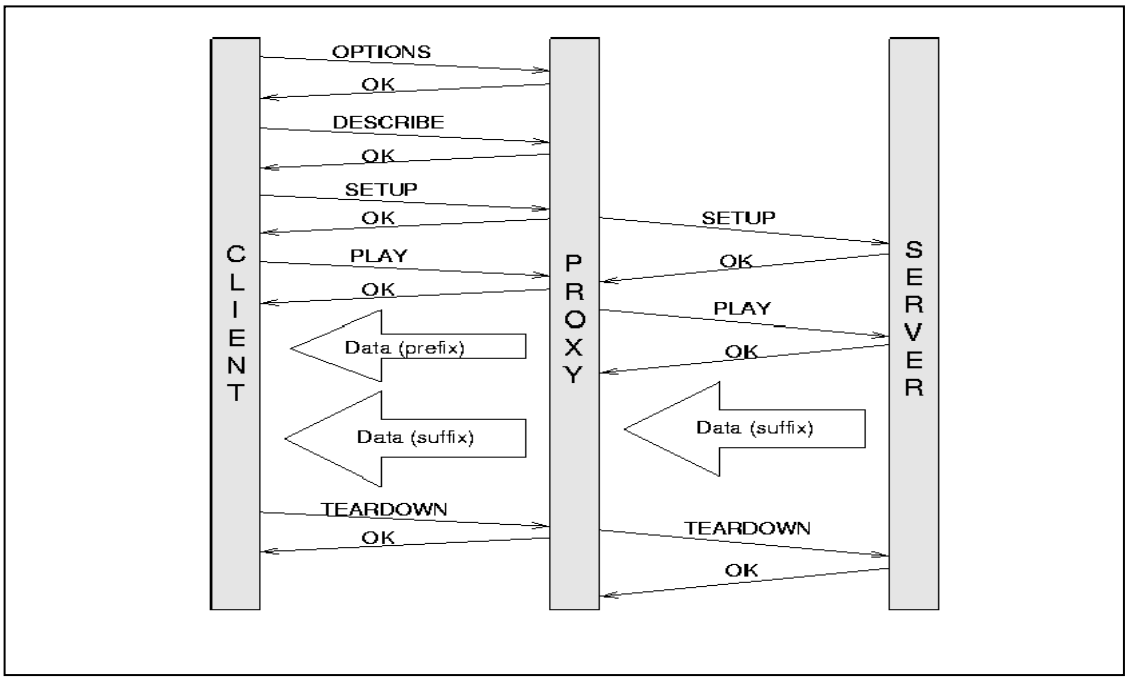

Figure 2.4: sequence of RTSP messages handling when the proxy has cached the prefix of each stream

2.5 Basic RTSP Operation

The tackling of user requests depends on whether the proxy has coached the prefix or not and he corresponding RTSP information. The OPTIONS message is used by the client to find out which techniques are supported by server. The basic techniques as OPTIONS, DESCRIBE, SETUP, PLAY, and TEARDOWN are supported by the server proxy [15][19][20]. The proxy can react directly to the request of user's OPTIONS to avoid the delay to contact the server, as given in Figure (2.5). Particularly, the proxy could act in response either with a default list of basic techniques or a list of techniques which are cached during earlier interaction with the server. An incorrect list of methods might be occasionally sent by the proxy when the server ceases to support any. If an unsupported method is to be invoked by the client, the server would prompt with an error message such as “Method not allowed" [15].

The proxy can cache the current list of methods and send to the user. This error does not cause streaming to stop working. The DESCRIBE message consists of one or more audio and video streams which is utilized to recover information related to the particular presentation. The DESCRIBE message is optional like the OPTIONS message and does not influence the RTSP state machine at the user or the server. The proxy can act in response directly to the user if the description stays in the cache; otherwise, the proxy must call the server to tackle the problem. However, some times the cached DESCRIBE information may not be up-to-date or upgraded. The proxy can employ a various methods to prevent the likelihood of stale information [17][18].

The SETUP request is utilized to discuss the transport parameters, containing the transport protocol (e.g. RTP) and the TCP/UDP port numbers. At the end of the RTP and RTCP connections port numbers are generated and session identifier is chosen after receiving the client SETUP message. The port numbers and session identifier are forwarded to the user. In the meanwhile, the proxy creates a separate SETUP message and sent to the server. Each stream in the multimedia presentation generates separate connection on both user proxy and server-proxy paths which consists of TCP or UDP connections for RTP and RTCP. The proxy could demand that the server stream RTP packets over TCP (or corresponding semi-reliable transport protocol) as the cached prefix could hide transient sections of TCP congestion when the server is not able to send packets fast enough for continuous playback. To synchronize the transport of RTP and RTCP messages, the proxy must sustain a mapping schedule to direct messages to the suitable outgoing connection with the suitable session identifier [15][16][17][18].

The PLAY message can be sent by the client to the SETUP request after getting the proxy reply. The proxy can reply instantly to the PLAY request or demand and start the streaming of RTP and RTCP messages to the used if the prefix is cached. The user can initiate playback with no waiting for the proxy to communicate with the server. The proxy may or may not receive the SETUP response from the server which depends on the delay between the server-proxy paths. Once a SETUP response from

the server is arrived at proxy and also indicates the server's session identifier, the proxy transfers the PLAY request to the server with the suitable Range header. The proxy can initiate sending RTP packets in the suffix which is retrieved from the server after transmitting the prefix of the stream to the client [16][17].

The proxy may choose to hold longer before contact to the server to transmit the suffix, depending on the size of the prefix. Huge storage resources are required at the proxy if the suffix is fetched too early which cause an extra server load and network may traffic if the user finally pauses or stops the transport of the presentation. When the proxy needs to initiate transmitting the corresponding data to the user, the suffix should be requested adequately ahead of. Requesting of the suffix bit early permits the proxy to hide the round-trip delay of server proxy (which can be determined from RTCP reports), or further delays for other functions as smoothing or transcoding can also hide[14][15][16][17].

2.6 Seamless Transmission of RTP Packets

In order to link the prefix and the suffix, all RTP headers must be consistent otherwise the two parts will not be associated to the same stream. The sensitive fields such as sequence numbers, timestamps, and source identifier (SSRC) are selected discretely by the proxy (for the prefix) and the server (for the suffix). Therefore, the proxy alter the RTP header fields of the suffix to match the SSRC chose for the prefix, the timestamps and sequence numbers to point out that the suffix must be after the prefix. The proxy overwrites the SSRC field in each RTP packet with particular value selected in streaming the suffix as part to initiate transmission of the prefix. The timestamp and sequence number are used to transmit the last RTP packet of the prefix known by the proxy. The base timestamp and sequence number transmit the suffix for the server which is provided in the RTP-Info header in the PLAY response. The proxy can then add/subtract the suitable constant for the timestamp and sequence number fields for each packet in the suffix [18][19].

2.7 Decoupling of Client and Server

Prefix caching disconnects the server communication from user reception. New challenges arise in terms of cache coherency and feedback control due caching partial streaming contents and overlapping of the prefix transmission to the proxy-user path with the suffix on the server-proxy path. This section identifies the major issues and provides different possible solutions [14][18].

2.8 Quality of Service

Quality of Service (QoS) originates in communications to explain particular technical characteristics of data delivery such as transit delay, throughput, error rate and connection establishment failure probability. These parameters are usually related to lower protocol layers and are not noticeable or confirmed by the applications. These parameters are still enough to characterize the transferring time independent data in communication networks [13][19].

When time dependent data is transferred over communication networks, for instance the real time streaming of audio-visual data, a wider view of the service quality have to be utilized, where the whole distribution system contributes to provide the guaranteed performance levels. The following definition of QoS has suggested by [20]:

"Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application". According to [20], the most important properties for real-time applications are temporal characteristics as delay, jitter, bandwidth and synchronization, and reliability characteristics as error-free delivery, ordered delivery and fairness.The required values related to these QoS parameters are estimated by using limits of human perception. For instance, if a round trip speech delay is exceeded more than 300 ms, the communication is disconnected [15][18].

In some cases, these parameters cause conflicts in requirements e.g. in the de-jittering buffer the selection of a low statistical bound of the delay variations is favored to minimize the delay. On the other side, it might results to a high packet loss ratio, as part of the de-jittering process [20].

2.8 Classification of Real-time Audio-Visual Streaming Services

Real-time streaming applications can be classified into diverse categories based on the providing service and its tolerated delay.

2.8.1 Information Retrieval Services

These different kinds of services contains video-on-demand, where the users choose a specific TV program or movie to watch. In usual practice, these services are not delay sensitive. The users accept to wait for few second from the moment presses "play" and the video sequence is initiated. These services are usually suitable for unicast [16].

2.8.2 Communicative Services

These types of service contains videoconferencing and video telephony. These services are delay sensitive and response time. The end-to-end delay should not exceed more than 150 ms in videoconferencing [16]. Infect, various researcher suggest different delay limits. (One suggested by Wolf considered as a quite stringent requirement.) These services can be either unicast or multicast.

2.8.3 Distributive Services

Distributive services are electronic services that have broadcasting/multicasting for continues stream such as video stream of a digital TV service. These services are very much sensitive to delay in receiving where viewers need to synchronize their watching for example they might call in live to the program and take part of it (e.g., quiz show and other competitions). Movie channels are less delay sensitive, however it should be noted that excessive buffering at the receiver may introduce a too long channel change time [16][18][20].

2.9 Principles of Streaming

There are two different principles of streaming of audio and video over networks where these streaming schemes (i.e., push and pull) have different policies for transferring media stream over the network to their designations. The synchronization problem is very different in these two cases [15][17].

2.9.1 Push Method

In the push case the source is controlling the rate of the stream of data. The sink has to estimate the time-base of the source and slave its play back rate to that of the incoming stream. This method is suitable for distributed services and the only method that can be used in broadcast/multicast applications, but it can also be used in unicast [15][17].

2.9.2 Pull Method

In this method, the sink controls the time-base/rate of the data from the source. Some sort of flow control protocol has to be used in this case. The source should transmit at a rate higher than the "normal play back speed", and the sink fills its buffer up to a certain level. At this level, the sink issues a "stop transmit" command back to the source, which temporarily stops transmission. Until the receiver buffer decreases to a certain level, the sink issues a "continue transmission" command, and another cycle starts. This method is suited for retrieval services. The pull method can only be used in unicast [15][16][17].

2.10 Overview of Streaming Audio and Video

People have experimented with transmission of sound and video over the Net after the explosion of interest in the Internet in 1993. Because of the long time required to transfer an entire multimedia file over slow links, it was a discouraging experience. You might wait 10-30 minutes in downloading an audio file whose elapsed playback time is only 2 minutes. Video, which carries much more information than audio, requires even longer download time, just to experience a 1/8 screen, slow-frame-rate, blurry movie [16][18].

The advent of streaming media changed all that. Streaming media uses a well known old concept –*buffering*– to allow the playback of multimedia content while it is being downloaded. A buffer holds a reservoir of content sufficiently large to smooth out the bumps in playback that may be caused by momentary server sluggishness or network congestion. This happens when you use an audio CD player designed for joggers, you will experience the same basic concept. The laser that reads the data pits on the CD may skip out of place when it hits a rough sector, which would normally cause an audible interruption in playback. A memory buffer holds enough seconds of sound for the player to continue the music uninterrupted until the laser can find the right track, and the CD can rotate to the right sector, so as to pick up the data stream. The

listener would never realize what is happening. In the same way, buffering allows Internet streaming media to maintain continuous playing of music or speech in spite of the occasional delay in file download.

Streaming media combines this concept of buffered real-time playback with *compression* to make viable what once might have been considered impossible – delivering to hundreds or even thousands of simultaneous listeners. Each of these listeners has his or her own Internet connection from remote server to local desktop, delivering audio in AM radio, or even FM, quality. In video streaming, there is so very much information in full-frame, full-motion video content which makes the story disappointing. Nonetheless, real-time streaming of video has also improved greatly as new “codecs” are designed with better and better compression. Video streaming via the internet is nowhere near the quality of conventional television, but it is serviceable for some applications, and it is improving continuously [15][16][18][19][20].

The basic steps, as presented in figure (2.5), in sending out content via streaming are:

- Create or obtain content. The content might be a recording you make, or it might be produced as part of a live event. (live events and manipulating these events are discussed in this chapter)
- Encode the content into the special streaming format
- Use a streaming server to send the content to your listeners.

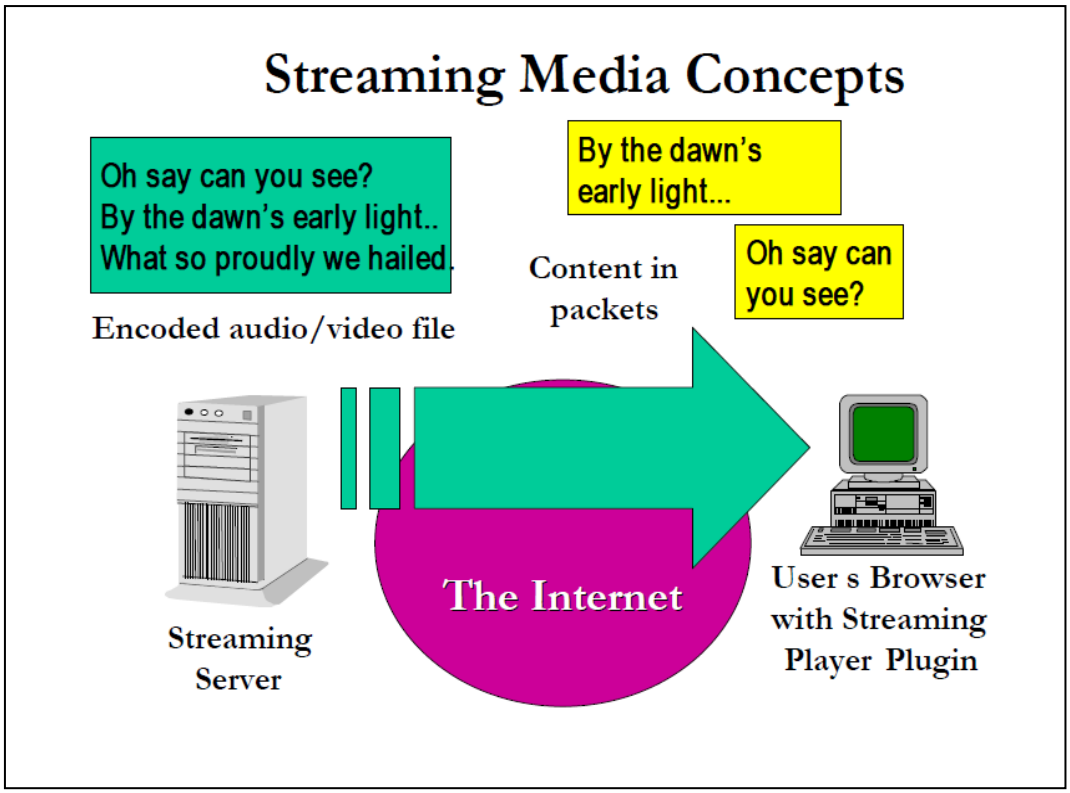

Figure 2.5: Media Streaming Concepts and Schemes

Here, we've encoded an audio rendition of the Star Spangled Banner. Playback occurs in this fashion:

- The entire audio file was placed on a streaming server to await requests for playback after being digitized. A streaming server is simply a server that has specialized streaming software installed – for instance, the Real System Server from Real Networks, Inc. The content has been specially encoded and placed into a file in the streaming server's file hierarchy [16].
- The server waits for a request from the user for a particular streaming document. When a user clicks on the URL for such a document, the user's browser sends a request to the streaming server [16][18].
- The streaming server finds the relevant content and prepares to send the file over the Internet. As the file transmission begins, the contents are broken into "packets;" each packet is sent as soon as it is prepared.

- The user's browser has had a streaming media plug-in installed, such as the Real Player. The plug-in places each packet into its buffer as it arrives, and, when the buffer is sufficiently full, the plug-in starts playing the content [16].
- Further packets continue to arrive. Thus, the buffer is being filled and emptied simultaneously, as playback continues – usually uninterrupted [16][17].
- Playback may pause when severe network congestion occurs. In this case, the user will experience a pause in the playback while the player attempts to refill the buffer.

2.11 Real System Media Types

The Real System offers several media types [15][16][18][19][20]:

- Audio: Suitable for speeches, oral history, and music delivery. Quality can be significantly good even over a 28.8 kilobit/second modem.
- Video: Video with audio in synchronization. High-quality video requires a large amount of bandwidth, so over dialup modems or even over faster links such as DSL or cable modems, it is typical to have video in a small window moving at a slow frame rate and with relatively little detail seen on screen. For "talking head" applications this conveys a sense of the personality of the speaker, but does not fully convey changes in facial expressions, gestures, etc.
- RealPix: This is a format for streaming delivery of photographic slide shows. Each frame is a still photograph, which may be on screen for many seconds; audio is streamed so that sound events match the time when a new still is downloaded. The effect is somewhat similar to some multimedia slide shows in museums.
- RealText: This format is used in training applications by streaming textual information.

- RealFlash: This format marries Real with Macromedia's Shockwave Flash format, so that efficient and impressive vector graphics can be sent in a streaming fashion.

There are definite applications for all of these media types, but most content providers have concentrated thus far on the audio and video types. RealPix could be a very promising format to consider for specialized applications, such as a slide show depicting community history through old photographs.

2.12 Tuning Your Streaming Content to Your Users' Bandwidth

There is a definite tradeoff between the resolution, or bit rate, at which you encode your streaming content, and the quality observed by your users. The higher the bit rate, the better the audio sounds, and the better the video appears. However, high speed network connections are not available to everyone. You may want to encode contents at more than one bandwidth rate as you are likely to have users connecting at a variety of speeds. Let's say most of your users connect over dialup, and some users connect over ISDN. You may want to encode your content at two bit rates –- say at about 20 kilobits per second for dialup (with video at a 160 X 120 frame size), and say 100 kilobits / second for ISDN [15][16].

Note that you can't encode at a rate equal to the modem speed, because you may not achieve that speed at a sustained pace. Note also that rated and width speeds may overstate true capacity. Many users of 56K dialup modems report that they seldom if ever connect at that rate.

The user may experience frequent pauses in playback, if you encode your data at a higher bit rate than his connection allows, or he may be unable to ever play the content at all. This is very frustrating to users. Although you cannot control the vagaries of network congestion on the global Internet, if you do offer your content at a

slow enough bit rate for the lowest common denominator, the more likely you are to have satisfied customers. Traditionally, in order to meet the bandwidth capabilities of a wide variety of users, publishers of streaming content have chosen to encode content at a variety of bandwidths, and offer hyperlinks to each title under each of the different bandwidths. For instance, you might offer content for 28.8 kilobit/second users, for ISDN users (up to 128 kilobits /second) and for Local Area Network users (up to 10 megabits / second). Each of these different speeds requires a separate, visible hyperlink on your Web site, targeted for users with corresponding connection speeds. The users need to know how fast their connection is, and choose among all the links offered.

The RealSystem G2 provides a mechanism called “SureStream” technology to obviate the need for offering multiple bandwidths. In a nutshell, this scheme stores multiple versions of the content, encoded at every desired bandwidth, in a single file. When the user requests the content, the RealPlayer and RealServer negotiate the appropriate bandwidth based on the user’s capabilities. In fact, if network congestion causes interruptions in delivery, the rate can be re-negotiated at a lower value, allowing the user to continue to partake of the content with minimal interruption. While the SureStream scheme is a great idea in theory, one problem is that previous versions of Real products do not support this mechanism. Because Windows 98 shipped with Real Version 4, users cannot participate in SureStream unless they upgrade their RealPlayers. Therefore, many content providers continue to “manually” offer separate links to separately encoded files [15][18][19]. (In fact, for that reason, the Toolkit video content is presented as separate files)

2.13 Components of the RealSystem

The RealSystem provides these components [15][16][19]:

- The RealSystem Authoring Kit. This includes the Encoder, which converts content into the Real format. You create your content in a

format you like – say AVI or Quicktime for video, or WAV or AIFF for audio. The Encoder transforms the content into Real's streaming format.

- The RealSystem Server. This is the specialized server that delivers content in real time to your users. Servers are available for Windows NT and for Unix. Real offers basic and "Plus" and "Pro" versions of these tools. You may want to evaluate streaming media using the basic versions, and graduate to the for-pay versions, which offer more features, when you decide to commit to streaming media as a methodology.

2.14 Preparing Your Content

The media content which is to be streamed has to be prepared using standard coding formats and tools; these formats should be popular and familiar to both parties involved in streaming session as shown in figure (2.6), For audio applications, Video editing demands quite a bit more in the way of computer resources than audio editing, but desktop video editing for streaming purposes is certainly viable today [16].

You will need a video-ready computer, or a video capture card such as the ATI All-in-Wonder or a TrueVision capture card. You will also need editing software such as Adobe Premiere. Alternatively, you may wish to rent or gain access to a professional editing device, such as a Media 100 or an Avid, used by TV stations, video production houses, and available at some schools. If you are capturing original content, you may do need a great deal of editing in order to make the material satisfactory. You may want to eliminate unwanted passages, or even reorder and significantly reduce the content as would be done for a documentary. Or, for some material, you may decide to make all the raw content available, a la C-Span. Edited material appeals more to some audiences; others like the cinema verite approach [15][16][17].

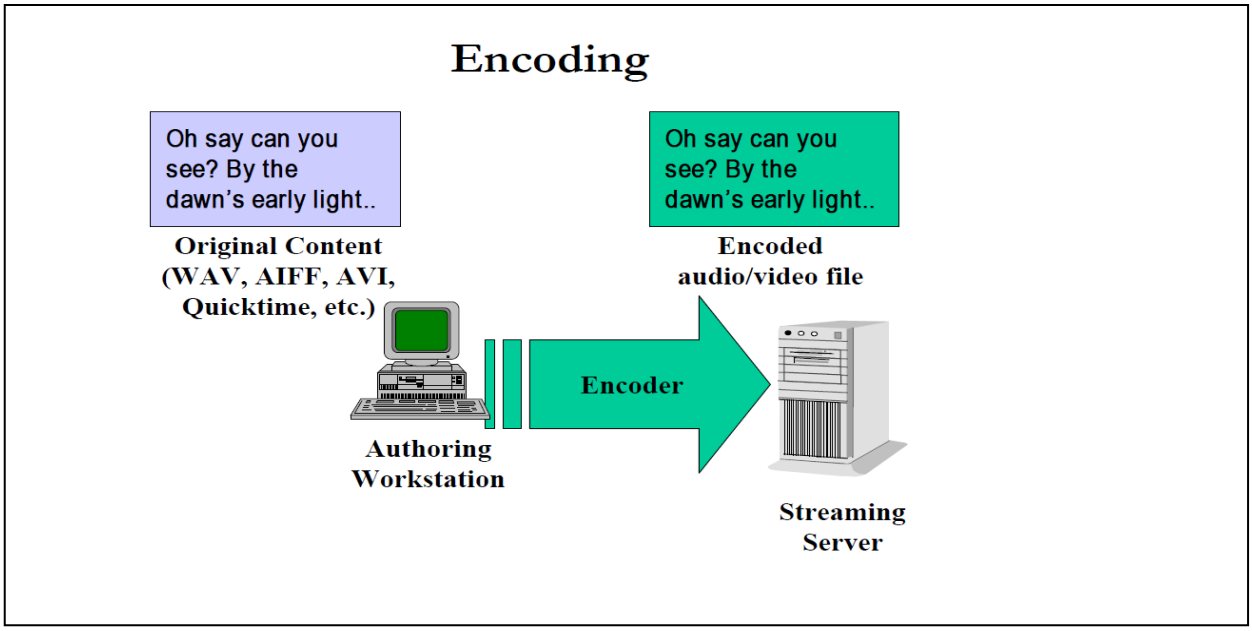

Figure 2.6: Media Contents are Prepared Using Standard coding Formats

Figure (2.6) depicts encoding of content that you have digitized into one of several popular audio or video formats. It is also possible to encode content into the Real format in real time. Real-time encoding is possible for a live event, or you can do encoding directly from an external source such as video tape or audio tape into the Real format [16][19].

Note that once the content is encoded, it is only usable by the RealSystem. A Real-encoded file is not meaningful outside the context of the RealPlayer, and there are no tools to translate from the Real encoding scheme back to any other audio or video format. Therefore, if you began with content in some other format, you probably will want to archive your source material even after encoding, in case you want to do further editing in the future.

2.15License Considerations

Real Networks offers free tools for preparing and playing content. It sells licenses for serving the content, as well as selling advanced content preparation and playback

tools. Although normally Real charges for streaming server technology, you may want to take advantage of exceptions to this [16][17][18][19]:

- Real Networks offers a “Basic” server with up to 25 simultaneous streams for free. A single entity can install one such server.
- School, library, governmental, and other nonprofits may be able to serve for free. Consult www.real.com for current license information.

A single “stream” is consumed for each user who connects to your streaming server for content at a time. If 1000 people are connected, you need a license for 1000 streams.

2.16 Enterprise Application Architecture

Enterprise Applications (EAs) are generally understood to be on-demand, user-interaction based applications that are meant to be accessed by multiple users, usually from the same organization. Web-based Enterprise Applications (WEAs) imply EAs made available through the Internet. These applications (EAs and WEAs) generally use databases for persistent storage. E-commerce sites (such as Amazon [AMAZON] and eBay [EBAY]), banking sites, webmail, online casinos and search engines are some of the many examples of WEAs [22][23].

WEAs are client-server applications. Making a WEA means implementing the server side of the application as well as what will run on client machines (usually in a client's web browser). This separation does not coincide cleanly with the boundaries of the three layers**Hata! Başvuru kaynağı bulunamadı.**. A common assumption is that all presentation components should exist on the client side, but this does not take into account server-side decisions about presentation or security [22][23].

Web applications have become an inseparable part of the Internet. They offer rich interactivity and functionality which would be unthinkable for older style, static web sites.

More and more companies are finding a way to offer their products and services online. For this, they need feature-rich web applications which allow them to register their clients, accept visitors' feedback, accept online payments, ensure that products can be easily found in their database and are properly presented to visitors and easily purchased [22].

More and more sophisticated web applications will be required by businesses over time, hence the requirement for web developers to make their work more productive and to make the web applications they produce more reliable, maintainable and easily scalable.

As a result, a significant number of Java web frameworks have appeared over time. Frameworks shift the focus of development to a higher level by bringing most low-level solutions "out of the box", thoroughly tested and ready for reuse. Many frameworks also bring with them some sort of architectural solution, based on the best practices of web development.

2.17Java Tools for Enterprise Applications

The main tendency in World Wide Web development has always been towards more functionality, more interactivity and riche user experience. Java, being platform-independent, looked perfect for WWW which spread across continents and computer networks. Java starts supporting enterprise applications by first introducing 'Java applets', which is a small programs downloaded to the client browser and executed there; this has added more interactivity to previously static web pages, this approach has faced degrading due to the need of the privilege required by applet code in accessing the machine. Figure (2.7) illustrate the process of downloading the applet to client side machine, it is client's responsibility to provide the environment enough to execute the applet [22][23].

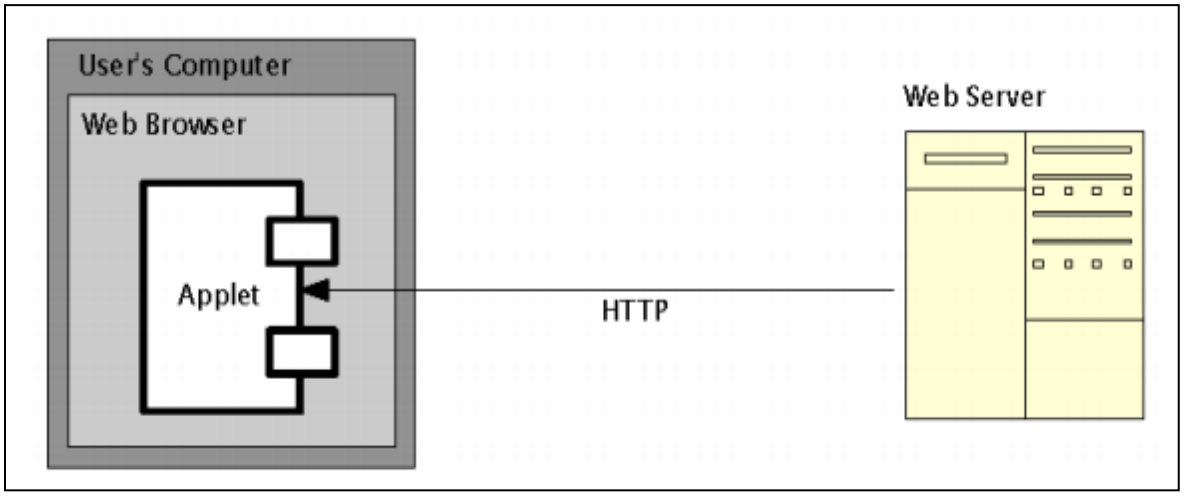

Figure 2.7: Applet executed in Client Side Machine with Support of Client Machine

The real success came to Java technologies when they began to work on the server side. Instead of downloading Java programs and running them in a client's browser, Java code could work on the server side and send only results to the client side; this is done by creating HTML pages "on-the-fly" and sending them to the browser [22]. This was exactly the idea implemented in Servlet technology, as it is shown in figure (2.8):

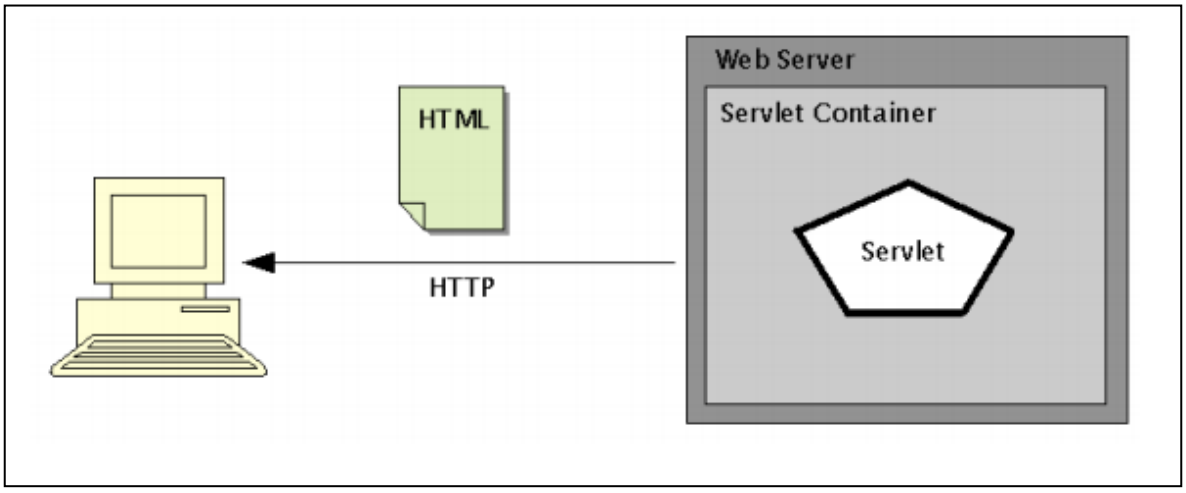

Figure 2.8: Java Servlets are executing at Server side and results transferred to client side

The idea of executing components of the enterprise application on the server side is not new where CGI programs written in languages like Perl or C had already been creating HTML pages dynamically for years. However, Servlets had significant benefits: higher scalability due to multithreading, functionality, readily available for developers in Servlet API and provided by Servlet container, inherent security and power of the Java language as well as all the services of J2EE platform which was beginning to emerge [22][23].

By using Servlet API, developers could concentrate on the functionality they were creating and leave many trivial tasks to the Servlet container. For example, all the request parameters were ready to use in the ServletRequest object and could be easily retrieved from it, and session management was very easy with all the burden of setting session cookies and most of the URL rewriting carried by the Servlet container [22][23][24].

The weak side of Servlets was that all the HTML output had to be created inside of Java code. All the design was tightly embedded into the code-hence the necessity to recompile Servlet with every little change in design. HTML pages created by designers would somehow have to be processed and converted into chunks legible arguments [22][23][24].

As web application grew and developed, the whole team had plenty of work to do. In other words, there was low maintainability due to the fact that presentation was embedded into the code. It was natural in this situation to invent an approach which would at least automate the conversion of HTML to be outputted into Servlet code. Indeed, many development teams created their own solutions for this problem, but they all became unnecessary after the emergence of JSP technology [24][27].

Java Server Pages: In their essence, JavaServer pages are the same Servlets with the only difference being that developers were allowed not to worry about how to process large amounts of HTML and insert them into Java code. This was done

automatically in the process of conversion of a JSP page into a Servlet. Insead of inserting HTML into Java code, the JSP developer was inserting Java code into HTML, using directives [22][23][25][26].

JSP was an invaluable solution for exactly those problems which were most difficult for Servlets – when the page to be sent as a response contained mostly static HTML, with just a small amount of dynamic content. However, when a large amount of Java code was embedded into the page, the mixture of Java and HTML was becoming very difficult to manage [25][26][27].

It is often implied that a web application can be considered as an enterprise level web application if it is distributed over a network and was built using Enterprise Java Beans (EJB) technology.

Figure (2.9) presents the architecture of Web application which used Java technologies:

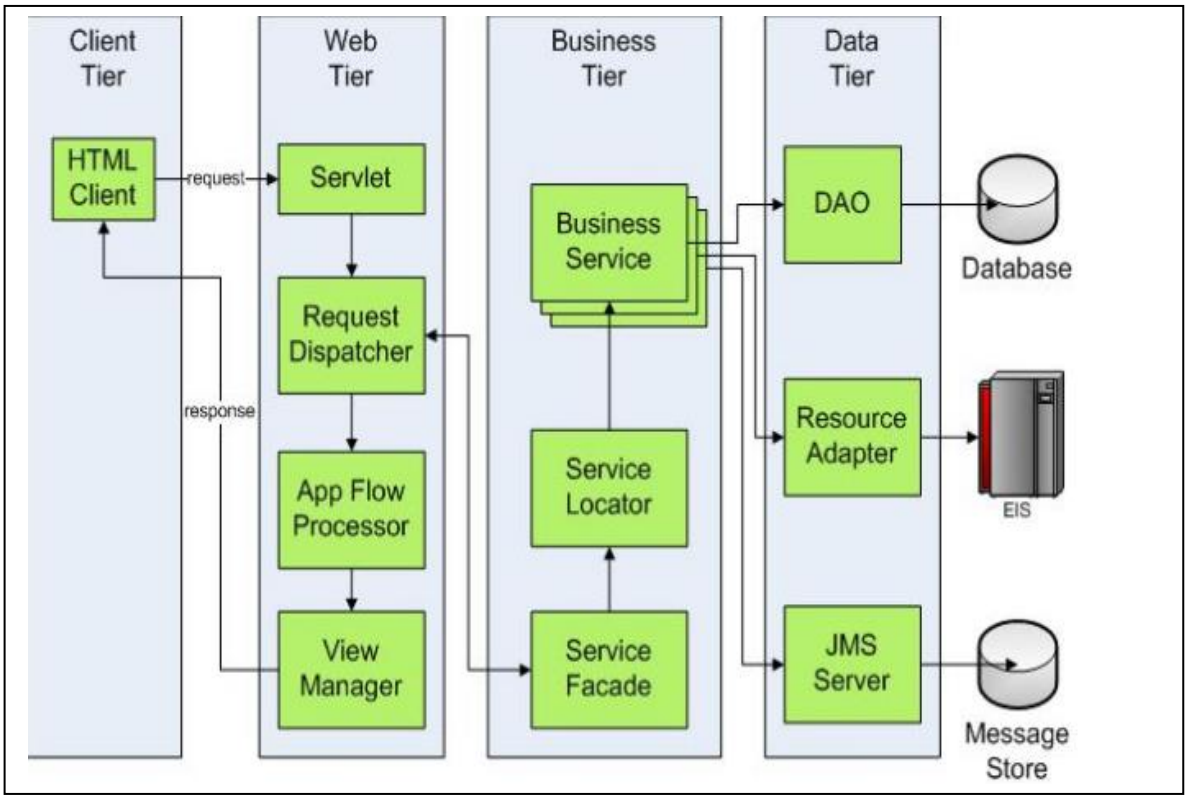

Figure 2.9 Java based Web Application Architecture

CHAPTER 3

RESEARCH METHODOLOGIES AND IMPLEMENTATION STRATEGIES

3.1 Introduction

This chapter is devoted to present the proposal of this thesis (i.e., the private interactive multimedia over the cloud). The presentation starts by introducing the structural scheme of the proposal and its essential components where the proposed system is a web application that is composed of multiple software technologies at server side and client side such as Apache/TomCat and Google Chrome.

After this brief introduction, this chapter presents the platform over which the proposed system has been built; the specifications of each component of the platform is introduced and justification for these required specifications is also presented. This chapter then move to introduce the conceptual view of the proposed with declaration of its basic concept.

Conceptual view is used in introducing the proposed system due to its correlation with the object oriented programming concepts used in the implementation. The methodologies deployed to implement every concept within the system are introduced using different presentation tools such as flowcharts, conceptual views and sequence diagrams.

Along this chapter the class hierarchy and organizational chart is introduced to allocate every concept within the entire system.

3.2Platform Setup

As it has been pointed out in chapter one that this work is a proposal to build a platform for conducting private multimedia sessions over the web, the first step in the

implementation was to setup the environment. The environment here is categorized into two categories: server and client.

In server category, http server (i.e., Apache server), web server (i.e., TomCat) and database server (i.e., MySQL) are the basic components, while in client category Google Chrome is the crucial component through which advanced software.,technologies can be launched (Media Source API, Web Socket and HTML5), figure (3.1) shows the basic components in the server side.

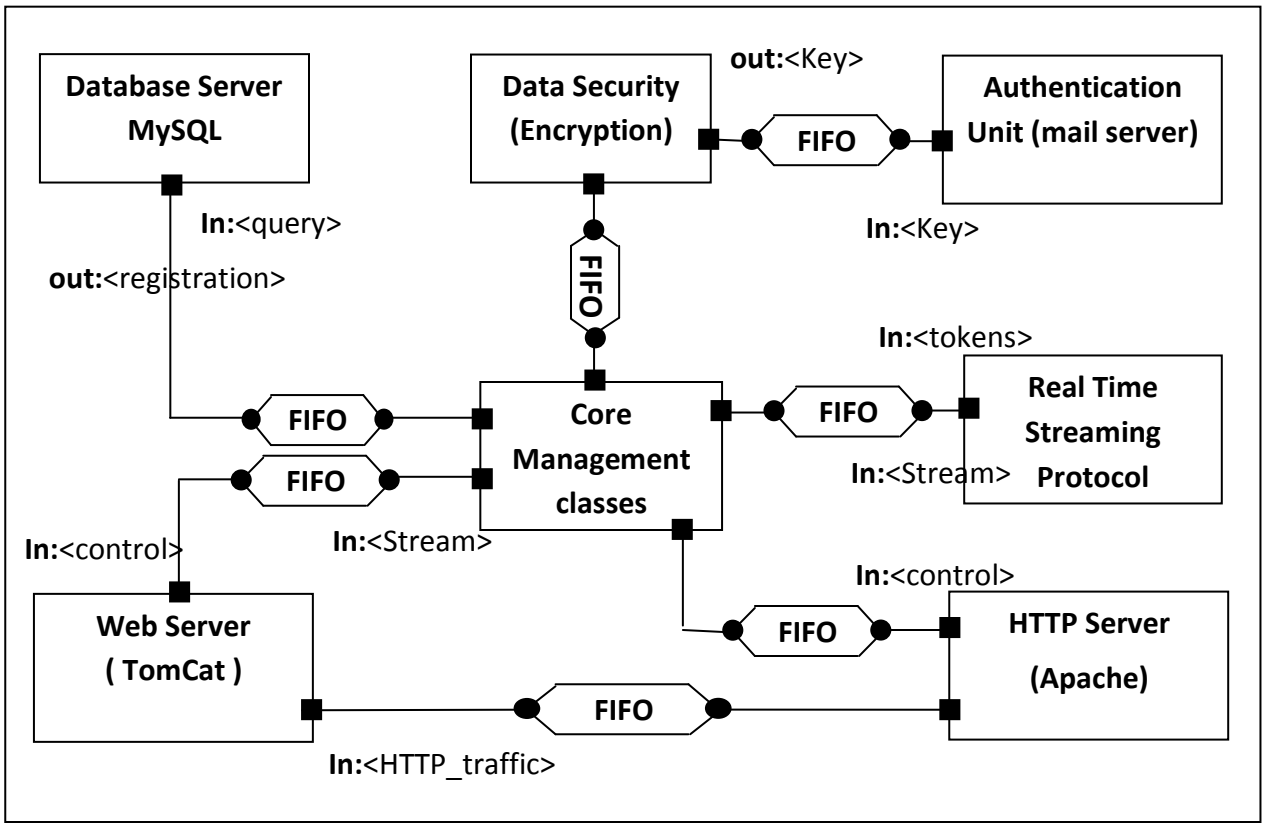

Figure 3.1: Conceptual View to Server Side Components

From figure (3.1), multiple components are required to implement server side of the proposal; these components will be briefed in the following sections of this chapter.

Anyway, the most important issue in setup developing environment is to install PrivatePro Class TomCat on a Virtual Private Server or Dedicated Server; this is the only way to start TCP based session (i.e., Web Socket Sessions).

In client side, HTML5 is the main player due to the fact that the proposal aims to build private interactive multimedia protocol based on Google chrome only. In other words no extra plugins are required as it is the case with flash player and other ActiveX components. Figure (3.2) depicts the platform required at the client side:

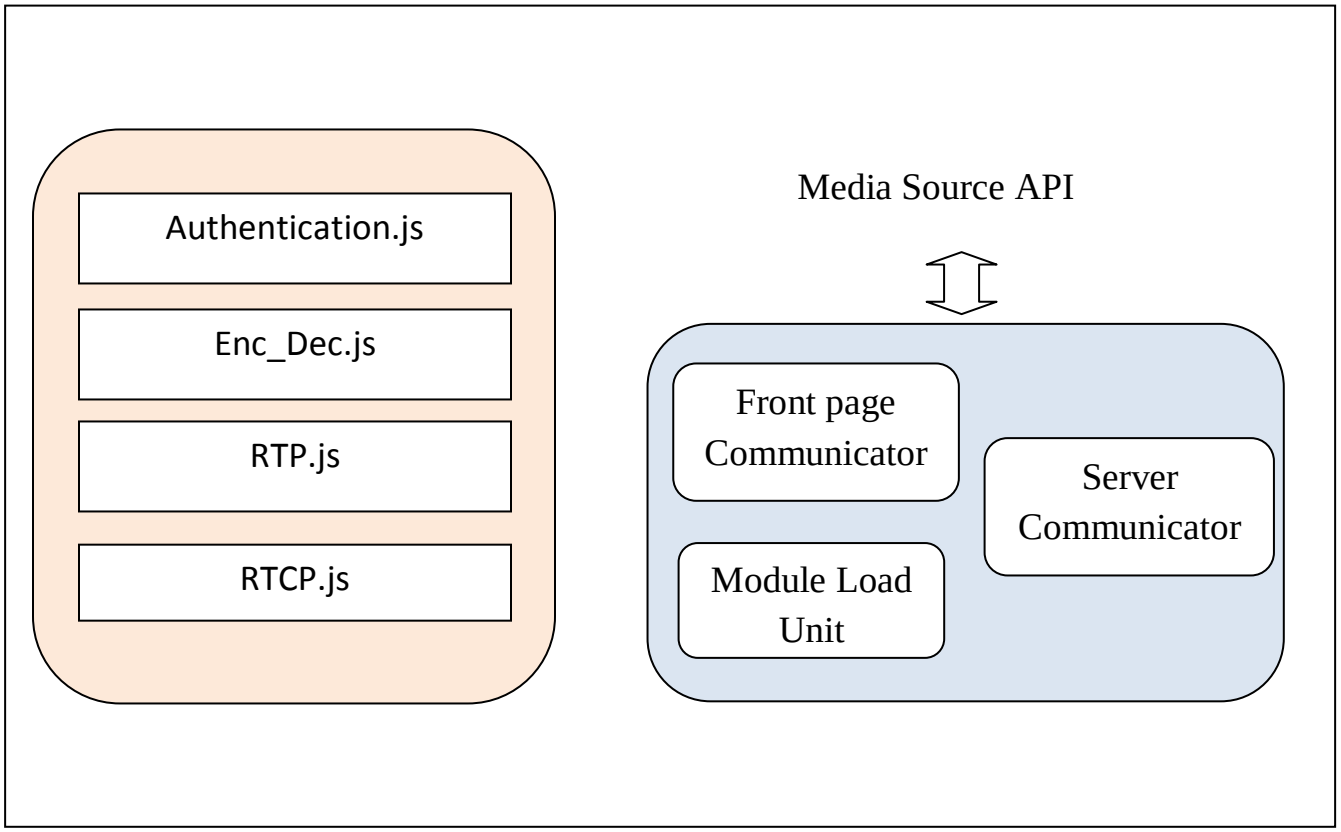

Figure 3.2: HTML5 Based Intelligent Agent Architecture

In figure (3.2), the platform is setup in client side by installing proper version of Google Chrome which introduces many software technologies demanded by the proposed system. The candidate version of Google Chrome is ≥ to (30); this is to guarantee its support to multi-threading, web socket, primitive data types and Media Source API.

In figure (3.2), RTSP protocol is represented by JavaScript Modules (i.e., RTCP.js and RTP.js) that are loaded to the system as worker classes; this is in order to make use of the significant multi-threading introduced by HTML5.

By setup Google Chrome (i.e., version ≥ 27) and required server side Java modules, as it is presented by this section, the environment becomes ready to develop the proposed system

3.3 Private Interactive Multimedia Stream Conceptual view

In this section, the general skeleton of the proposed system is presented to introduce basic concepts in the client side and server side. Figure (3.3) represents the conceptual view to the entire system where three components are shown: Video Stream Server, Client Side Media Player and Client side Worker Thread.

Video Stream Server is an Enterprise application built using java technology (i.e., Servlet/JSP and Enterprise Java Beans - EJB). The hosting environment should fulfill the requirement for starting WebSocket sessions (TCP-Connections), web socket sessions are the most proper technology utilized to carry out high rate transmissions over the web.

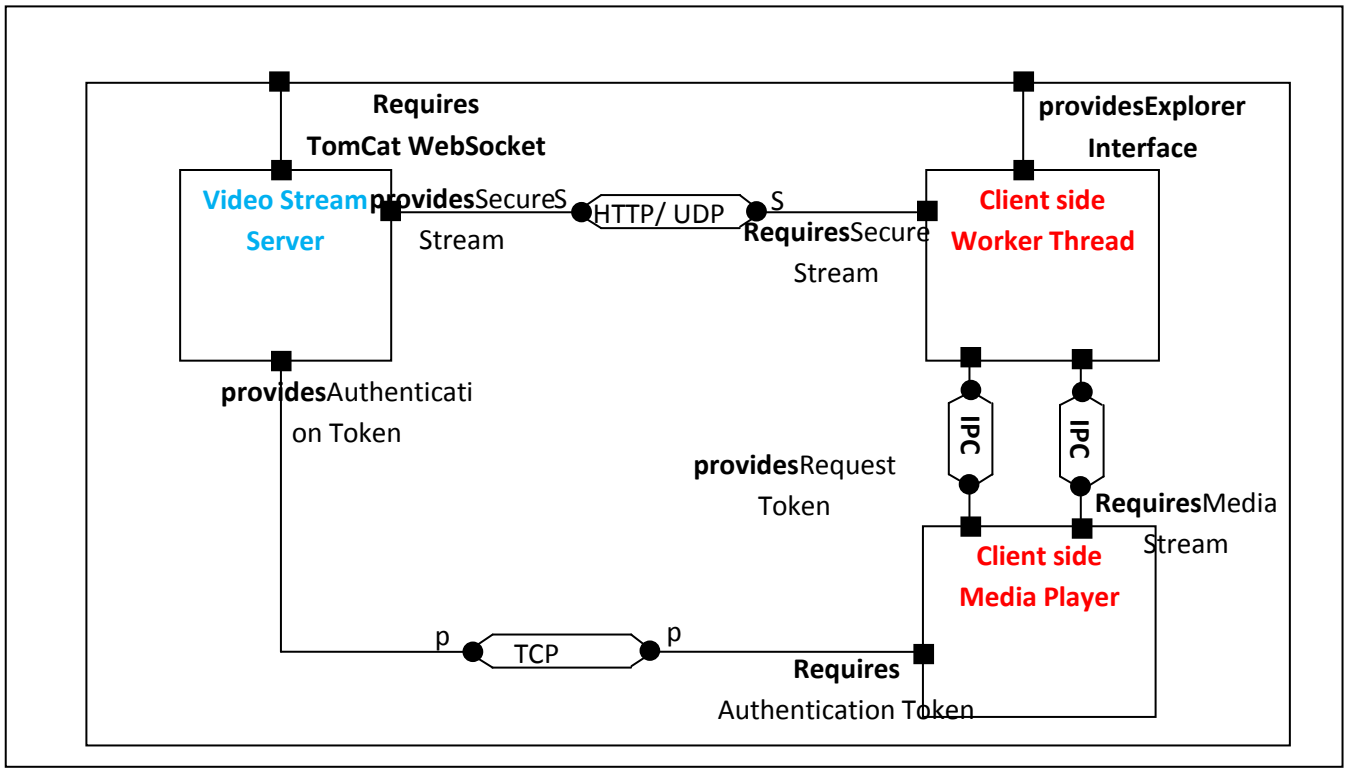

Figure 3.3: Conceptual View to Private Multimedia Streaming using HTML5/ WebSocket

Figure (3.3) shows that two transportation protocols have been deployed in this implementation: HTTP/UDP and TCP; these protocols are crucial in the implementation due to the specification of RSTP where control commands are sent over UDP and media streams are delivered through TCP connections. The most significant feature of WebSocket is the ability to conduct TCP over the Web platform by tunneling application level (i.e., HTTP) to transport level (i.e., TCP); this approach provides many security issues and guarantees the flow of multi-media streams through the firewall.

<u>Stage One:</u> Implementing Video Streaming Infrastructure

In this stage the internal architecture of the proposal is implemented where server side modules have been built up as web application using special java/servlet class “WebSocketServlet”; this class encapsulates all methodologies needed to

conduct TCP sessions and it extends the functionalities of traditional Servlet class which provides the ability to process standard HTTP Request (i.e., POST and GET).

Figure (3.4) presents the architecture of server side multimedia streaming application; in this figure, streams are assigned through slots (i.e., concurrent synchronized queues), each slot is identified by unique number changed on every session; this number is assigned to the slot and sent to the client through his/her email as it will be presented in next sections of this chapter.

Anyway, the proposed system is a private interactive multimedia streaming, thus, streams are delivered to the requestors upon satisfying certain constraints. In this thesis, requestors would receive identification key over the internet through email sent to their private email box; this key is used later on as slot identification ID.

As it is presented in figure (3.4), the essential components are Java enterprise modules that follow the framework of MVC (Model View Control); this framework isolates the presentation layer of the business logic where Java classes are built to implement different functionalities of the proposal (i.e., Read Binary File, authenticate client requests, key management , data base management and others).

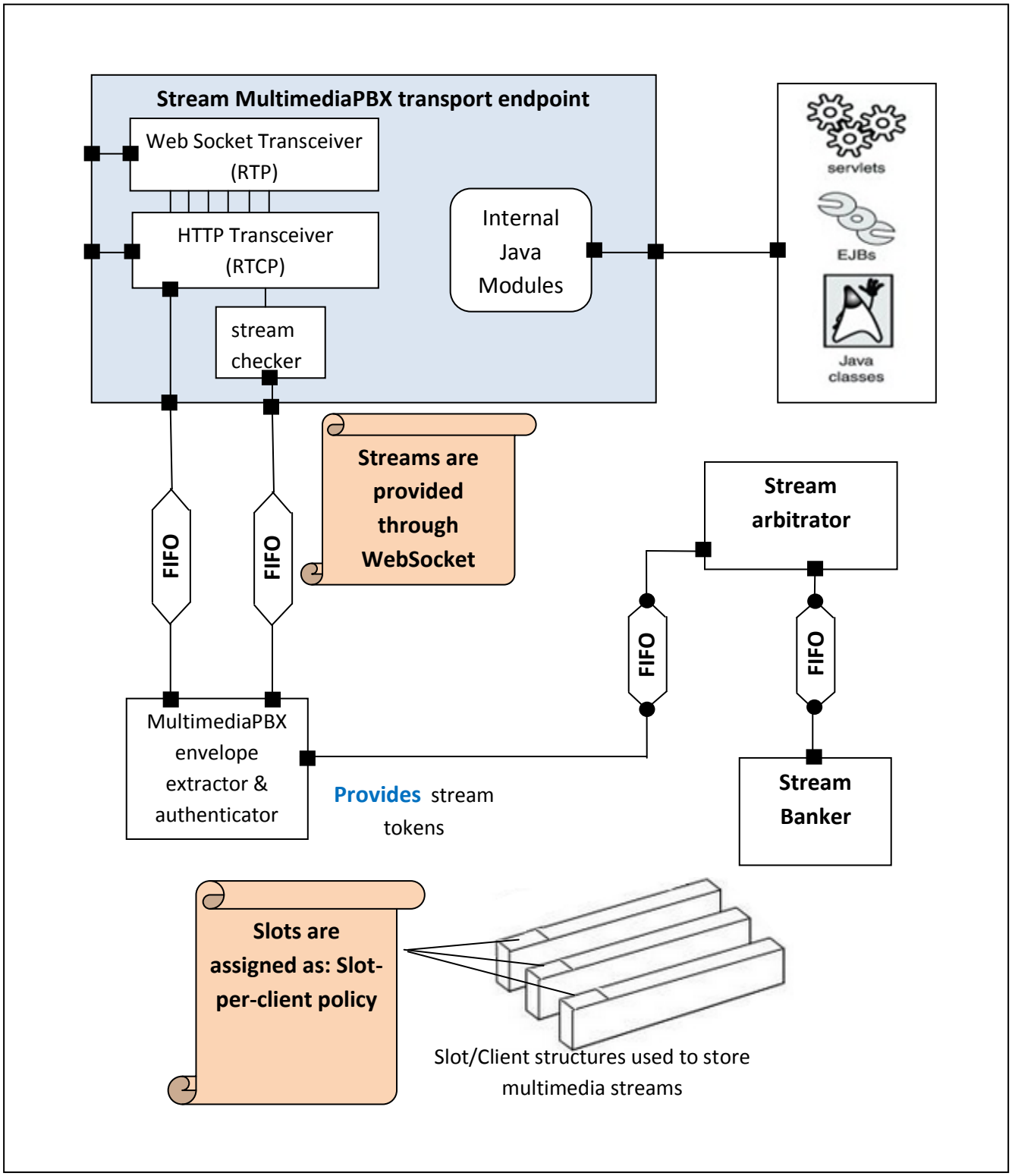

Figure 3.4: Architecture of Server Side Multimedia Streaming protocol

In the server side, multimedia server is built up basically as Java Servlet, though it is a web socket Servlet but it is essentially a Java Servlet, Algorithm-1 is the pseudo code for the creation of Web socket Servlet.

Algorithm 1: Create Web Socket Servlet

[Step 1: Instantiate Web Socket Servlet]
Public class VideoStreamServer ***extends*** *WebSocketServlet*
Begin
[Step 2: Implement essential methods of WebSocketServlet]
StreamInbound createWebSocketInbound(String string, HttpServletRequest rtcpCode)
Begin
[Step 3: Create new object of type – StreamInbound- and hand it back]
Return new InboundObject();
[Step 4: End createWebSocketInbound method]
End

Algorithm-1 is the skeletal of creating web socket Servlet but the real code is maintained by the object of type *StreamInbound*; this important object contains two virtual methods to be overridden in the implementation which are OnTextMessage() and OnBinaryMessage(); these methods represent the respond of web socket server to client requests, as it is presented in Algorithm-2:

Algorithm 2: build Stream Inbound Object

[Step 1: Instantiate new object of type – InboundObject

InboundObject extends MessageInbound

Begin

[Step 2: Respond to incoming Binary messages or streams – OnBinaryMessage()]

On Receiving Binary Message do

Begin

Process Uploading Media Stream

Update Authorization key list

End

[Step 3: Respond to incoming text messages – OnTextMessage()]

On Receiving Text Message do

Begin

Parse incoming message;

Decode incoming command;

Authorize Requests;

End

[Step 4: Send Media Streams]

If requester is authorized then

Begin

Create Stream Queue;

Enqueue Streams for Requestor;

End

[Step 5: End]

Multimedia documents (i.e., video, music… etc) are stored in the secondary memory at the server side, thus, when request is arrived to the server a local stream is opened

to the file and streamed through the web socket to the requestor. In this thesis the streamed to the requestor consecutively.

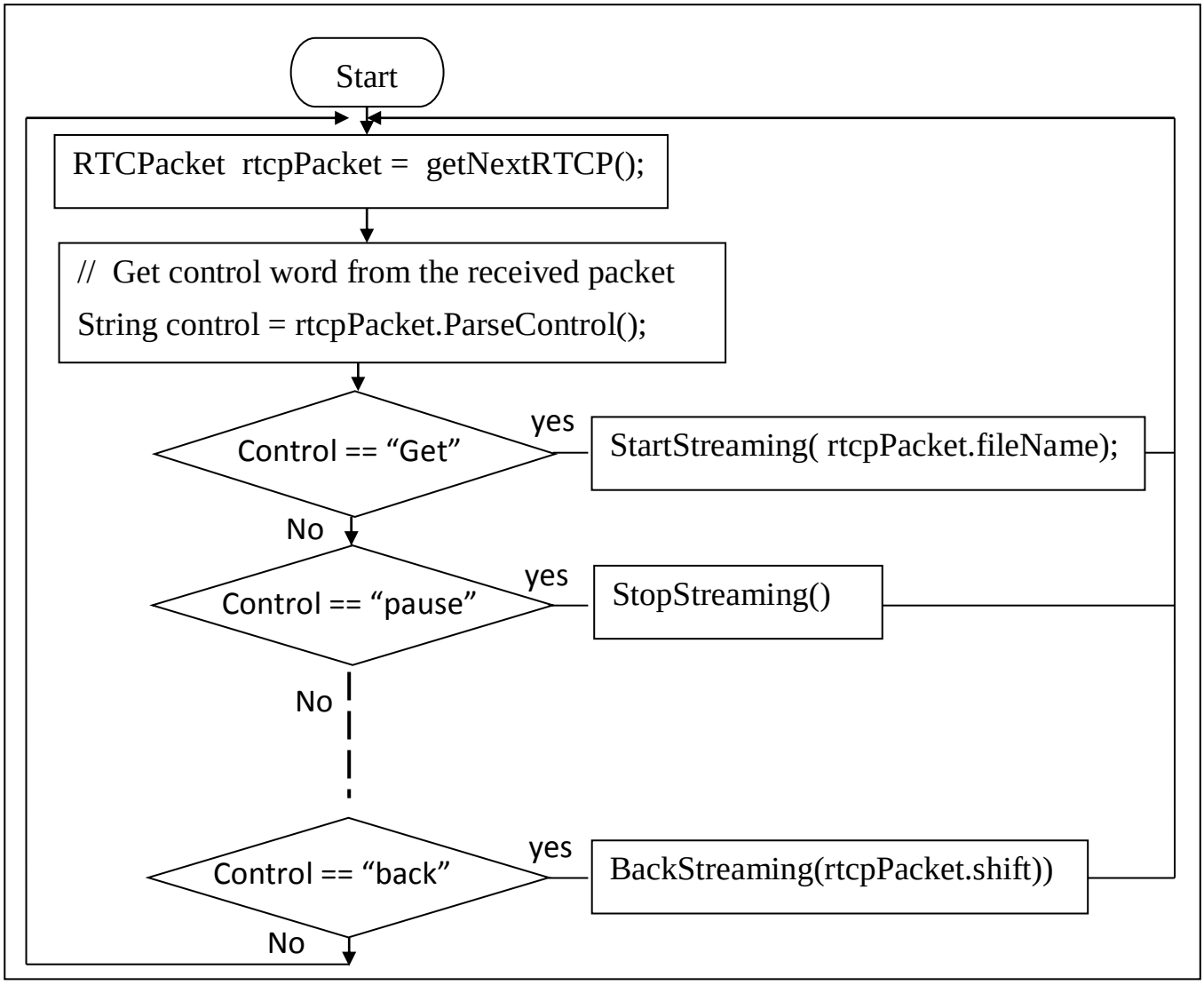

Figure 3.5: Multimedia Stream Server Responding to incoming RTCP packet

Despite the fact that most of the heavy load is introduced at the server side, client side is holding an innovative approach which is building media player based on the software technologies (i.e., HTML5 tags, Media Source API, Web Sockets and web worker) supported by the internet explorer (i.e., most candidate internet explorer is Google Chrome).Client side is composed of multiple classes of JavaScript that are

loaded as worker classes, as it was shown in figure (3.2). The essential step in the client side is to bind media source API to HTML5 video tag; this is introduced by algorithm-3 and algorithm-4.

Algorithm 3: Bind Media Source to video tag

Start JavaScript Section

[Step 1: Check Support of Media Source API by the Explorer]

If (MediaSource OR WebKitMediaSourc) is not Supported then

Begin

Alert for lacking support

Exit

End

[Step 2: Initialize Media Source]

Initialize Media Source using Algorithm-4

[Step 3: Bind HTML5 video tag to Media Source]

Get pointer to 'video' tag into → videoVar

Create URL Object and for Media Source and bind to → videoVar.source

[Step 4: Start Streaming to Video tag]

Wait for incoming stream over the web socket

End of Java Script Section

Algorithm 4: Initialize Media Source

Function Start

[Step 1: Create Media Source instance with specifications]

Var mediaSource is instance of MediaSource with

video type = 'video/webm'

codecs = 'vorbis, vp8'

mediaSource = new MediaSource('video/webm; codecs="vorbis,vp8"');

[Step 2: Create event listener and bind it to the media source]

Add event listener function to respond to → 'sourceopen'

[Step 3: Respond when even occur]

Wait for event and add process received

[Step 4: End]

Web workers communicate the front page using event driven scheme where data collected by these workers or deciphered are sent to the front page using a mechanism introduced by figure (3.6)

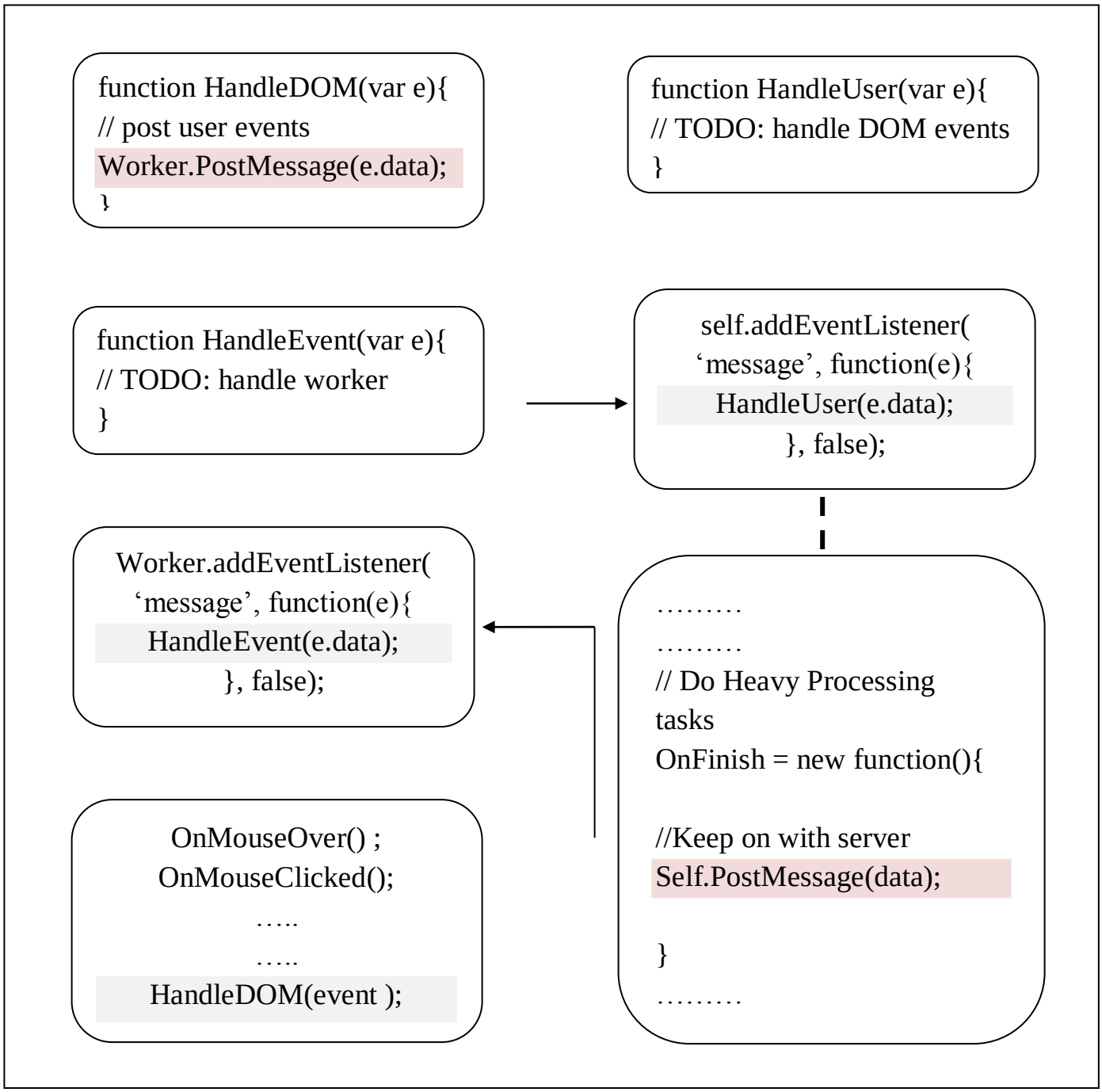

Figure 3.6: Communication Scheme between web worker and front page

Stage Two: Implementation of the Second Stage

In this stage, and after streaming '.webm' multi-media content over the internet, issues regarding uploading this type of content are managed and a special focus was dedicated to the security of the stream. This thesis work is after all about private streaming for multi-media content.

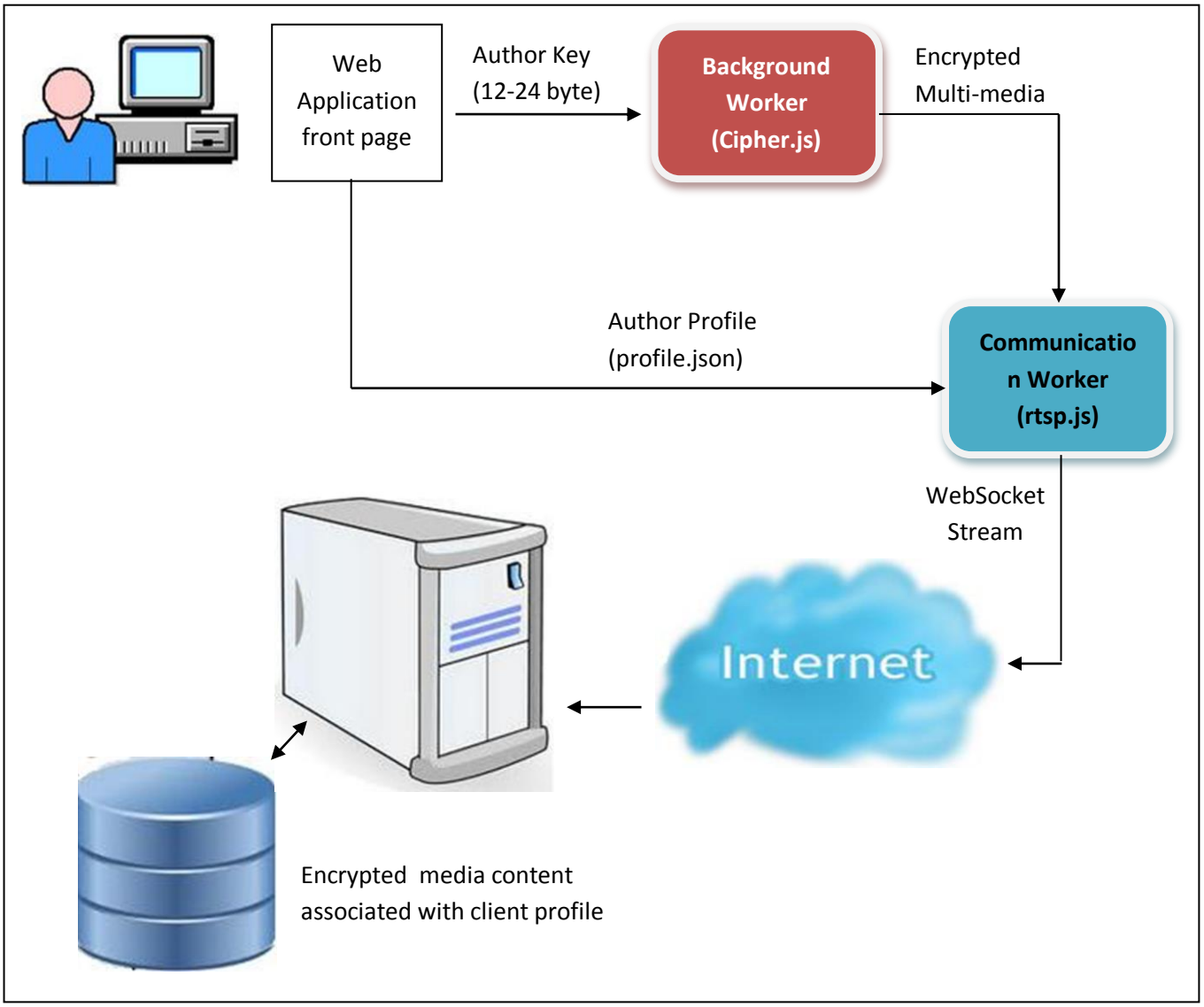

Figure 3.7: Uploading encrypted multi-media stream to stream server over the cloud

Multi-media contents are indexed using hashing system; the input to this system is the client profile but it does not include the author's key. Privacy and security are maintained as figure (3.8) presents:

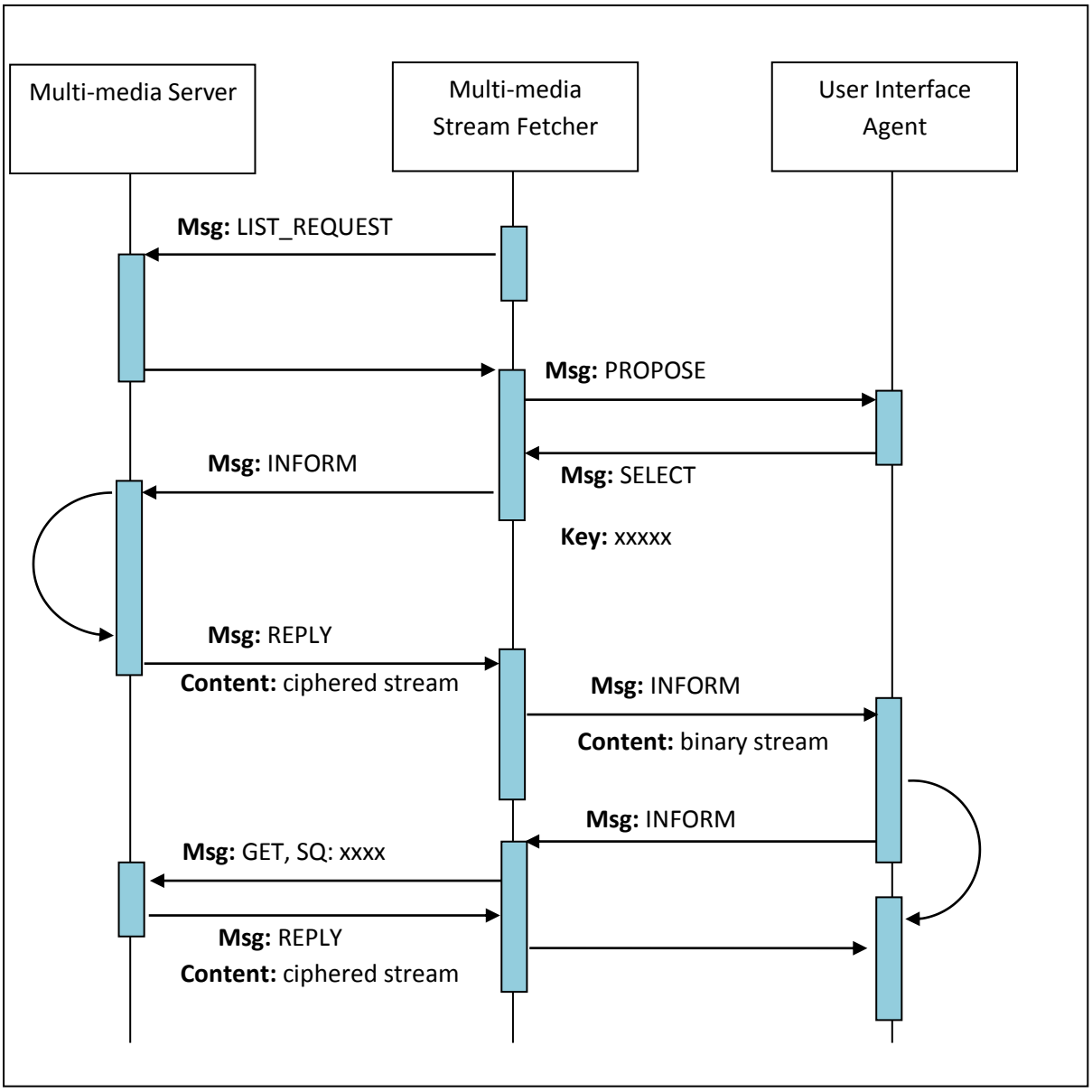

Figure3.8: Sequential Diagram of Delivering private multi-media streams

Multi-media strategy to grant multiple users access to media content without jeopardizing the privacy of the content, is performed using public key algorithm (i.e., RSA) to create secure envelop for the stream decryption key and send it over public channel.

To grant the environment more security standards, every registered user receives security token over internet on his / her personal email box; this security token is used later to retrieve the key to decipher multi-media content. Along the implementation, two communication channels have been used: the asynchronous HTTP based AJAX Requests and synchronous TCP socket oriented protocol. These two channels are used to control/manage and deliver multi- streams over the internet.

Web Worker communicate to the main web page using event driven scheme where events generated due to user interaction with the DOMs, are captured and sent to background worker module through notification message as figure (3.6) depicts.

CHAPTER 4

PRACTICAL IMPLEMENTATION, COLLECTING TESTING RESULTS AND ANALYSIS

4.1 Introduction

This chapter is dedicated to present the implementation of the proposed private interactive multimedia system where web application has been built and hosted in virtual private server. As it has been mentioned in previous chapters, the proposed system is designed to be used as web service, in other words, no many interfaces exist and the processing is conducted under the hood.

Private interactive multimedia system is a collection of java classes and JavaScript modules distributed over the server side and client side, and the streams are delivered behind the scenes, anyway, after this short introduction the chapter elaborates the front page interface which has been built using HTML5 and the web service is embedded within this page.

After presenting the front page and its main components, this chapter continues in implementing the procedure of uploading new media, getting authorization key (i.e., decryption key that is sent automatically over the internet) and starting streaming the requested multimedia contents.

Along the presentation, essential software modules are highlighted as code snippets are presented to illustrate crucial concepts in the proposed system.

4.2 Web Application Front Page

The proposed system is encapsulated in a web service, thus, it is an embeddable module and the user interface is the responsibility of the web application developer,

despite that, we have designed a web page using HTML5 to demonstrate the operability of the proposed system in all its essential functionalities.

Figure (4.1) presents an HTML5 web page with three areas and two control buttons: one for the login and the other for uploading new media file.

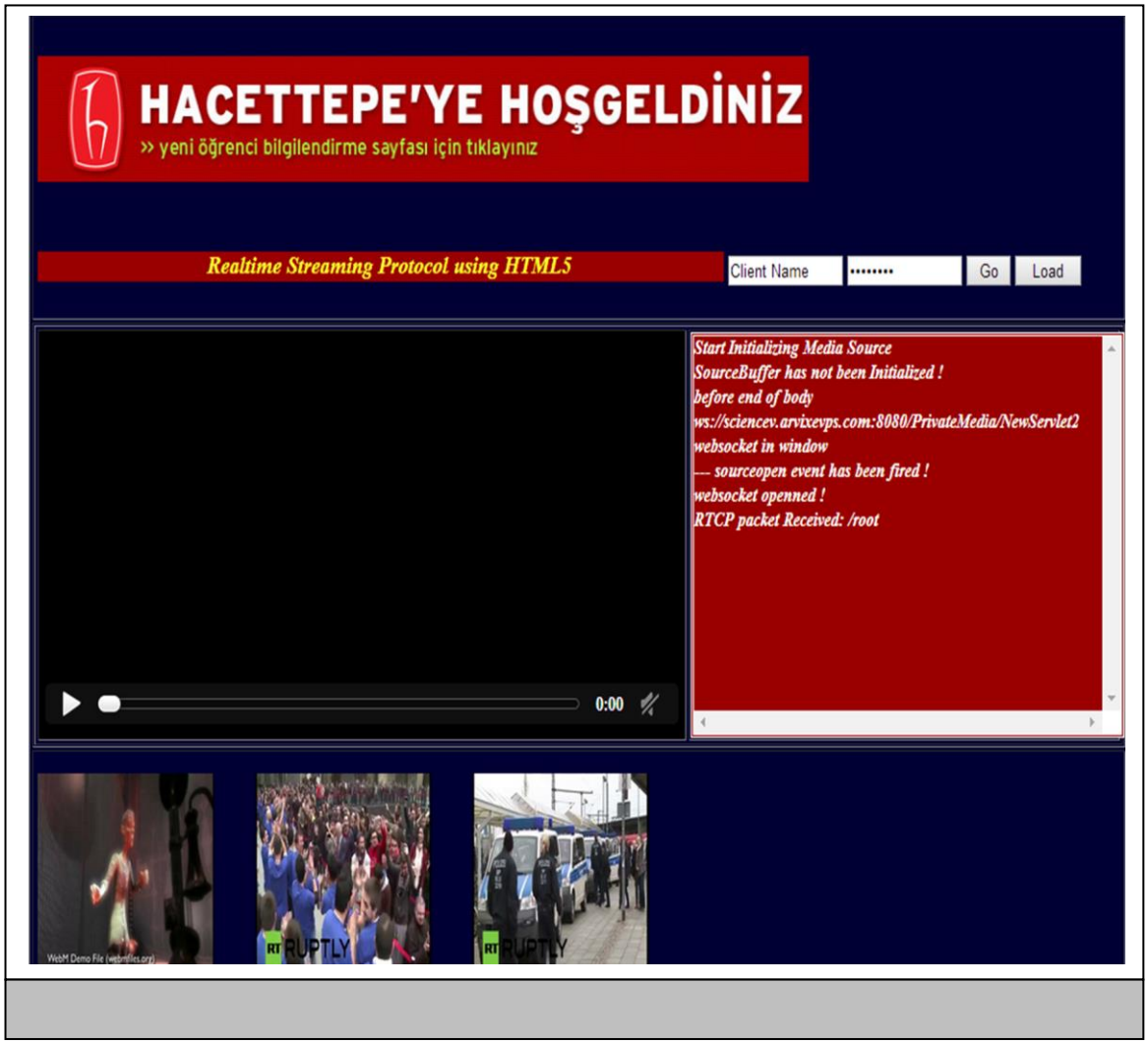

Figure 4.1: HTML5 Web Application Front Page for the Proposed

Web application starts by acquiring available multimedia files in the server database where there is an indexing for multimedia files that are stored in secondary storage.

Indexing table is manipulated using MySQL database engine. Simple query is sent to fetch some of the multimedia files randomly; fetched files are listed at the bottom of the front page as it is shown in figure (4.1).

The most important components that every web application should contain in its front page are the <video> tag; this is an HTML5 tag that accepts video streams and <audio> tag which accepts audio streams. In the proposed system which is encapsulated as web service, JavaScript code snippets are injected to bind itself to available <video> and <audio> tags as shown in figure (4.2); this is to attach the MediaSource API to these tags.

```
<script type="text/javascript">
    window.MediaSource = window.MediaSource || window.WebKitMediaSource;
if (!window.MediaSource) {
        alert('MediaSource API is not available');
          }
    log("Start Initializing Media Source");
 initMediaSource();

video = document.querySelector('video');
video.src = window.URL.createObjectURL(mediaSource);

audio = document.querySelector('audio');
audio.src = window.URL.createObjectURL(mediaSource2);

  //initMediaSource();
 if(sourceBuffer == null){log("SourceBuffer has not been Initialized !");};
 log(' before end of body ');
   connect();
</script>
```

Figure 4.2: JavaScript code Snippets to Bind <video> and <audio> tags to MediaSource

As shown in figure(4.2), two instances of MediaSource have been attached to <video> and <audio> tags; this technique is used to achieve streaming to these tags and eventually to the web application using it. Using MediaSource instances can't be done without initializing them first in separate JavaScript section as figure (4.3) presents.

```
    function initMediaSource(){

   mediaSource  = new MediaSource('video/webm; codecs="vorbis,vp8"');

 mediaSource.addEventListener('webkitsourceopen', function(e) {
 log("webkitsourceopen event has been fired !");
 sourceBuffer =    mediaSource.addSourceBuffer('video/webm; codecs="vorbis,vp8"');

});
mediaSource.addEventListener('sourceopen', function(e) {
 log("--- sourceopen event has been fired !");
 sourceBuffer =    mediaSource.addSourceBuffer('video/webm; codecs="vorbis,vp8"');

},false);
    }
```

Figure 4.3: Initializing MediaSource to be Attached to <video> tag

In figure (4.3), a MediaSource API has been initialized to be attached to <video> tag and the same sequence is used to initialize MediaSource to work with <audio> tag. The crucial element of the code presented in figure(3) is the appearance of 'sourceBuffer' variable which is a primitive data type added by HTML5 to enhance multimedia streaming.

After initializing MediaSource API and attaching it to proper HTML5 tags, the TCP connection to the server is initialiazed; this is done by relying on Web Socket protocol which is supported by all modern web browsers. Web socket protocol is used in this proposal to transfer streams by implementing custom RTP using JavaScript, while

HTTP based protocol is used to implement RTCP as mentioned earlier in previous chapters.

Figure (4.4) presents the code snippets used to initialize communication channel between client front page and the stream server. Web socket channel can be initialized over HTTP and HTTPS where ‘ws’ is a web socket protocol used over HTTP and ‘wss’ is WSS (WebSockets over SSL/TLS) used over HTTPS.

```
<script type="text/javascript">

      if(window.location.protocol == 'http:'){
            server = 'ws://'+window.location.host+'/PrivateMedia/NewServlet2';
        }
        else{
            server = 'wss://'+window.location.host+'/PrivateMedia/NewServlet2';
            }

       // try to connect based on websocket type
        if ('WebSocket' in window) {
         ws = new WebSocket(server);
         log('websocket in window ');
      } else if ('MozWebSocket' in window) {
         ws = new MozWebSocket(server);
         log('MozWebSocket in window ');
      } else {
         alert('WebSocket is not supported by this browser.');

      }
</script>
```

Figure 4.4: Initializing Web Socket Channel To Multimedia Stream Server

Figure (4.5) presents the debug and output window for messages shown on the front page which indicates proper initialization for web socket channel and SourceBuffer primitive variables, after the proper initialization an RTCP packet is sent over the internet carrying the path to the web application context in the server side; this path is essential for debug reasons as well as for the FTP (File Transfer Protocol) clients.

Figure 4.5: Debug Messages and Output Window

If the initialization process goes normal with no error messages, then we are ready to move to the next stage which is starting the streaming of multimedia contents over the internet.

The proposed system first concern is the security of the stream transferred over the internet; this security guarantees the privacy of the entire streaming session and prevents eavesdropping on the transferred stream. Many applications are available to intercept multimedia streams delivered over the internet and record intercepted streams for later playback without the need for the original resource. In this proposal, multimedia contents are encrypted before sending them over the internet which prevent the eavesdropper from making use of intercepting the encrypted stream; this is besides of the addition of another security layer by encrypting streams using a different key; this key is sent to private email box and hence decrease chances to decrypt multimedia streams.

Anyway, the process starts by selecting certain multimedia by clicking on the media icon as figure (4.1) presents, after selecting the media, a window appeared

asking for the key used to authenticate the requestor demand for this media. Figure (4.6) presents the pop up windows appeared when clicking certain multimedia content.

Figure 4.6: Pop up Window Asking for Authentication Key

In figure (4.6), the server asks for the key code which is sent to the private email address given by the user in the registration process.

The registration process is accomplished by sending email to the administrator of the web application (i.e., naeem@sciencegate2all.com); this email contains the user name and the password of the user; the administrator asks the client to validate this email by clicking on the verification link sent in an automatic reply to his/her email.

If the server program accepts the key entered by the user as in figure (4.6), then the streaming starts transfer from the server side to the client side in chunks as shown in figure (4.7), where chucks of the multimedia stream are loaded at the client side and decrypted using the key entered by the user.

Figure 4.7: Multimedia Stream Starts Coming In and Decrypted

If the user does not have decryption key, then he/she has to request it by using 'go' button after entering the user name and the password. User name and password here are used to authenticate the user but it does not provide him/her with any information about the decryption key; this is because the decryption key is sent to the private email used by the original user at the registration process

When the user clicks the 'go' button, an email is sent automatically to his/her email box containing the decryption key as shown in figure (4.8) where an email sent automatically through the administrator account (i.e., naeem@sciencegate2all.com) to the requestor.

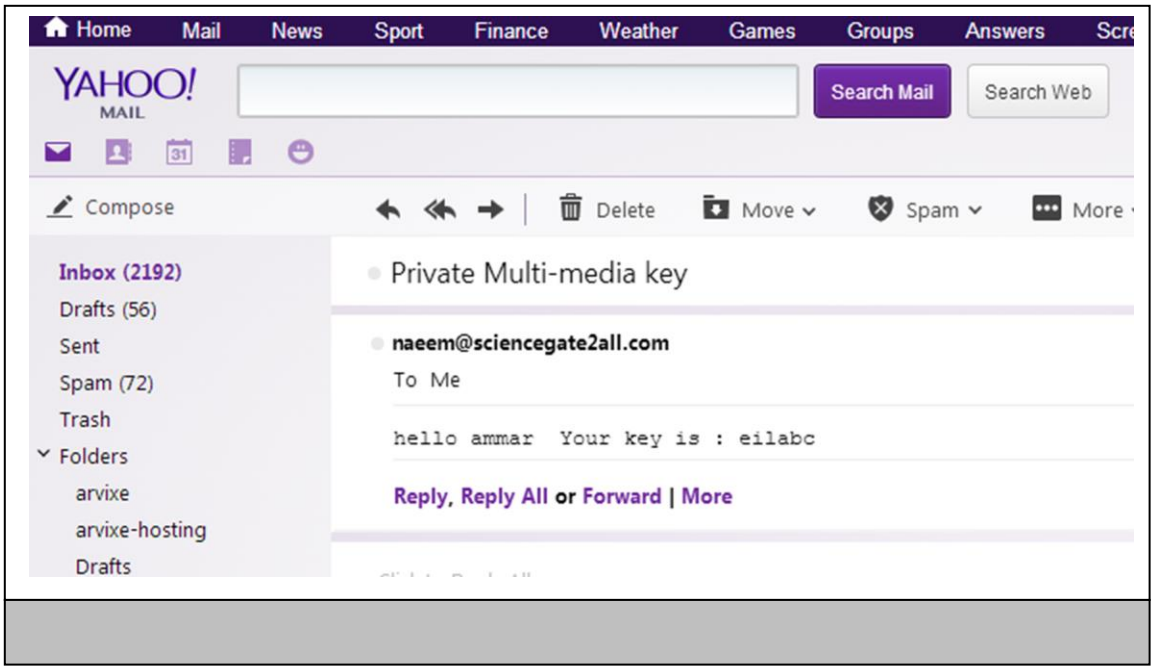

Figure 4.8: Automatic Email Sent over the Internet with the Private Key

In figure (4.8), the key is composed of 6 letters but this is just for demonstration purposes while in real application it can be expanded to include more letters, special characters and numbers.

Multimedia contents are uploaded using ‘load’ button where by clicking this button a new windows pops up as shown in figure (4.9). In this figure, information is collected about the user who uploaded the multimedia such as the email and the password; this information is used automatically for two purposes: first: to communicate the user through his/her private email and second: to authenticate multimedia contents.

Figure (4.9) illustrates the window used to upload new multimedia content which is stored on a secondary storage. The uploading is accomplished in separate web worker which provide an opportunity to stream contents in bidirectional fashion, thus, user can keep uploading large size multimedia content in the background and return to the front page for streaming in new multimedia from the server.

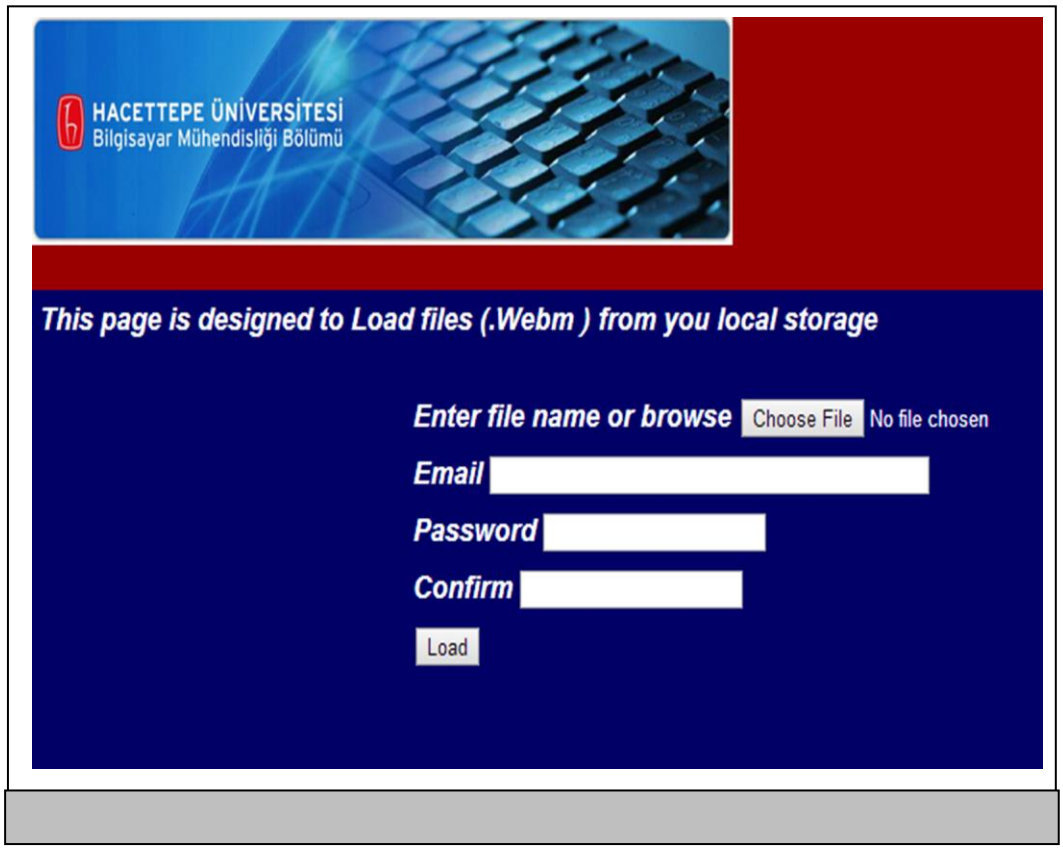

Figure 4.9: Uploading New Multimedia Contents

In this work different web worker JavaScript modules have been deployed to smooth the process of streaming multimedia content and decrypting it before presenting received content using the MediaSource. HTML5 provides efficient platform for executing pool of web worker due to the internal handling of synchronization issues and prevent those web worker from entering race competition for resources such as the DOM (Document Object Model) within the front page.

5. Conclusions

By implementing the proposed system and collecting results many conclusions have been reached and many obstacles are addressed, we will introduce theses conclusions and obstacles briefly in this section, also some challenges are specified for further investigations in the future efforts:

1- HTML5 introduces excellent programming potential to build complete solutions where advanced programming utilities such as multi-threading, primitive data types to hold binaries and conducting TCP based communication sessions; this is in term of developing web based applications but we faced many issues in the stability of the built program with different internet browsers especially Firefox where the same web application has been collapsed many times; this due to the modifications in strategies of manipulating binary data.

2- Security has been maintained efficiently when web applications do not need installing third party software modules (i.e., plug ins and ActiveX). Flash players and other third party video players count on installing more modules with the binary libraries of internet explorers (i.e., Internet Explorer, Firefox and Google Chrome); this would compromise security protection shells at two levels: the network level and the system level, where installing new software modules grant attackers an opportunity to spoof original these modules and inject malicious code and this would eventually compromise the security at a level beyond the capabilities of internet security software or anti-virus.

3- Node.js has shown a great superiority in comparison to other web servers (i.e., TomCat, GlassFish and other web servers) where node.js is easily can be exploit to build http server or web servers with server side javascript. The only issue that faced employing it massively in developing web applications is lack of documentation for its internal structure despite the fact that many modules are produced and published in Git Hub. Node.js has many issues when the integration with TomCat is needed where in many cases TomCat provides great functionalities especially in session management and the huge libraries

designed to work with. A configuration file has to be manipulated manually which needs intensive knowledge about the components of the configuration file and their meanings.

4- HTML5 Web Worker can survive the competition of creating browser built in media player. Streams have been transferred smoothly to web worker and forwarded internally to main page instance. It is not yet clear if objects are serialized or streamed from worker to the main page but data blocks have been transferred with high bit-rate.

5- Web worker still lack the multi-threading policies, yet it provides awesome background task burning by efficient utilizing of multi-core CPUs. Security context for calling Web Worker is also missing and this can introduce many critical problems in maintain stability especially in initiating pool of threads. Anyway, accessing DOM objects within the main HTML page still manipulated in single threaded fashion which prevent streaming multiple streams at the same time; this is a very critical issue due to the need for multiple streaming capabilities with the increasing demands for the market to unify multimedia channels (i.e., TV channels, Radio and Internet).

6- For video streaming issue, the perfect scenario for data transferring between Web Worker and the main page is by transferring ownership due to the zero-copy style of transferring, the only drawback of using this style is losing access of the main page to the transferred object.

7- Implementing security algorithm as a script is not the more efficient idea due to the readability of JavaScript but it can perform as the front edge of a security algorithm implemented in native code. JavaScript runs in a very restricted context that does not allow scripts to access any lower level code or libraries, thus, the best approach is to provide mobile intelligent agent within the client browser, and this mobile agent will handle the security of the communication session.

6. Future Work

As a future work for the implemented work in this dissertation is to maintain the security issues of the deployed web service, where candidate web service for streaming multimedia should authenticate streams exchanged over the cloud. In this manner streams can be counterfeited along the transmission and spoofed. In our implemented proposal the security issues are maintained using simple authentication policy but it is not enough to preserve the privacy.

Another approach for extending proposed system is to implement secure caching system to promote the performance of multimedia streaming web services that uses streaming gateway as it is presented in this dissertation.

REFERENCES

[1.] Pelin Aksoy and Laura DeNardis, Information Technology in Theory, Course Technology, *Division of Thomson Learning inc*., **2008**.

[2.] Johnny Ryan, *A History of The Internet and The Digital Future*, Johnny Ryan, USA, **2010**.

[3.] Janet Abbate, Inventing the Internet, *MIT Press paperback edition*, **2000**.

[4.] Brian Winston, *Media Technology and Society: A History From the Telegraph to the Internet*, Brian Winston, **2001**.

[5.] Jim Shuman, Callen Coorough, and James E. Shuman, Multimedia for the Web: Revealed: Creating Digital Excitement, *Thomson/Course Technology*, **2006**.

[6.] Borko Furht and Armando Escalante, Handbook of Cloud Computing, *Springer Science Business Media*, LLC, **2010**.

[7.] Michael Miller, Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online, *Que Publishing*, **2009**.

[8.] YouTube – YouTube HTML5 Video Player [WWW]. Accessed over the web on http://www.youtube.com/html5 *,* **2010**

[9.] Ramesh.B, Savitha.N, and Manjunath.A.E, Mobile Applications in Multimedia Cloud Computing, *International Computer Technology & Applications*, Vol 4, **2003**.

[10.] Kim Paananen, *Comparison of Flash, Silverlight and HTML5 technics*, Tampere University of Tecnolgoy, **2011**

[11.] Sebastian Jeuk, Jakub Szefer , and Shi Zhou, Towards Cloud, Service and Tenant Classification For Cloud Computing, *Proceedings of the Workshop on Assured Cloud Computing (WACC)*, May,**2014**.

[12.] Jim Summers, Tim Brecht, Derek Eager, and Bernard Wong, To Chunk or Not to Chunk: Implications for HTTP Streaming Video Server Performance, *22nd SIGM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Toronto,Canada,pp.15-20, June, **2012**.

[13.] Arun Kumar B. R., Lokanatha C. Reddy, Prakash S. Hiremath, Naresh.S.S, RTSP Audio and Video Streaming for QoS in Wireless Mobile Devices, *IJCSNS International Journal of Computer Science and Network Security*, VOL8 No.1, **2008**.

[14.] Jani Peltotalo, Jarmo Harju, Lassi Vaatamoinen, Imed Bouazizi, and Igor D. D. Curcio, RTSP-based Mobile Peer-to-Peer Streaming System, *International Journal of Digital Multimedia Broadcasting*, VOL **2010**.

[15.] J. Peltotalo, J. Harju, A. Jantunen, et al., Peer-to-peer streaming technology survey, *In Proceedings of the 7th International Conference on Networking (ICN '08)*, pp. 342–350, April **2008**.

[16.] Ranjan Parekh, Principles of Multimedia, *Tata McGraw-Hill*, USA, **2013**.

[17.] P. Shah and J.-F. Paris, Peer-to-peer multimedia streaming using BitTorrent,*In Proceedings of the 26th IEEE International Performance, Computing, and Communications Conference (IPCC '07)*, pp. 340–347, April **2007**.

[18.] P. Shah and J.-F. Paris, Peer-to-peer multimedia streaming using BitTorrent, *In Proceedings of the 26th IEEE International Performance, Computing, and Communications Conference (IPCC '07)*, pp. 340–347, April **2007**.

[19.] Jenq-Neng Hwang, Multimedia Networking from Theory to Practice, *Cambridge University Press*, **2009**.

[20.] W3C, Extensible Markup Language (XML) 1.0, *World Wide Web Consortium (W3C)*, 4th edition, **2006**

[21.] Altiparmak, N. ,Tekeoglu, and A. Tosun , A.S., DoS Resilience of Real time Streaming Protocol, *IEEE, Performance Computing and Communication Conference (IPCCC)*, **2011**

[22.] Alexander Kolesnikov, *Enterprise Application Development Using Jakarta Tapestry*, MSc. Thesis, Glasgow Caledonian, **2006**.

[23.] Nash, M., Java Frameworks and Components. *Cambridge University Press*, **2003**.

[24.] NursuriatiJamil and SitiAisyahSa'adan, Proceeding, *Visual Informatics: Bridging Research and Practice*, *Visual informatics conference*, IVIC 2009, Kuala Lumpur, Malaysia, **2009**.

[25.] Sandeep Chatterjee and James Webber, Developing Enterprise Web Services: An Architect's Guide, *Prentic Hall PTR*, USA, **2004**.

[26.] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju, Web Services: Concepts, Architectures and Applications, *Springer-Verlag Berlin Heidelberg*, USA, **2004.**

[27.] Alex Duong nghiem, IT Web Services: A Roadmap for the Enterprise, *Prentice Hall PTR*, **2003**.

CURRICULUM VITAE

Credentials:

Name, Surname: Naeem Th. Yousir

Place of birth: Baghdad

Marital status: married.

E-mail: naeemms@yahoo.com

Address: Iraq- Baghdad – palistine st. sq

Education:

High school: Aljazeera for boys

BSc. : Statistics and computer science / Rafidain University collage.

MSc. : Iraqi committee for computer and informatics / institute of higher education

PhD: Hacettepe University- Computer Engineering dept.

Foreign Language:

English:

Arabic:

Turkish:

Work Experiences:

- Teaching at the Institute for Studies and Research / Iraqi Commission for Computers and Informatics 2000.
- Member of the Iraqi Association for Computer Science 2001
- Lecturer in institution of administration / rasafa 2001 in data base subject.
- Rapporteur of the Board of Directors of the National Center for electronic computers 2002.
- Director of the department administrative and legal in the Iraqi Commission for Computers 2003.
- 10 years of experience in computer maintenance.
- Lecturer in Al- Mansour University College 2004/2005 in internet technology.
- Representative Ministry of Higher Education and Scientific Research In e-government 2003.
- Lecturer in Al- Mansour University College 2006/2007 in software engineering and multimedia.
- Consultant President of the Iraqi Commission for Computers and Informatics 2006/2007.
- Teaching AL- Nahrain University- College of Information Engineering 2007-

Area of Experience :

Projects and Budgets:

Publications:

Oral and Poster Presentation