

**Multi-Protocol Video on Demand System for Distance
Education
with Pedagogical Enhancements**

by

Cem KARACA

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements
for the Degree of

**Master of Science
in
Electrical and Electronic Engineering**

Eastern Mediterranean University
January 2004

Approval of the Institute of Graduate Studies and Research

Assoc. Prof. Dr. Osman YILMAZ
Director

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

Assoc. Prof. Dr. Derviş Z. DENİZ
Chairman

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

Assoc. Prof. Dr. Derviş Z. DENİZ
Supervisor

Examining Committee

-
1. Prof. Dr. Seyit G. GÜLER_____
 2. Assoc. Prof. Dr. Derviş Z. DENİZ_____
 3. _____

CONTENTS

ABSTRACT	v
ÖZET	vii
Acknowledgements	ix
Dedication	x
List of Tables	xi
List of Figures	xii
1 INTRODUCTION	1
2 VOD SYSTEMS AND DISTANCE EDUCATION	7
2.1 Types of Video on Demand Systems	7
2.1.1 True Video on Demand.....	10
2.1.2 Near Video on Demand	14
2.2 Quality of Service in VoD Systems	15
2.2.1 Network Infrastructure	15
2.2.2 Data Compression Methods	17
2.2.3 Mass Storage and Video Scheduling Algorithms	20
2.3 Distance Education	21
3 CURRENT TECHNOLOGIES AND APIS FOR VOD SYSTEMS	24
3.1 C/C++, Object Oriented Programming and Classes	25
3.2 Microsoft Foundation Classes	26
3.3 Component Object Model (COM)	27
3.4 Microsoft DirectX	30
3.5 ActiveX Controls	31
3.6 MySQL C API	33
3.7 Windows Sockets (Winsock) API.....	34
3.8 Win32 Internet Services (Winlnet) API	35
3.9 Globally Unique Identifiers	37
3.10 Cyclic Redundancy Check (CRC)	37
4 DESIGN AND ANALYSIS OF THE VOD SYSTEM	39
4.1 VoD System Configurations	42
4.2 The Organization of the VoD Client-Server System.....	44
4.3 VoD Servers.....	49
4.3.1 Mediation Server.....	50
4.3.2 Content Servers.....	53
4.3.2.1 Implementation of the Content Server Applications	55
4.4 The VoD Clients	63
4.4.1 Implementation of the VoD Player	64
4.4.2 System Model of the VoD Player.....	70
4.5 System Characteristics	74
4.5.1 Streaming, Buffering and Memory Mapped Files	74

4.5.2	Smooth Buffering	77
5	PEDAGOGICAL ENHANCEMENTS FOR THE VOD SYSTEM	81
5.1	The Project Editor	85
5.1.1	Project Editing Tree.....	87
5.1.2	Synchronization of HTML Pages to Video.....	87
5.1.3	WYSIWYG DHTML Editor	88
5.1.4	Synchronization Data Structure	90
5.2	Pedagogically Improved VoD Player	92
6	MULTI-PROTOCOL ENHANCEMENTS	94
6.1	VAP and VoD System in OSI Reference Model	94
6.2	Transport Control Protocol (TCP).....	97
6.2.1	VAP commands for handling TCP communications	98
6.3	File Transfer Protocol (FTP)	99
6.4	User Datagram Protocol (UDP)	100
6.4.1	Slow Start Algorithm for UDP.....	100
6.5	Reliable UDP.....	102
6.5.1	RDUP Header and Data Format.....	103
6.5.2	VAP commands for handling RDUP and UDP Communications.....	105
6.6	ATM LAN Emulation	106
7	SYSTEM PERFORMANCE MEASUREMENTS	107
7.1	Network Protocols Performance Tests	107
7.1.1	VoD Network Performance Results	110
7.2	Download Speed and Playing Speed Comparisons	111
8	CONCLUSION AND FURTHER WORK.....	113
8.1	Conclusion	113
8.2	Further Work	116

ABSTRACT

MULTI-PROTOCOL VIDEO ON DEMAND SYSTEM FOR DISTANCE EDUCATION WITH PEDAGOGICAL ENHANCEMENTS

Cem KARACA

M.Sc., Electrical and Electronic Engineering

Supervisor: Assoc. Prof. Dr. Derviş Z. DENİZ

January 2004

The revolution in the networking technologies has increased the importance of Distance Education (DE) applied through computer systems. The approach to computerized DE is to present the DE material as Web pages or as other presentation forms through the World Wide Web or CD-ROMs enriched with video clips where necessary for the demonstration of the text based DL material. In this thesis, the approach taken is the reverse. It is envisaged that the pedagogically enhanced video can be a very powerful learning environment, when the video session covers the actual presentation while the video is supported with “html” based content related materials.

Video on Demand (VoD) refers to the instant delivery of the videos from a distance through the network on user requests while carrying out the capabilities of the VCR systems such as, pausing, fast forwarding or reversing the video. Hence, VoD is a suitable infrastructure for implementing on-demand Asynchronous Distance Education (ADE) in the absence of a guiding online instructor.

This dissertation describes the design of a VoD system for the fulfillment of the requirements of a complete ADE system. The main objective is the provision of a learning environment that will allow the traditional VOD systems to include “html” based pages as additional clarifications on the subject material. One of the main features of the system is the provision of built-in editing software for adding and synchronizing information in the form of “html” pages, which may include pre-requisite or more advanced topics necessary or useful for the understanding of the concepts presented in the video. Hence, these topics in the video can be further explained. A further feature is that, learner can also add additional content to the system provided material. Course based discussion is allowed through a Learner Discussion forum which is implemented as an additional service on the Content Servers.

The designed VoD system is based on the distributed servers network architecture in order to distribute the network load. A light, efficient and fast transport protocol Reliable UDP for video transmissions is also designed for this dissertation.

It has been found that the distributed architecture is an efficient platform for VoD content delivery as well as pedagogical enhancements and shown that a practical Computer Based Learning environment can be implemented.

Keywords: Video on Demand, Enhanced Distance Learning, Multi Protocol, Distributed Servers Architecture, Internet Based Distance Education, Reliable UDP.

ÖZET

UZAKTAN EĞİTİM İÇİN PEDAGOJİK OLARAK GELİŞTİRİLMİŞ TALEPLİ VİDEO SİSTEMİ

Cem KARACA

Elektrik ve Elektronik Mühendisliği Yüksek Lisans

Tez Danışmanı: Doç. Dr. Derviş Z. DENİZ

Ocak, 2004

Acknowledgements

I would like to express my sincerely appreciation to my advisor, Assoc. Prof. Dr. Derviş Z. DENİZ, who has been a guide for me and he shown the way of “know how” as more than just being a developer, also for orienting me to work in the area of Distance Education, multimedia and network communications.

I would like to thank to BROMCOM PLC for supporting me with the scholarship for my masters education and to Information Technologies R&D Center (INTERDEC) which is under the supervision of Assoc. Prof. Dr. Derviş Z. DENİZ for the computing and networking facilities dedicated for my studies.

I am very grateful to all INTERDEC members for their brotherhood and especially to Emre ÖKTEM to be the INTERDEC mate for me while we spent our months for studying.

I would like to thank to my parents Gönül KARACA, Faruk KARACA and my elder sister Özlem GÖKTAN for their love, support, patience and encouragement.

Most special thanks to my wife Bestem who deserve much more attention than I could devote to her during this study and for keep loving me.

Finally, I want to thank God, who allowed me to left a written book on earth.

Dedicated to my wife, Bestem

List of Tables

Table 2.1 Classification of VoD systems 10

Table 2.2 Comparison of Unicast and Multicast transmission..... 12

Table 4.1 VoD Client-Server applications 42

Table 4.2 Comparison chart for client side applications..... 64

Table 4.3 Parameters used to calculate initial buffer 78

Table 5.1 Available DHTML commands in the VoD Project Editor..... 90

Table 5.2 The html Page Structure 91

Table 5.3 Synchronization Data 91

Table 5.4 A sample synchronization data 92

Table 6.1 Basic comparison chart for the protocols 97

Table 6.2 VAP internal commands for TCP 98

Table 6.3 UDP Header format..... 100

Table 6.4 RUDP Acknowledge data format..... 104

Table 6.5 RUDP Bulk data format..... 104

Table 6.6 VoD internal commands for TCP..... 105

Table 7.1 Download times in milliseconds 110

List of Figures

Figure 2-1 Architecture of Video on Demand system on computer networks.....	7
Figure 2-2 Architecture of Video on Demand system on TV networks.....	8
Figure 2-3 Working principles of commercial media players.....	11
Figure 2-4 Channels in NVoD System	14
Figure 2-5 TCP one pipe vs. multi-pipe transmission.....	17
Figure 3-1 A typical logic circuit.	28
Figure 3-2 Sample Component Object Model used in DirectX.....	28
Figure 3-3 DirectX hardware interfacing.....	30
Figure 3-4 VoD Player ActiveX pane showing an Acrobat document	32
Figure 3-5 VoD Project Editor ActiveX pane DHTML editing facility	33
Figure 3-6 WinInet flowchart for FTP operations.....	36
Figure 4-1 VoD Client-Server Architecture combining the VoD System.....	40
Figure 4-2 VoD Project Editor Configuration.....	43
Figure 4-3 VoD Web services configuration.....	43
Figure 4-4 VoD DE configuration	44
Figure 4-5 Client - Server communication.....	44
Figure 4-6 VoD Player - VoD System network negotiation	45
Figure 4-7 VoD Editor – VoD System negotiation (edit mode).....	46
Figure 4-8 VoD Editor – VoD System negotiation	47
Figure 4-9 Typical VoD Server distribution.....	50
Figure 4-10 VoD Database connection handling.....	51
Figure 4-11 VoD relational database organization.....	52
Figure 4-12 VoD Content Database structure.....	53
Figure 4-13 VoD Content Database connection handling.....	54
Figure 4-14 Content Server application's main threads	57
Figure 4-15 UDP Server message redirection system	58
Figure 4-16 UDP and RUDP service thread.....	59
Figure 4-17 TCP listener module	60
Figure 4-18 TCP processor module	61
Figure 4-19 VoD TCP Server.....	62
Figure 4-20 VoD UDP Server.....	63
Figure 4-21 VoD Player authentication screen.....	65
Figure 4-22 VoD Player main window.....	65
Figure 4-23 Project selection window	66
Figure 4-24 Content material download by the FTP process.	67
Figure 4-25 VoD Player Video and html browser screen	68
Figure 4-26 User contribution drag and drop box.....	69
Figure 4-27 Client application diagram	70

Figure 4-28 Main flowchart for the VoD Player	71
Figure 4-29 Database Module.....	72
Figure 4-30 Project Selection Module	72
Figure 4-31 Network Connection Module.....	72
Figure 4-32 Buffering and playing steps of VoD Player	76
Figure 4-33 Buffering and Playback Processes	77
Figure 4-34 Smooth Buffering strategy	78
Figure 5-1 Pedagogical enhancements on client side, VoD Player video slider and “Pause and Drag” windows.....	82
Figure 5-2 VoD Project Editor	86
Figure 5-3 Schematic showing the VoD Project Tree	87
Figure 5-4 Html to video synchronization.....	88
Figure 5-5 a- DHTML viewer, b- DHTML editor, c- Source code editor.....	89
Figure 5-6 Enhanced VoD Player	93
Figure 6-1 Inputs and the outputs to layers of OSI-RM by VoD system.....	96
Figure 6-2 Speed selector dialog box	101
Figure 6-3 Slow start algorithm flowchart.....	102
Figure 6-4 RUDP Speed/Reliability comparisons.....	103
Figure 7-1 Campus Wide Network as a test configuration	108
Figure 7-2 Download speeds of the protocols.....	111
Figure 7-3 Buffering vs. Playback $S_p < S_d$	112
Figure 7-4 Buffering vs. Playback $S_p > S_d$	112

1 INTRODUCTION

The increasing speed in computer networks and better compression ratios in multimedia resources make the internet and intranet more suitable for multimedia applications such as Video on Demand (VoD), internet telephony and video conferencing. VoD will find its place in the market applied to many fields. Distance Education is one of the fields where VoD is going to make an important impact. This thesis investigates the technology and use of VoD systems for Distance Education. The approach includes design and development of pedagogical enhancements for Distance Education using True Video on Demand (TVoD) systems.

Distance Education is the representation of the instructional material to the student in the absence of the instructor or in the conditions where instructor and the student cannot be in the same place or are separated in time. Distance Education can be separated into two parts:

- Synchronous education.
- Asynchronous education.

The proper application layer for synchronous education is to use video conferencing in order to allow interaction between the student and the instructor. On the other hand, a system including both streaming video and a white board system could also facilitate interaction between the instructor and the student. These types of systems must be served in a multicast manner and must be scheduled for a certain time

in order to reduce the number of online teaching staff and increase the number of online students. Since we think of these types of systems as multi-user type or in the terms of networking; multicasting, the online video system will not have the VCR like facilities, like pause, forward, reverse and stop. The student must watch the entire video before asking any questions to the online instructor or before writing to the white board system. A system must adapt it self for the worst case situations, in this manner a Multicast Video on Demand (MVoD) system must adapt itself for the slowest connection in the range of all connections that are currently using the MVoD system. LAN connection speed nowadays is not less than 10 Mbps but, the connections may vary from an ISDN line to a LAN connection in a group of online clients in an MVoD system, and a LAN user could also face fragmentation on multicast video because the system adapts itself to the ISDN line speed.

On the other hand, asynchronous education could be implemented using the True Video on Demand (TVoD) or Unicast Video on Demand (UVoD) approaches. This system is mostly the reverse of the video conferencing system; there is no time restriction, the students are able to play the video at any time and the system is interruptible in a manner in which the video could be paused or stopped at any time. Also no time scheduling is required between the server and the client since the system serves static videos on client requests. Since there is no time restriction in this system, the existence of the online instructor is not of concern.

In order to make the VoD system behave as near “interactive” as possibly i.e. similar to the synchronous distance education systems, the VoD system prepared, designed and presented in this thesis is able to show supporting course material as html (hyper text markup language) pages in its “embedded” html browser. A VoD session is

better called a “VoD Presentation” and it incorporates combination of video, voice and other content related material. The system also includes an interactive design studio for VoD presentations that is called “VoD Project Editor”. This editor is capable of creating, editing and synchronization of html pages with the related video segments. The number of html pages that should be attached to a VoD project is not limited, but download times of the additional content material must be taken into account.

VoD system designed for this thesis is based on a “restriction-free” concept in order to make a user friendly application environment. To obey this concept the system is not restricted to specific media formats and specific network connection protocols. In other words, the VoD Player is capable of presenting any video type using different transport protocols which include UDP, TCP, FTP (over TCP) and Reliable UDP (RUDP) where reliability comes from its “connection oriented” structure, implemented with the latest technology components such as Microsoft DirectX to present all types of videos, Microsoft Internet Explorer Server is integrated into the system in order to serve related presentation material provided as html pages and NTFS (Windows NT File System) file system to collect and hide many streams under one file name.

Database connectivity of the system is also designed to authenticate the users, track the user activities and to store VoD presentation data. For this purpose the MySQL database system is used in the backend. This kind of system implementation is described in Frank Rousseau’s [1999] paper on “An advanced multimedia infrastructure for WWW-based information systems”. In the proposed server system architecture, the VoD database and media servers are physically separated and called “Mediation Server” and “Content Server” respectively, but integrating both into one server machine is also possible. The Mediation Server is responsible from user authentications, storing

server addresses and VoD Project databases and the Content Server stores the multimedia content used in the VoD Projects. To administer and track user activities, WWW based administration and statistics gathering system is also included. The complete specification of the pedagogically enhanced asynchronous distance education VoD system is:

1. Support up to 250 simultaneous users, (depends on the selected video bit streams and server hardware configuration)
2. Support of VCR-like facilities, i.e. Play, Pause, Stop, Rewind,
3. Includes Project Editor system for adding and editing video clips with related content material, also related material can be edited using the built-in html editor,
4. Uses suitable stack of protocols on unicast basis, which includes TCP, UDP, FTP and RUDP,
5. WWW enabled supervisor console to maintain databases,
6. User database for presentation authorization,
7. Embedded html based content material server,
8. Capable of playing all types of media formats (mp3, AVI, mpeg), i.e. if the related video codec is present in the client machine,
9. Individual server side applications for UDP, TCP and FTP; for FTP, third party deamons could be used,
10. Multi-threaded application environment to prevent hang outs when streaming (playing the video while downloading) the video.

This dissertation is separated into eight chapters. The detailed explanation of synchronous, asynchronous systems and the taxonomy of VoD systems are presented in Chapter two. Also previous implementations of the VoD systems are investigated and

the cost-benefit analyses of the other connection protocols used in VoD systems are described.

Chapter four describes the client-server structure of the VoD system. VoD Mediation Server infrastructure, VoD Content Server applications and content media allocation systems are explained in the first part called “Server-side” of Chapter four. VoD Player and VoD Project Editor are presented in the second part called “Client-side” of Chapter four. The buffering strategies and the initial buffering time measurements are also implemented in presented Chapter 4.

The related content material synchronization with the presentation video in VoD Project Editor, WWW user authentication and user statistics manipulation are presented in Chapter five which is titled as “Pedagogical Enhancements”.

In Chapter six, the multi-protocol enhancements on the VoD system and the effects of network protocols on video streams are studied. VoD Player is able to stream presentations using various transport protocols by connecting to server-side VoD UDP or TCP servers, currently limited to the Win32 platforms. In order to make the VoD player independent of the server-side platform projects can be transferred from an FTP server assembled on the *nix (UNIX, SCO, Linux, FreeBSD) systems.

In Chapter seven performance measurements of the various transport protocols used in the system are presented. The main performance indices are;

1. Download speeds,
2. Reliabilities and,
3. System resource allocations.

One of the disadvantages of the True Video on Demand systems is the cost of the system in resource allocations. TVoD system makes full buffering of the entire video and the content material in order to reduce the network traffic and to enable VCR-like facilities.

This work presents more than one way to deal with the buffering and streaming of the videos in scientific manner and also pedagogical enhancements gained by this system on distance education in social manner. All these results and future works will be described and concluded in chapter eight.

2 VOD SYSTEMS AND DISTANCE EDUCATION

2.1 Types of Video on Demand Systems

Video on Demand refers to instant delivery of video streams over internet, local area network or TV broadcasting immediately on user or subscriber request. Figure 2-1 represents basic VoD architecture on computer networks. In the architecture shown in the figure, the content server stores media files to be presented including video, sound and text, mediation server is used to store user related data including authorization, statistics and validation services. The development on VoD systems could be separated into two categories; TV broadcasting systems and computer aided multimedia systems. Except TVoD and MVoD (Multicast Video on Demand) systems, the other types are mostly used on TV broadcasting.

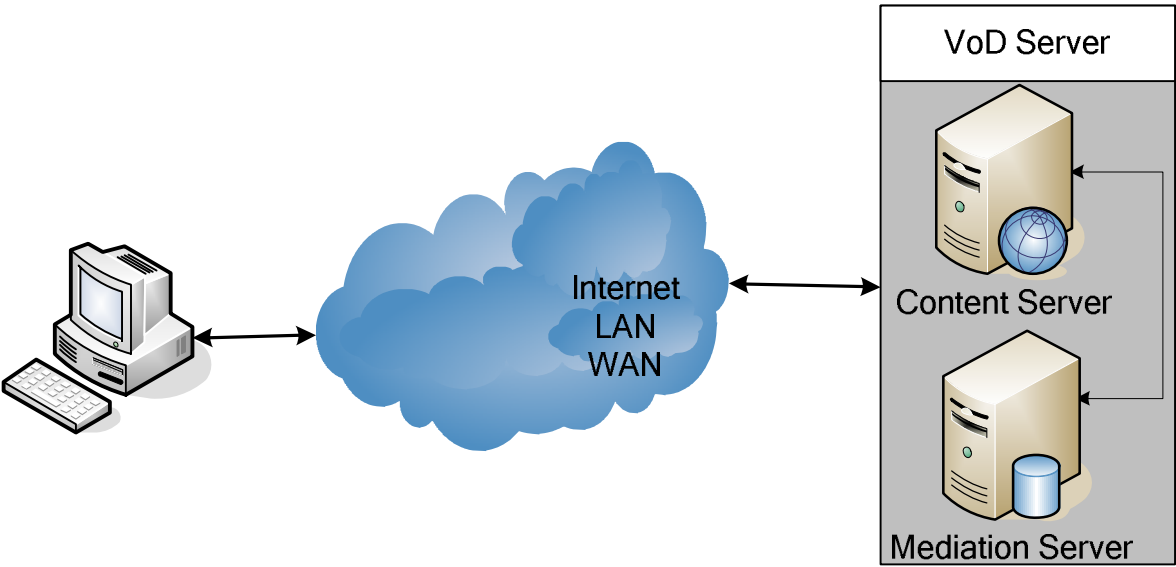


Figure 2-1 Architecture of Video on Demand system on computer networks

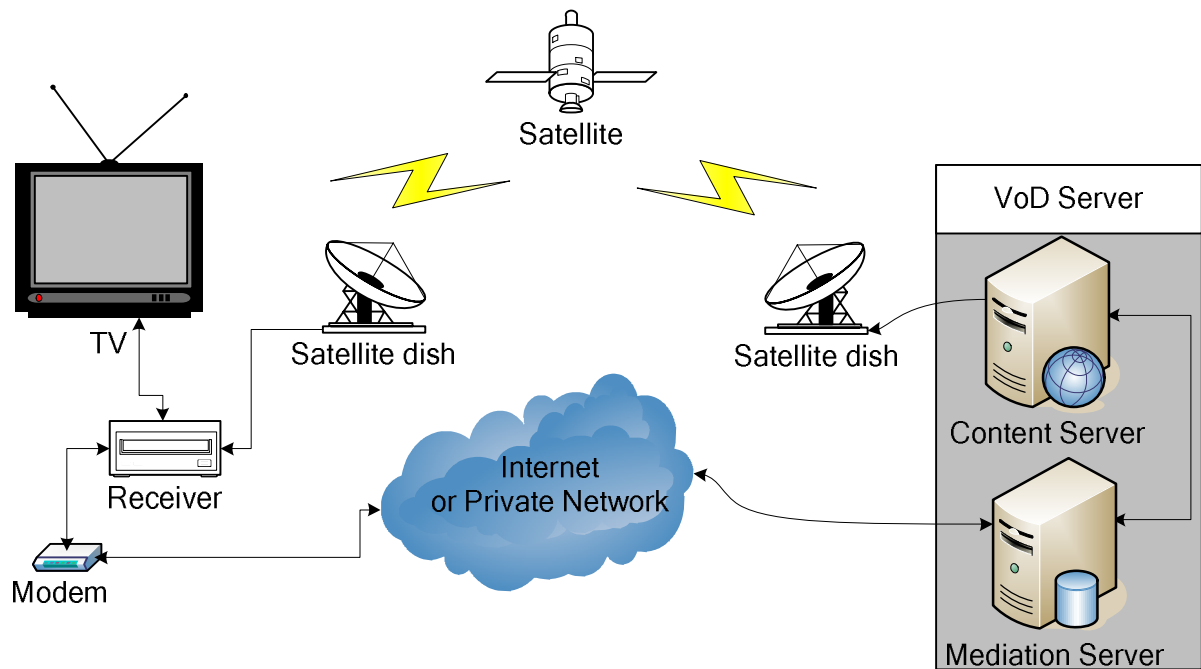


Figure 2-2 Architecture of Video on Demand system on TV networks

Figure 2-2 shows typical architecture of VoD system on TV networks. In this type of configuration, user first subscribes over the private network and then waits or immediately starts watching the program depending on the type of subscription. People are interested in watching much more personalized programs; this is the main reason for switching to VoD type of TV channels instead of classical TV channels. Today's TV network infrastructure mostly allows Subscription type VoD systems in which the subscribers buy the program before the session starts and wait for the session to start.

Traditional TV system infrastructure is not suitable for True VoD because of the one transmission channel per user requirements. Most of the development and research on TVoD and MVoD are made on computer networks but there are many reasons for TV broadcasting companies to switch to user oriented systems instead of group oriented systems, because user oriented advertisement features provide hundreds of times more profit than group oriented advertisements. The fundamental

reason for that is; a customer for TV VoD systems should subscribe for the related video in order to watch it. Therefore the company that is offering this service should have knowledge about the subscriber, the company will know the job, hobbies and the type of movies that the subscriber likes. Since the company will offer one transmission channel per user, there shouldn't be any restriction for user oriented advertisements.

VoD systems are separated into many categories by their functionalities, interactivities and the type of connection that they use. These services may include pausing, stopping, forwarding or rewinding of the media where we call them as VCR-like facilities. The names of these services listed according to their functionalities are described below:

Classification	Properties	Distribution	Orientation
True VoD	Supports VCR-like facilities. User has full control over the video. The session can be started immediately on user request.	Unicast	User
Near VoD	Supports partially VCR-like facilities but allocates multiple transmission channels for the same video by time shifts on each channel. E.g. 5 minutes of shifts per channel.	Multicast	Group
Quasi VoD	Does not support VCR-like facilities but can be accessed in any time. The users can switch to another group which has different functionalities.	Multicast	Group
Subscription VoD	User subscribes for a program and delivery time is scheduled and must be waited for the program to start.	Multicast	Group

No-VoD	Continuous un-interruptible media.	Broadcast or Multicast	Group
---------------	------------------------------------	---------------------------	-------

Table 2.1 Classification of VoD systems

As seen in Table 2.1, except the TVoD, all the other types are approximations to True VoD. Some implementations of Multicast TVoD are also made, but those systems use localized buffering when client makes pauses on the video. The details of the two fundamental VoD types are described in the following sections.

2.1.1 True Video on Demand

TVoD is the most user-oriented, end personalized service as compared to other services. TVoD supports VCR-like facilities and if the underlying system supports it, the user also will be able to select other features such as the media quality, subtitles, camera angles and video language.

On the other hand, TVoD is the most expensive system compared to all other types. The increasing number of simultaneous users affects system scalability because of the transmission channel requirements. There is no doubt that, TVoD system must be served in unicast manner in order to deliver the video right at the requested moment. The videos are watched by their popularity and mostly in the prime-time; this state is described by Zipf distribution, more about Zipf distribution on videos can be found in the SITARAM's [1994] paper on "Scheduling Policies for an On-Demand Video Server with Batching". On the other hand if the content server of TVoD system is connected to an ATM (Asynchronous Transfer Mode) switch, that will be equal to 155 Mbps (Mega bits /

second) and if we think of the average video rate for 255 kbps (kilo bits/second), a single server should be able to serve at least five hundred simultaneous users at the same time. For the multiples of this result, distributed network architecture for content servers could be applied.

Some real life examples of TVoD systems are Real[®] Player, Apple[®] QuickTime Player and Microsoft[®] Media Player. These systems mentioned here require the media files to be published in their compression formats which also require a server interpreter and client side plug-in to play these types of videos. These types of policies make the technology to get stuck at some point, because the respective companies do not share the system source.

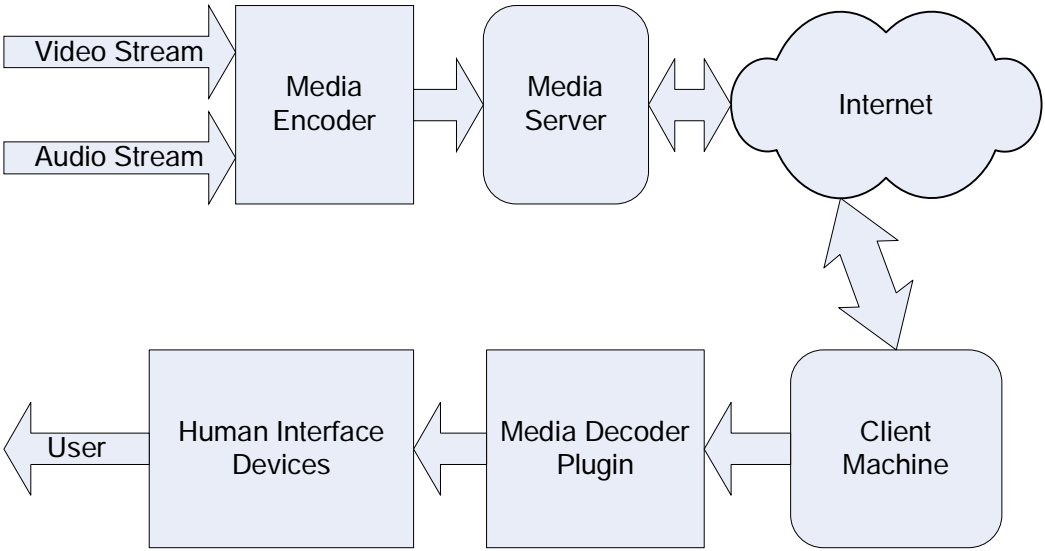


Figure 2-3 Working principles of commercial media players

Figure 2-3 represents typical architecture of real life TVoD examples, since all of them use closed-loop systems, they do not have enough media flexibility. On the other hand open-source projects are getting more sophisticated. An example of this kind of project is used in Nicholas Framster’s [2000] “Adaptive Delivery of Real-Time Streaming Video” thesis. In this thesis, Framster used OpenDivX [2001] technology to

support MPEG-4 video and used RTP (Real Time Transport Protocol) [1996] for streaming video transmission. The RTP protocol is especially suitable for multicast systems; this protocol is described in the network section of this chapter. The multicast type of TVoD system is implemented by Huadong Ma and Kang G. Shin [1999] in the paper “Multicast Video-on-Demand Services”. Both multicast and unicast connections have advantages and disadvantages. Comparison of these connection types are represented in Table 2.2.

Property	Unicast	Multicast
Number of channels	One to many	One
Network load	$0 < \sum_0^n uc_n < nc^{**}$	Few
Cost	High	Few
Real Time	Yes	No
Interactivity	Yes	No
Heterogeneity	Yes	No
Reliability	Yes	No

Table 2.2 Comparison of Unicast and Multicast transmission.

n: Number of users, uc: bandwidth assigned for the user depending on the video quality, nc: network’s available bandwidth.

Table 2.2 briefly compares the benefits of the two network connection types, there is also another type called “broadcasting” which is not of concern here and it is not a new technology. Multicast increases the quality of multi-peer communications and it is a built in feature in IPv6 described by Internet Engineering Task Force (IETF) [1998]. Multicast systems use one channel per video and the clients in the multicast group have to wait for the video to start, this limitation decreases the QoS (Quality of Service) but effectively decreases the number of channels allocated. Since the users are starting to watch the video at the same time, surely there wouldn’t be any option to pause, forward

or reverse the video. Hence, Multicast VoD technology is losing from its lack of interactivity. For the reliability case; if one of the users in the multicast group loses one or more data packets, the compressed media could result in jitters, or could be completely corrupted; in most cases, there shouldn't be any chance to re-transmit the lost packets. Heterogeneity is also a problem for multicast groups, in a multicast group, the client's connection may vary from a dial-up to a LAN connection, and this leads a multicast transmission to be made over separate channels for different media qualities to fit connection boundaries. On the other hand unicast video delivery seems to be the best way to implement TVoD systems in order to deliver the functionalities of a VCR but the cost of the implementation and the underlying infrastructure is a quite large problem.

Someone should combine multicast and unicast transmission techniques together in one application platform and popular videos could be served in multicast manner, while the others could be served in the unicast mode. The network load of a typical TVoD system should dramatically increase with the increase in number of online users and this will lead to high, un-scaleable traffic in the network. In order to alleviate this problem, it is better to implement distributed network architecture for the TVoD systems. This type of application is described in the paper of Loeser, Altenberd, Ditze, Mueller [2002] named "Distributed Video on Demand Services on Peer to Peer Basis" and in the paper of Chan, Tobagi [2001] named "Distributed Servers Architecture for Networked Video Services".

2.1.2 Near Video on Demand

Today's network infrastructure allows NVoD type applications because of its less resource requirements than TVoD systems. For a system to be NVoD, the system must be serving popular and scheduled videos. The schedule of the videos is announced earlier and users wait for the scheduled video time, after the starting time of the video, the video is re-played on different channels by time shifts like 5 or 10 minutes as seen in figure 2.4. This gives the users the ability to pause the video for an amount of time depending on the number of shifting channels.

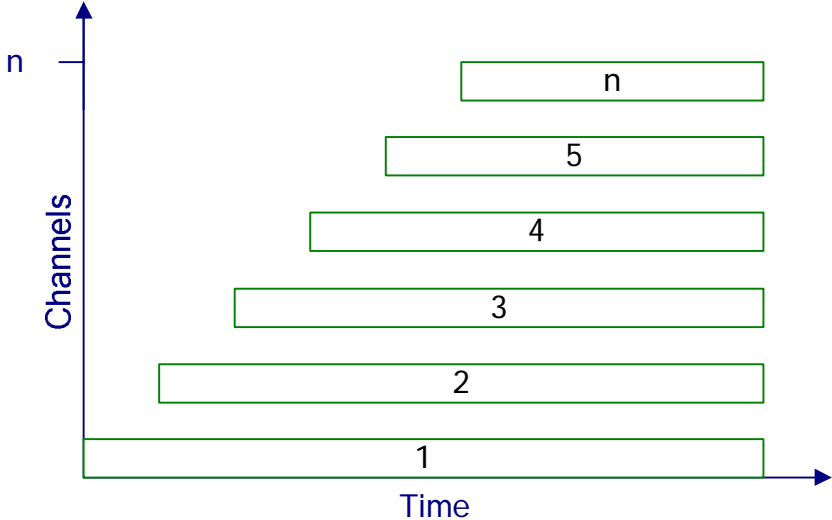


Figure 2-4 Channels in NVoD System

Nowadays, most of the NVoD systems are using multicast network transmission protocols in order to reach many clients, while the rest are using the broadcasting scheme. Since NVoD is a light system, it can be applied to digital satellite broadcasting; it doesn't require a large client side buffer and can be adapted to digital receivers. If the system is broadcasting, then it is also compatible with today's satellite systems.

2.2 Quality of Service in VoD Systems

There are many quantities that affect the Quality of Service (QoS) in VoD systems. A VoD system is dependent on the underlying network infrastructure, mass storage systems, disk scheduling and multimedia data compression.

2.2.1 Network Infrastructure

Networks for multimedia communications can be separated into two groups; unicast and multicast. Unicast is peer-to-peer and multicast is peer-to-many communications platform. In the multicast mode, data packets are passed from one server and the clients joining the multicast group catches the packets sent from the server. Both uses different protocols for communicating, unicast connection can be made in any protocol, such as TCP or UDP, but multicasting can be done in UDP protocol or in lower level protocols. UDP is a connectionless, non-reliable protocol; there is no guarantee that the sent datagram will reach the desired destination. In order to make multicasting reliable, new protocols that work over UDP are designed. One of them is Real-Time Transport Protocol (RTP) standardized by IETF [1996].

RTP is a transport protocol designed for the streaming of multimedia data. It is designed to cover the disadvantages of UDP protocol. A multicast UDP datagram may travel several paths and routers to reach for to destination and datagrams sent in sequence may reach the destination in different sequence orders. RTP provides timing and stream synchronization in order to solve this problem. RTP does not guarantee QoS or delivery guarantee; it just helps the system by its timing system and stream

synchronization mechanisms. The receiver system must track the datagram timings and must store un-synchronized packet deliveries to re-construct the whole message. RTP also uses a control protocol called RTCP (Real-time Transport Control Protocol) in order to get transmission reports from the receivers to track the quality of the stream. More on RTP could be found in the Internet RFC 1889 of IETF [1996].

Overall network performance on multimedia communications depends mostly on the error rate, throughput and delay, because most of the multimedia traffic consists of constant bit rate data such as video and audio. The throughput of the underlying network system must handle enough bit rates for desired number of clients. High error rates on un-reliable transmissions may corrupt the entire presentation, if the video or audio forming the presentation is compressed.

If the transmission is connection oriented like TCP, the server will not send any packages until it gets the previous packages' acknowledgement. Hence, because of this situation the round-trip delay must be small enough; this makes TCP not a good transmission protocol for multimedia traffic, because TCP will spend a large amount of time in waiting the acknowledgement data; hence, it will not be able to use the full performance of the underlying network technology. On the other hand, there hasn't been any rule to limit peer-to-peer connections to one TCP pipe; more pipes can be opened in order to use the available capacity of the network. With this method W. Allcock [2002] proved that 80% capacity of the network resources could be used. Figure 2-5 below shows the detailed diagram of one pipe vs. multi-pipe data transmission.

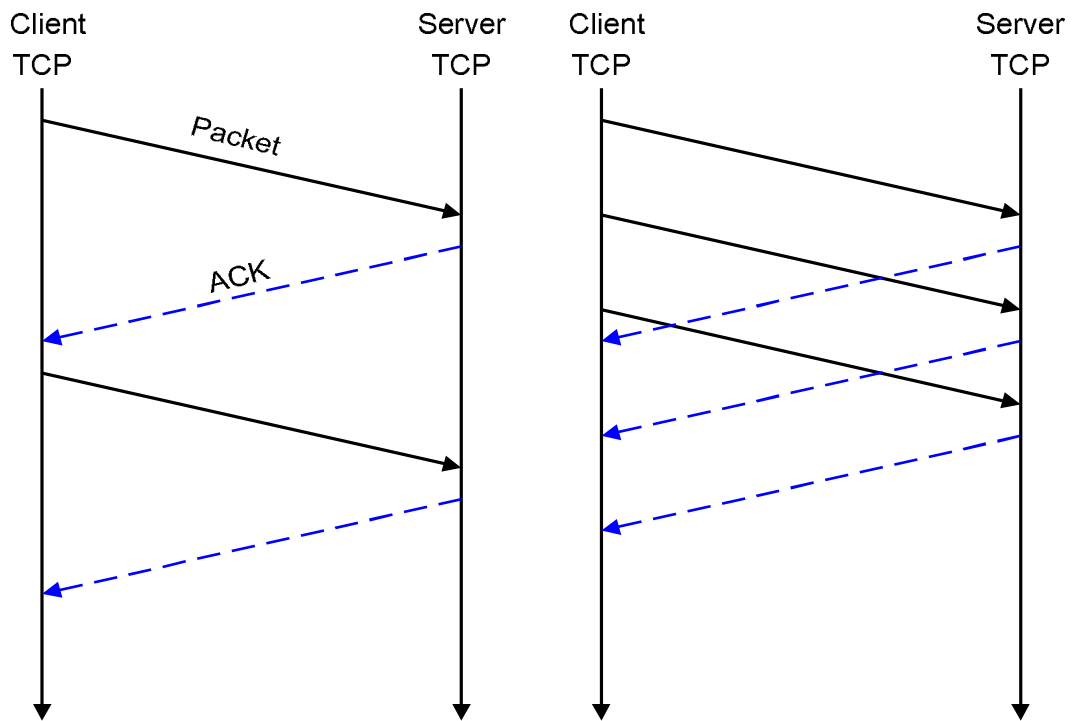


Figure 2-5 TCP one pipe vs. multi-pipe transmission

“Asynchronous Transfer Mode (ATM) is the world's most widely deployed backbone technology” noted by the ATM Forum [1995]. ATM uses “cells” of size 53 octets to transmit data through the network. These 53 octets of data are composed of five octets for header information and 48 octets for the data to be transmitted. ATM supports many connections at the same time using the virtual circuit concept and Quality of Service (QoS) is natively present in ATM networks. Nowadays an ATM infrastructure can handle up to 622 Mbps data rates. These features make ATM a preferable network system for multimedia transmissions.

2.2.2 Data Compression Methods

One of the main reasons for the improvement of multimedia services over the internet and in general networking is the developments in data compression. Data compression

has found itself a place in almost every field of networking technology. Data compression means reducing the size of the data by several ways [1987];

- By removing unused data,
- By applying compression methods or algorithms to select the minimal data that shows the main characteristics of the original data with or without loss.

In media files, unused data may consist of silences or if the media quality is too high, the frames per second could be decreased. In video data, human eye can see a movie in 24 frames per second without flickering, by reducing this amount the movie could still be seen, but with some flicker on it; however, the size of the video will be less. In audio files, if reducing is required, the stereo data could be converted to mono, this will also reduce the file size.

Run-Length encoding (RLE) is a basic coding method used in data files. This algorithm searches for the repeated characters in the data and the compression ratio depends on the repetitions available on the data. Another compression algorithm was published in 1977 by Lempel Ziv and modified in 1984 by Terry Welch and the algorithm got its name "LZW". LZW simply replaces the characters in the string and makes an index from those character groups. When decompressing, it uses the index to reconstruct the data. LZW algorithm is good especially in cases where repetition is available. These types of algorithms make lossless compressions and are not suitable for multimedia transmission on networks. By the nature of images and sounds, the resulting raw data may not contain too many repetitions and the streaming media need not be lossless, because human senses rarely feel the difference between lossy and lossless media. In multimedia communications, losing from media quality is not as

important as constant media delivery.

The relation between media quality and network bandwidth requirements is straight forward. If the media quality approaches to lossy, the bandwidth requirements would be less. In order to fulfill the quality and bandwidth requirements, new compression technologies are developed; these are discrete cosine transform and wavelet transform techniques which form the infrastructure of the MPEG (Moving Picture Experts Group) technology.

MPEG is the common name for some compression standards used in audio-visual information as described by the MPEG WWW home page. MPEGs are numbered by their functionalities and the areas that they are used. Nowadays the last addition to MPEGs is MPEG-21. MPEG-1 was introduced in 1993 and was standardized by ISO/IEC [1993]. MPEG-1 is a very popular standard and MPEG-1 compressed videos on standard CD-ROMs are available in commercial name as "Video-CD". MPEG-1 is used for digital audio and video for a bandwidth not more than 1.5 Mbps. MPEG-1 can also be used for digital audio compression which is named MPEG-1 audio layer-3 or MP3 in short has made itself a standard for audio compression. MPEG-2 is designed for interlaced video up to 100 Mbps stream rates and used as a standard for DVD (Digital Versatile Disc) video and HDTV (High Definition TV). MPEG-4 standard is based on the previous MPEG standards and instead of a new coding method ISO (International Organization for Standardization) decided to add artificial intelligence to existing MPEG standards. Instead of compressing the whole scene, MPEG-4 codes the steady background and the moving objects independently with different compression methods.

2.2.3 Mass Storage and Video Scheduling Algorithms

Mass storage is an important subject in VoD research. Video itself covers huge volumes of storage space, however, nowadays mass storage per gigabyte is less than 1\$ and with good media compression policy, the cost of the mass storage on the total system cost should be low enough. On the other hand, scheduling of media files in mass storage is still a problem for VoD servers. The system performance rapidly decreases on many connections when appropriate mass data scheduling algorithm is not applied and makes continuous delivery of the video streams impossible. Especially on modern network infrastructures like ATM, connection speeds may vary from 1 Mbps to 622 Mbps and the mass storage hardware not be able to satisfy the high speed requirements because of the fragmentation on stored data, hard disk bandwidth and buffer capacity. These three factors effect the total QoS of video servers, the effects of all factors can be decreased when storing and serving the videos. While storing the videos, an algorithm may prevent video data from fragmentation and another algorithm may schedule the video data positions in hard drive in order to prevent overheads in hard drive head movements. On the other hand while serving the videos; buffering policies may be applied to guarantee the continuous delivery of the video streams. The buffering and caching techniques are discussed in the papers of DAN and SITARAM and also in the paper of DAN, DIAS, MUKHERE, SITARAM and TEWARI [1995]. Storage schemes like “original phase based, phase based with replication, phase based with pre-replication and phase based with on-RAM replication” is described in the paper of SUMARI, MERABTI and PEREIRA [1999]. Researches on disk scheduling algorithms are discussed in the papers of LEE and SRIVASTAVA [1999], in the paper of AGGARWAL, WOLF and YU [1996] and in the paper of LEE and DU [1997].

2.3 Distance Education

Distance education, as described in Chapter 1, is separated into two parts by their functionalities and interactivities that we call; synchronous and asynchronous distance education. Synchronous Distance Education (SDE) term came to our lives after the revolution of the sub-networking technologies because SDE requires two sided communications media and this concept cannot be applied to telephone or TV systems. The previous researches carried out on the SDE are mostly based on finding the center point between interactivity and available system resources. In other words; a chance to find someone on the other side of the cable to ask a question about the education material requires at least two sided communications media, high bandwidth and a person to answer the questions. Nowadays, availability of multicasting, ATM technologies and high compression ratios make interactivity easier but at some point SDE also requires a scheduling of the lesson time for the presentations and leads the system to an un-automated condition, i.e. the subscriber or learner cannot start lesson at a certain time, and the learner must wait for the teacher to start the session. On the other hand, Asynchronous Distance Education (ADE) can be fully automated with the help of knowledge engineering and the system at some point should be able to answer the questions directed to the system knowledge base. This would lead the system to be fully automated in time on every new addition to the knowledge base of the related educational material. ADE applications could be supported with course related materials within the application or with the huge resources of the Internet. However, we are still a long way away from such automation at present time.

Before the revolution of the Internet, Distance Education (DE) was dependant on the TV programs and supported by ordinary classroom examinations to organize a

Distance Learning scheme. This kind of programs were made available in Turkey, UK and other countries under the name “Open University” or “Open Education for university courses”, and the students registering to this program gain the right to join the examinations without continuously attending classroom lectures. The disadvantages of “Open Education” are; the student must wait for the lecture time to be presented on the TV and the student does not have any chance to ask a question to the instructor. These systems are called: Asynchronous DE (ADE) systems. ADE took the second step with the improvements in technology ADE courses were made available on CD-ROMs. An example of this kind of implementation is that proposed by Aleksic-Maslac and Jeren [2001]. They made “Signals and Systems” course available on CD-ROMs as Power Point presentations and presented the results in the paper called “Asynchronous Distance Learning Model”, but the revolution of the Internet totally changed the boundaries and the concepts of ADE. One of the most significant changes to ADE are; the learners shouldn't have to wait for the lecture time, the educational material could be supported by numerous resources of the Internet and at the end of each session, short quizzes, examinations and some tests could also be applied online and if the ADE system is thought to be insecure there are many companies on the Internet making online examinations with proven trust-worthiness. Nowadays, DE is mostly dependant on multimedia and sub-networking technologies.

Multimedia technologies are developing faster than sub-networking technologies and the infrastructure of network programming does not change with the increase in the speed of sub-networks; because the network protocols are standardized by international bodies such as The Internet Engineering Task Force (IETF) and are used for long period of time. On the other hand, if a multimedia standard changes, most probably the DE application must change. We face this situation in home videos also;

with the help of image compression technologies like MPEG-1, a laser disc is minimized to a CD that is called VCD (Video CD) and with the introduction of MPEG-4 home videos could be placed in the same physical size of a VCD but the media quality is increased many times. In this case the hardware technology has not changed but the technologies for compression and de-compression of these media faced significant changes. The latest development on Internet based video applications is made by an open source contribution group under the name "Project Mayo". Their research is based on the transmission of MPEG-4 video over the Internet by using Real-time Transport Protocol (RTP) and more details could be found on their web site.

The previous research on DE is reported in references. The difference between previous research and this thesis is; the system designed for this thesis is made on the principles of ADE but the system is aiming to minimize the disadvantages of the ADE by supporting instant access course videos with "html" based course related materials synchronized to the video stream and the system has the advantage of allowing the choice of the networking protocols and the media types.

3 CURRENT TECHNOLOGIES AND APIS FOR VOD SYSTEMS

VoD incorporates a number of software and hardware technologies as of necessity. These include the server and client side hardware platforms and the software systems. Rather than designing and developing everything from the scratch, it is wiser in this case to utilize the available software technologies widely used in the computing environments considered. The interest is in the multimedia and programming tools and techniques as well as the data communications infrastructure in order to design and develop a Video-on-Demand system. Hence functions and features of various existing application programming interfaces (APIs) that may be used for VoD development are of interest in this thesis.

Application Programming Interface (API) is the name for a set of functions or class packages that are written by software engineers or the programmers to make an interface to computer hardware or software. APIs have many advantages; first of all, APIs encapsulate the low-level routines and serve easier functions or classes to handle low-level routines and prevent the programmer from code re-writing; hence the application programming becomes easier. Some APIs such as DirectX and OpenGL is made as alternatives to the operating system's (OS) graphical device interfaces (GDI) APIs. Standard GDIs are too slow for multimedia applications, because these APIs are designed for general case situations but multimedia applications require more than general cases. In Microsoft Windows operating systems, a call to graphical device interface must pass from the system messaging queue and this queue is shared by

other system calls such as keyboard and mouse calls; thus it cannot be processed instantly. Some APIs may also use other APIs within itself e.g.; WinInet API is using Winsock2 API to simplify network connections using FTP, Gopher etc.

The detailed explanation of the APIs and the technologies used in the VoD system development is covered in the following sections.

3.1 C/C++, Object Oriented Programming and Classes

C was developed by Dennis M. Ritchie in 1970 and in 80s C became the primary system development programming language. C can be described as a mid-level programming language by its attributes. The functionality, structured programming style and platform independence makes C more powerful than many other programming languages. The benefits of the C programming language can be described as:

- By its flow control mechanism, it could be used as a high level language but can handle low-level routines,
- Bit, byte and word manipulations,
- Inline assembly usage facility,
- Memory management, information pointing instead of information moving.
- Structured programming, i.e. GOTO or JUMP statements are not used but are available for backward compatibility.

UNIX and Microsoft Windows operating systems are the two outstanding operating systems programmed by C programming language; besides, C is also suitable for special purpose applications. On the other hand if the code gets too long, C loses its functionality, becomes very difficult to understand and hardens code

correction. Because of these reasons C must be converted to a more programmer friendly structure without losing its functionality and its benefits; a new programming language is developed in the 1980s by Bjarne Stroustrup in Bell Labs called C++ and standardized by ANSI/ISO C++ committee in 1994. This language is derived from pure C and is fully compatible with C but this new language made a revolution in the style of coding; it is based on object oriented programming (OOP) paradigm. Instead of separate type definitions, OOP combines these types into classes. The benefits of C++ and OOP are;

- User defined types,
- Encapsulation,
- Inheritance,
- Polymorphism,
- Virtual functions,
- Class hierarchy,
- Code reuse.

Through the availability of code reuse, Microsoft developed classes for Windows API that is collectively called the Microsoft Foundation Classes which are described in the next section.

3.2 Microsoft Foundation Classes

Microsoft Foundation Classes (MFC) is a collection of C++ classes designed for Microsoft Windows to handle and encapsulate complex Windows situations [1995]. MFC includes more than 200 classes to handle visual GUI components, file I/O, memory management, networking classes and more. MFC libraries are updated by the system on every fundamental change on the OS to work properly in all conditions. Prior

to MFC, to show an html page in an application would require the complete code of the html browser or use of Active Template Libraries (ATL). Hence, application development becomes more complex and coarse, but in MFC; only using the html browser class is enough. This improves the system performance by using available system libraries and requires less space because these libraries are already used by the system and they are installed with the system. The VoD system designed in this thesis is also developed using the MFC, because the VoD Player has a built-in html browser. During the development, standard C and ATL libraries also tried but the application size reached more than 30 Mbytes, resulting in increased compilation time and decrease in the system performance. MFC provides an object oriented interface to make application development easier. Using a built in html browser for the applications necessitates the use of Component Object Model (COM), Object Linking and Embedding (OLE) or ActiveX; MFC simplifies the development of these technologies by its internal capabilities and there is no reason to re-write a html browser.

3.3 Component Object Model (COM)

Component Object Model is the latest technology that comes after Dynamic Link Libraries (DLLs). DLLs helped software developers to separate their end user application into many parts and instead of compiling and distributing the updated versions, hence only re-compiling and re-distributing the updated application DLLs is necessary. Microsoft Windows DLLs could also be used in special purpose applications, some of the Windows DLLs used in VoD system are windows2.dll which covers basic network communications such as TCP and UDP, GDI32.dll used for GUI.

Component Object Model uses the same principles of integrated circuit (IC), a typical IC has input, outputs, a power input and a ground connection. COM technology is built on the same idea; a COM programming only requires the developer to know the inputs and the outputs of the COM object. Besides, a COM object could be connected to another COM object if necessary. As an example, playing a video resource would require the video file, a video-sound splitter, video decoder, sound decoder and video renderer. Nowadays video files have different compression techniques and algorithms that require different decoders. Instead of making different applications for different video types, COM enabled application could select these decoders and could connect their pins on demand without any change in the software. An implementation and a comparison of digital IC systems and COM are given in Figure 3-1 and Figure 3-2.

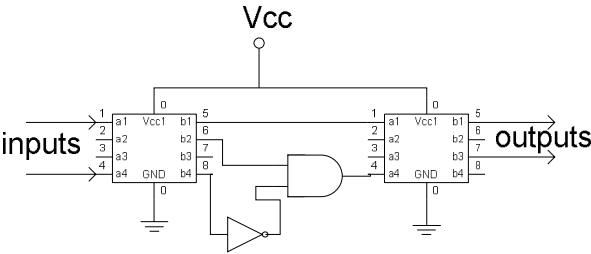


Figure 3-1 A typical logic circuit.

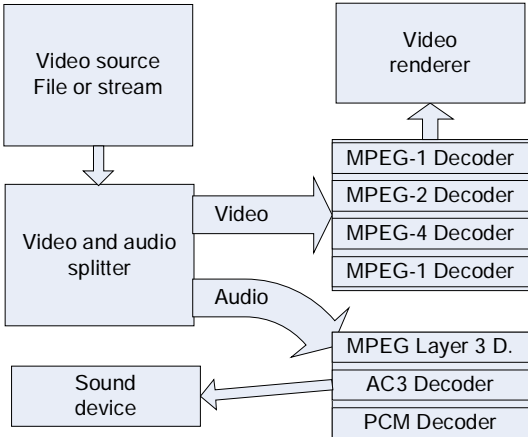


Figure 3-2 Sample Component Object Model used in DirectX

The system in Figure 3-1 has a number of inputs and outputs. The outputs of the system are formed by the effects of the inputs to the system. As long as the inputs and the outputs are in the same format, new versions or improved versions of the ICs could be replaced by the old ones, thus the overall performance of the system could increase depending on the type of the system. To change ICs on a circuit board, the IC mounts must be designed in a way to allow un-mounting of these ICs. In modern boards IC sockets are heavily used. The same idea also applies to COM objects, COM objects could be changed by newer versions without recompiling the whole application. Standard applications are formed of binary data. The compiler once generates the binary code; there should be no possibility to change the code without re-compiling. In every hardware or operating system improvement, the application must be re-compiled to get the benefits, but COM objects could be dynamically linked to the application with DLLs and enormous amounts of updates could be prevented as soon as the COM input and the output pins are the same. The other benefit of the COM is that it is programming language independent and once registered to the system; every application can use the components. An example of COM is the MPEG-4 codec by the DivX Group. Most of the video player applications use the DivX codec, thus the application size decreases dramatically since the codec is usable by the video players which support DirectX.

Microsoft DirectX also uses COM within its code in order to make updates easily; because nowadays both hardware and software is changing rapidly. Since DirectX makes interfaces directly to hardware, updating is compulsory. COM also supplies an applications functionality to be used by other applications.

3.4 Microsoft DirectX

Microsoft Windows Graphical Device Interface (GDI) is too slow for generating multimedia applications or games and Microsoft introduced low-level APIs for better hardware interfacing. In older versions of the PC OS, such as DOS, to play any multimedia file, the application must be compatible with the hardware. For an application to support all the hardware, the programmer had to write a set of codes that was specific to each hardware vendor. Nowadays, more than 100 different vendors are producing different types of multimedia cards and it is impossible to write specific code for all of them. DirectX and Windows Plug and Play system gets into the picture at this point. In DirectX the interface for the multimedia cards are the same, the programmer just writes the code what (s)he intended to write, no more interfacing and messy applications are needed to be developed at all. DirectX uses all the advantages of the hardware. If the requested feature does not exist in the hardware, DirectX emulates this feature with its Hardware Emulation Layer (HEL) else if the hardware supports the requested feature, DirectX directly makes interfaces by its Hardware Abstraction Layer (HAL) to the hardware with the vendor supported DirectX drivers as seen in Figure 3-3.

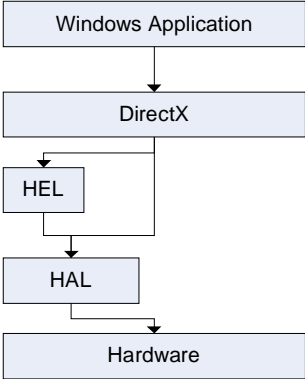


Figure 3-3 DirectX hardware interfacing

DirectX is used in the VoD Player and the VoD Studio for displaying videos and playing sounds; thus low-level codes required for video and sound interfacing are left to DirectX. To make a classical interface to a video file or stream, first of all, the AVI or MPEG header must be examined, depending on the header, the appropriate codec, video size, frame rate and the bit rate should be selected. Those steps are simply disregarded and are left to DirectX. As long as the required codec type is present in the client OS, DirectX can “open” any video type. This is indeed a critical benefit for the designed VoD system because in this manner projects can be built from any video resource, in any bandwidth and in any quality videos such as DivX, Xvid, mov and MPEGs.

3.5 ActiveX Controls

It is better to explain Object Linking and Embedding (OLE) before ActiveX. The OLE approach allows the placement of objects created by one application within another application, without the necessity for the applications to be the same. It is usual to copy a text from an html browser to a text editor, but if the text is copied to a word processor such as Microsoft Word, Word will not lose the properties of the text that is copied, such as; type face, thickness, italic or underlined. Also a photograph edited by a photo editor could be copied and pasted to another application. In the first example, the html browser acts as an OLE server and Word acts as an OLE container. OLE uses COM to serve this facility and ActiveX uses OLE inside.

ActiveX was developed by Microsoft after the Internet revolution. Microsoft decided to make the Internet more interactive and implemented OLE for the Internet

and named it as ActiveX. Viewing an Excel spread sheet, Adobe Acrobat document or a Macromedia Flash object is then enabled. ActiveX uses COM to implement OLE objects in html pages and an html page can also be shown within an application with all the benefits and properties of Microsoft Internet Explorer.

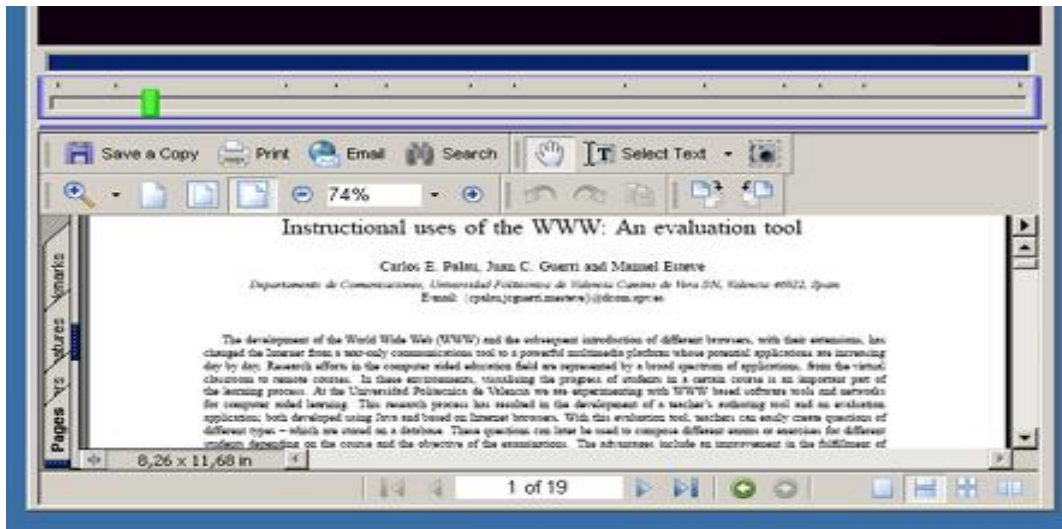


Figure 3-4 VoD Player ActiveX pane showing an Acrobat document

Figure 3-4 is a representation of ActiveX capabilities of the VoD applications. The Internet Explorer Server pane linked to VoD application by ActiveX has the capability to show or animate external documents as well as online html pages and Java applets. The need to show such kind of documents is to give the end users, i.e. students' ability to examine related material content in the boundaries of the educational tool, i.e. in the VoD Player.

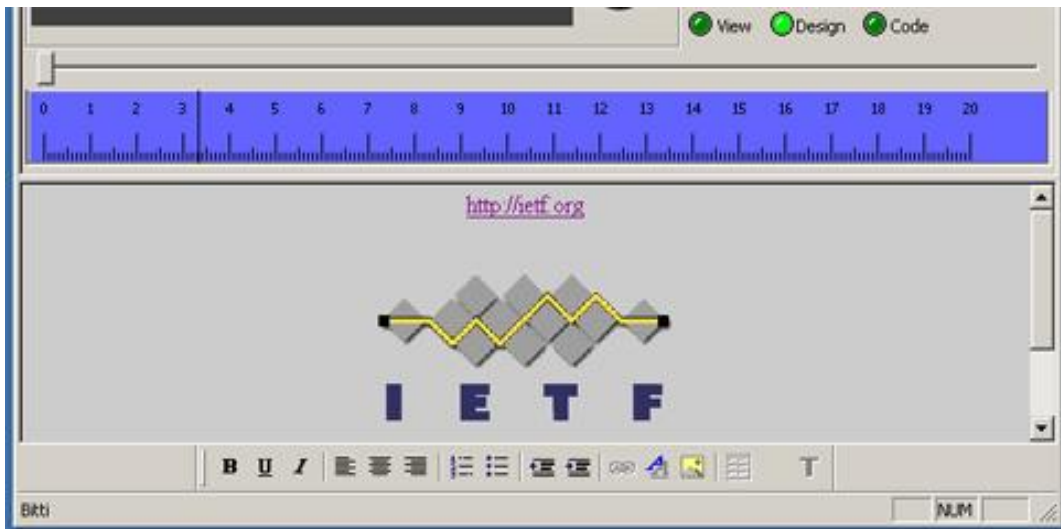


Figure 3-5 VoD Project Editor ActiveX pane DHTML editing facility

Figure 3-5 is a screen shot of VoD Project Editor which is used to design course projects by adding video and project related html pages to the project. The VoD Project Editor will be discussed in more detail in Chapter 5. This application includes three ActiveX controls in order to make the publishing process easier; an html viewer, a DHTML editing tool which is shown through the screen shot in Figure 3-5 and an html code viewer. The instructor designing VoD projects is able to design, copy, paste and link html pages inside the VoD Project Editor software utility.

3.6 MySQL C API

MySQL is a Structured Query Language (SQL) server developed as an open source product by MySQL AB. The reasons for selecting MySQL from the group of other SQL servers are; first of all MySQL is distributed free of charge under a GNU license, it is multi-threaded, multi-user and MySQL has better compatibility with Internet based scripting technologies such as PHP or Perl. Thus the Internet side scripting of VoD system is handled with the same database in order to track the user activities and other

administrational works.

MySQL C API is designed to make direct connections through an application without using external components such as Open Data Base Connectivity (ODBC). To get the benefits of this API, static linking of the DLL "libmysql.dll" to the application is required.

3.7 Windows Sockets (Winsock) API

A socket is a handle or number for transportation media interface. Windows sockets are derived from the original Berkeley Software Distribution (BSD) which was developed for UNIX systems [2002]. BSD sockets could also be used in Windows systems but the Winsock suit is designed to be compatible with the Windows messaging architecture. In UNIX systems, the network sockets are identified by "integers" while the Windows system uses different "handles" for the sockets. The difference comes from a reason; all I/O implementations in UNIX systems are file descriptors, hence network communications are made as reading or writing to a file which is simply identified by an integer. On the other hand Windows system only points file descriptors to resident files and network sockets cannot be handled as file descriptors. Another difference between UNIX and Windows networking is; UNIX systems use signaling and hardware interrupts heavily, while the Windows systems starting from Windows 95, does not use hardware interrupts.

Winsock makes an interface to the existing network protocols as an API. The API includes connection oriented (stream) and connectionless (datagram) protocols.

Those protocols include; Internet Protocol (IP) [RFC-791], IPX/SPX, NetBIOS, AppleTalk, ATM, Infrared and Raw sockets. In this thesis only the IP protocol is used and will be described briefly. IP is an abbreviation for Internet Protocol and is found in the third layer of the OSI model which is the network layer and is responsible from routing and addressing of the data packets (tracking and forwarding) in the network. The transport layer protocols; TCP (connection oriented) and UDP (connectionless) lay over this protocol.

The major property of the Winsock API is to determine the network protocols present in the OS and defining the characteristics of these protocols. After the initialization of Winsock sockets, the rest of the process is nearly the same as that of the BSD UNIX sockets programming.

3.8 Win32 Internet Services (WinInet) API

WinInet is a high-level API for FTP, Gopher and HTTP protocols. WinInet hides the low-level connection routines for the related protocols as described in the MSDN website [2003]. This API is used to make FTP file searches and creating, setting, removing directories as shown in Figure 3-6 below.

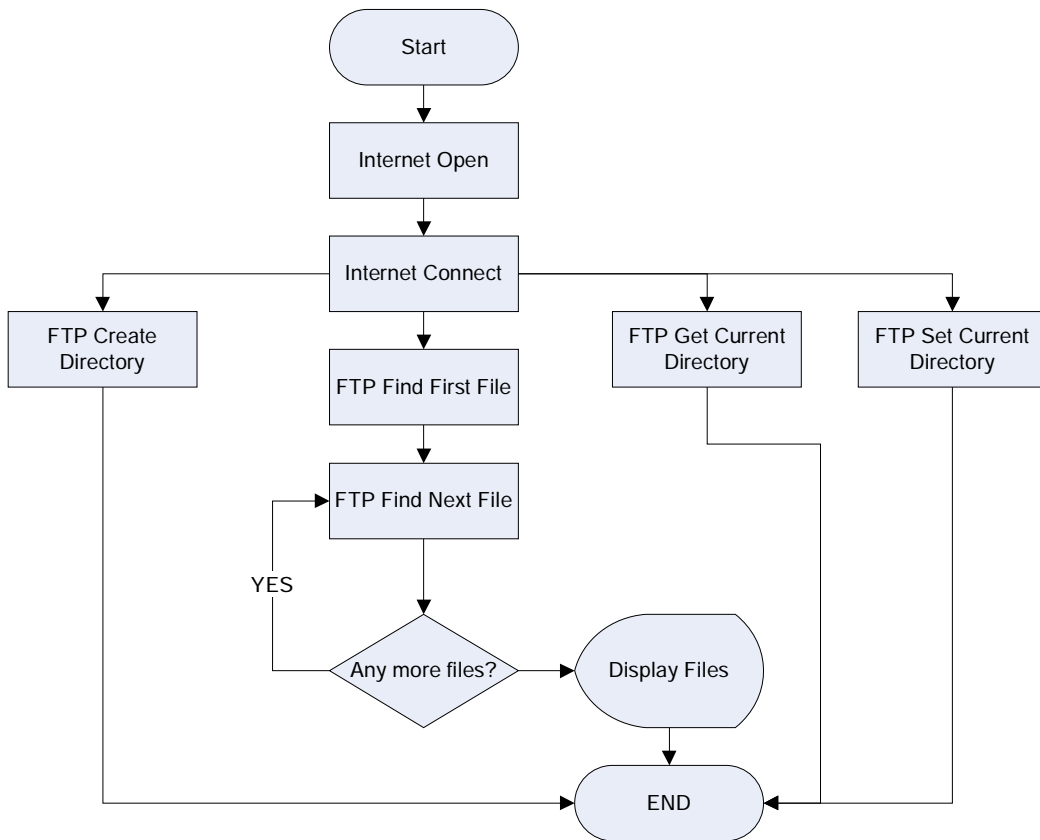


Figure 3-6 Winlnet flowchart for FTP operations

In the thesis, for more complex FTP functionalities the Winsock API is used instead of the Winlnet, because Winlnet does not allow its process to be controlled, i.e. if a Winlnet process starts, it is not possible to watch the process continuity, and therefore during the download process, the percentage of downloaded material can not be discovered. It is required to know the download percentage for VoD player, since the system starts playing the video after a certain percentage is downloaded. The advantage of Winlnet is to simplify common FTP commands and prevents the developer from re-writing raw TCP codes for the FTP process described in the RFC 354 [1972].

3.9 Globally Unique Identifiers

Globally Unique Identifier (GUID) is a pseudo-random number generation technology developed by Microsoft and is supposed to be unique and cannot be generated again. The standard for GUID is 16 bytes long and the words separated by “-“sign written in hexadecimal form as show below:

D871C341-061C-4c4a-9F44-CB90BB1D0D11

When creating a directory on a multi-user server, it is possible to try to create the same named directory. To prevent this situation, GUID is used on the server side of the VoD Project.

3.10 Cyclic Redundancy Check (CRC)

The Cyclic Redundancy Check is well defined by Sarwate[1998] as "Cyclic Redundancy Check (CRC) codes provide a simple yet powerful method of error detection during digital data transmission. Use of a table look-up in computing the CRC bits will efficiently implement these codes in software." in the paper of Sarwate [1998].

CRC is the name of a number which is derived from a calculation made over a chunk of data. CRC is based on binary division. A polynomial (or CRC generator or divisor) is used to divide the data bits. The CRC sequence of bits added to the data field so that the resulting data unit becomes exactly divisible by the polynomial. This polynomial always gives the same result over the same data, if the size and contents of the transmitted data hasn't changed. After the arrival of the packet at the destination,

header and the data are extracted from the packet and the CRC algorithm is re-applied to the same data. If the CRC information sent with the header is the same with the new CRC information, the data received is error-free and the chunk of data is stored, else, there are three possibilities; CRC data could be corrupted, data itself or the whole packet could be corrupted, in all cases the whole packet is requested again by the client from the server machine.

The advantage of CRC is that it uses a lookup table and a change in a single bit could change the resulting CRC data. CRC is used in the Reliable-UDP part of the VoD Project to test the datagram in the UDP packet. If corruption occurs in the data, the data is re-requested by the client. More on Reliable UDP protocol will be covered in Chapter 6.

4 DESIGN AND ANALYSIS OF THE VOD SYSTEM

Video on Demand systems necessitate the availability of digitized and compressed video sources which are managed and accessed through a VoD titles database. The video library created need to be indexed, sorted and linked based on appropriate subject indexing and according to level-instructional material related to the video material also needs the linking with the video libraries. This chapter presents the design and analysis of a VoD System for Computer Assisted and Distance Learning environments.

The VoD System infrastructure is built on the networking and software technologies and the design of the VoD System is based on the client-server architecture as shown in Figure 4-1:

The VoD application architecture consists of a number of clients and number of associated servers. Three types of clients and two types of servers are proposed. The server architecture is based on the proposed by Rousseau and Duda [1999]. Two server types are proposed: The Mediation Server and the Content Server.

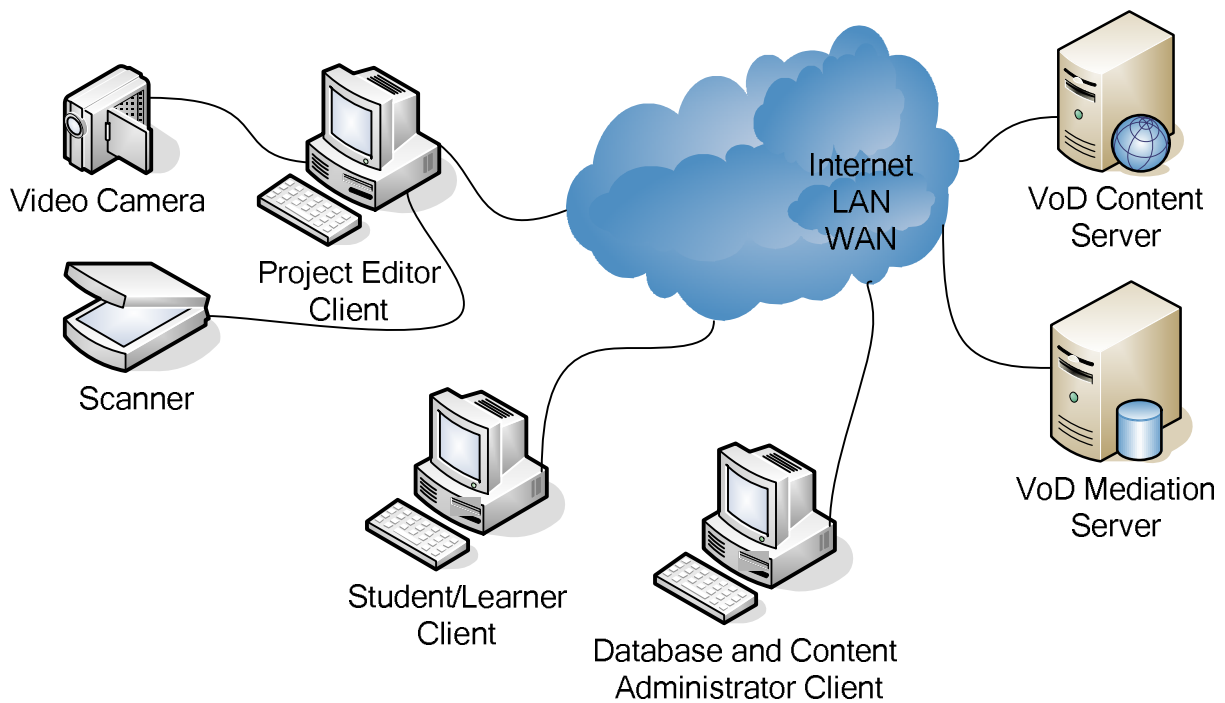


Figure 4-1VoD Client-Server Architecture combining the VoD System

The Mediation Server mediates between the users (administrators, students etc.) and the VoD system. Hence it keeps the necessary information about the users, courses, projects and the other system related data; it controls the access, carries out the authentication and stores the links to the VoD libraries.

The Content Server is only responsible from storing the multimedia contents and the content related supplementary materials such as images, html pages, sounds and etc. The Content Server does not make authentication on user basis; instead, the VoD system applications such as the VoD Player and the VoD Project Editor authenticate them selves to the Content Server(s). Further, the users who are using these tools must be authenticated via the Mediation Server when the above mentioned applications start running.

There are three classes of use and hence three types of users are identified in the VoD System, as depicted in Figure 4-1.

1. System Administrator: A person, who manages the servers, maintains the system, removes unused projects, sets the internal communication passwords for the VoD tools to authenticate themselves to the Content Servers, and manages the users, authors and the other administrators.
2. Project Editors/Authors/Instructors: These are either academic staff or the administrators who edit or prepare the VoD Presentations and have the rights to use VoD Project Editor, send the projects to the Content Server and register those projects in the Mediation Server.
3. Students/Learners: These are the end users who have access to use the VoD Player application in order to display the VoD presentations, access to web based facilities such as discussion threads and forums.

The overall VoD system has the main functionalities as described below:

- Networking: The end user is free to select the transport protocol to download the video stream from the Content Server from the project offered selection. Further, third party FTP servers may be used instead of the VoD content server applications such as the VoD UDP server and the VoD TCP server.
- Multimedia: There is no restriction on multimedia files and compression types in the VoD system. The VoD projects can be designed with uncompressed media such as wave file (.wav) or the latest compression types such as MPEG-4. And the media could be video with sound, just video or just sound. Also still images such as JPEG or Bitmap may be used in the presentations main screen.

- Related material: VoD system can show html pages inside the application itself. Since the related material manipulation is html based, there is no restriction on the content of the material, i.e. animations, graphics, images, java[®] scripts, ActiveX controls and online discussion boards or chat scripts may be used in the content material. Additional objects such as images and animations used in the html pages are automatically uploaded to the Content Server by the VoD Project Editor and the Project Editor also includes an html editor inside itself.

4.1 VoD System Configurations

VoD System has three main configurations that the end users, authors and administrators can use. Every configuration has different applications or interfaces to perform different tasks. These applications are shown in the Table 4.1:

	Client-Server Applications	Software Component
Client Side	Student/Learner Client	VoD Player
	Project Editor/Instructor	VoD Project Editor
	Database and Content Admin	Web access to server resources
Server Side	VoD Mediation Server	User and password databases, Video Project databases, Content server IP databases.
	VoD Content Server	TCP, UDP and FTP Servers, Video Library, Supporting materials.

Table 4.1 VoD Client-Server applications

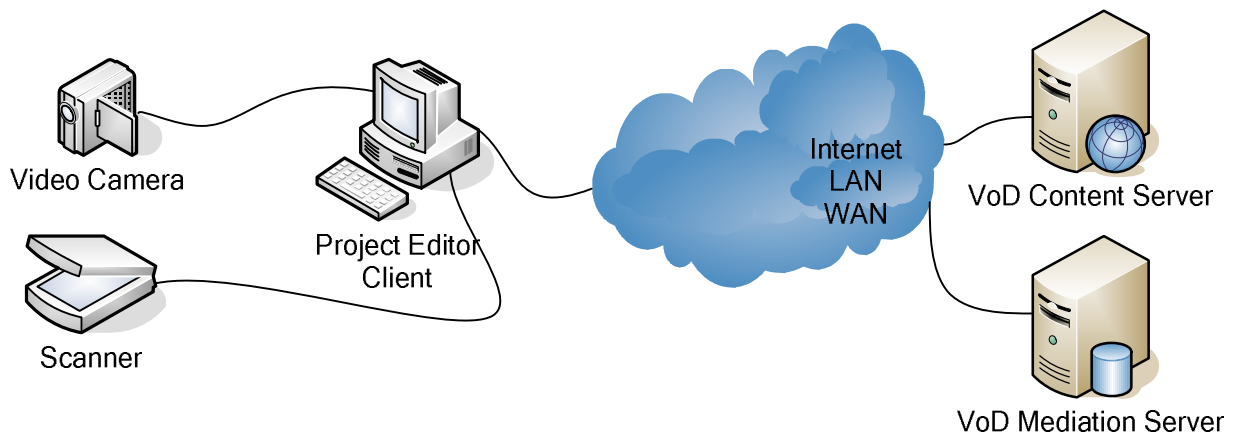


Figure 4-2 VoD Project Editor Configuration

The Figure 4-2 represents the configuration for using the VoD Project Editor; the Project Editor can be used by the registered project authors, lecturers or by the administrators, by adding the videos taken from a video camera and the content materials scanned from the lecture notes.

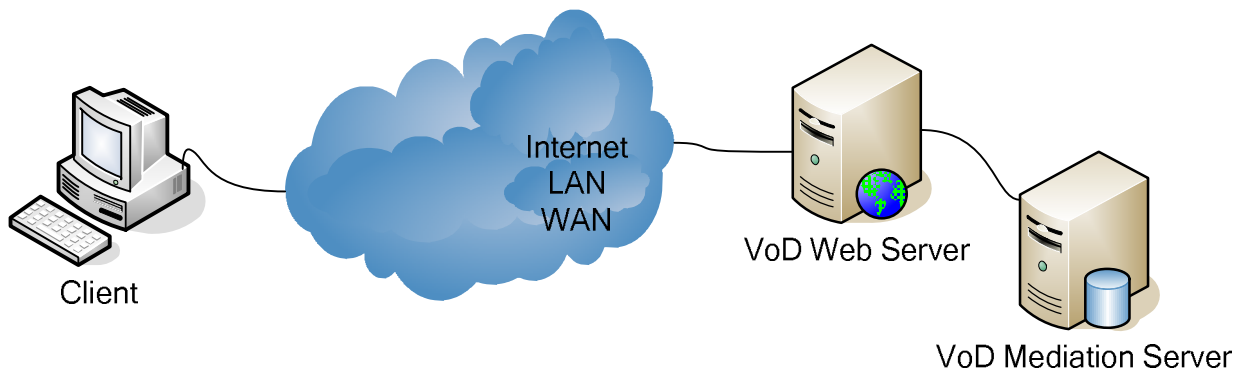


Figure 4-3 VoD Web services configuration

The VoD Web services can be used either by the administrators to manipulate the projects and take statistics or by the students to view the discussions made on a topic. Both the students and the administrators must be authenticated before logging into the VoD web services.

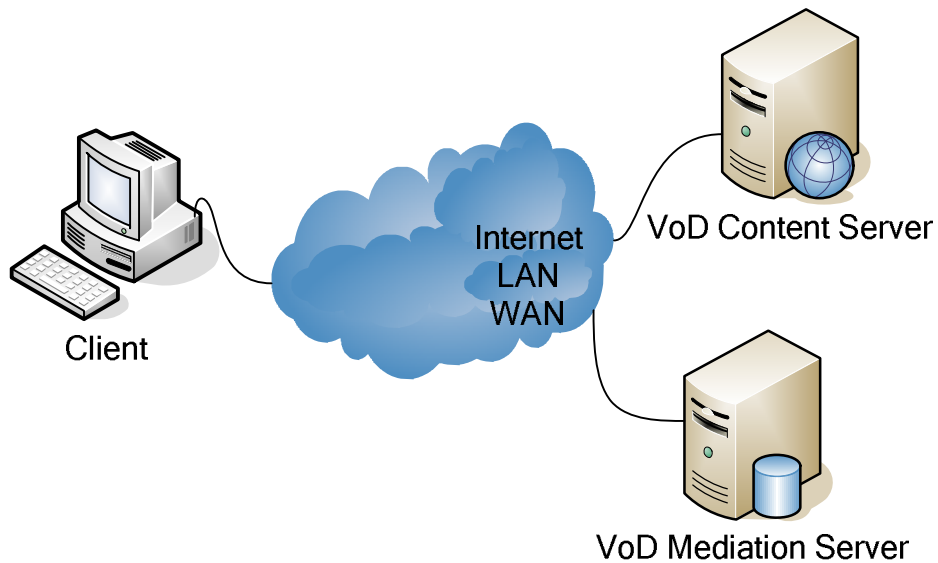


Figure 4-4 VoD DE configuration

The last feature of the VoD system is the VoD Player or in other words VoD Distance Education (DE) Tool. This tool could be used by the students to view, watch and surf in the VoD Projects designed by the VoD Project Editor.

4.2 The Organization of the VoD Client-Server System

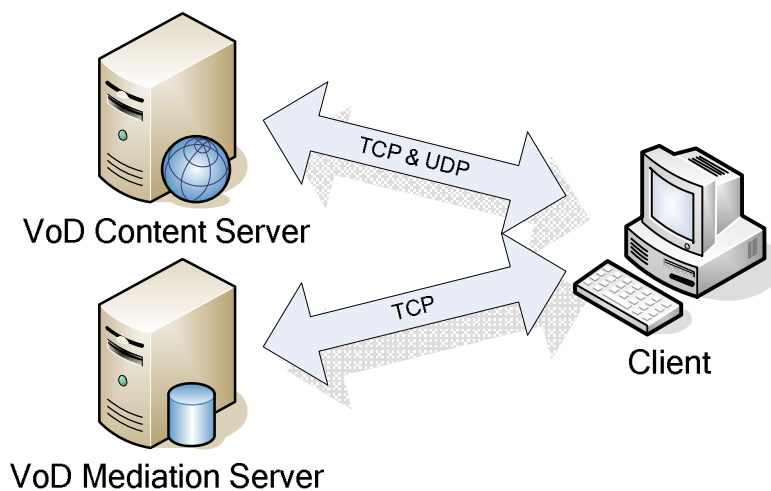


Figure 4-5 Client - Server communication

Figure 4-5 shows the network communications between the servers and the client of the VoD system. The servers do not communicate with each other, because the system is designed to allow more than one content server to be active in the system in order to distribute projects to the related Faculty Computer Centers. The client software makes the required negotiation with the Content Server(s) after getting the related information from the Mediation Server as shown in Figure 4-6. The data bases for the whole project are stored in the Mediation Server. In this way, the user and content management can be handled easily, i.e. the same users and the projects cannot be added to the system data base and a user cannot access the system at the same time from different locations.

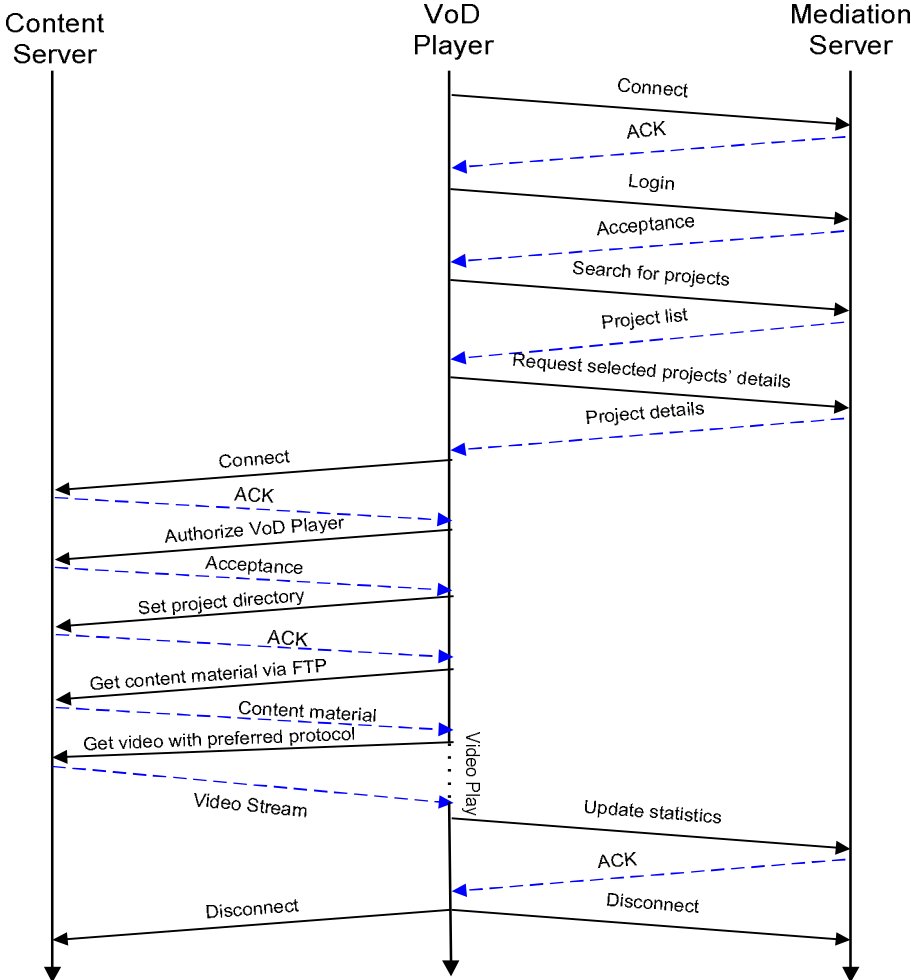


Figure 4-6 VoD Player - VoD System network negotiation

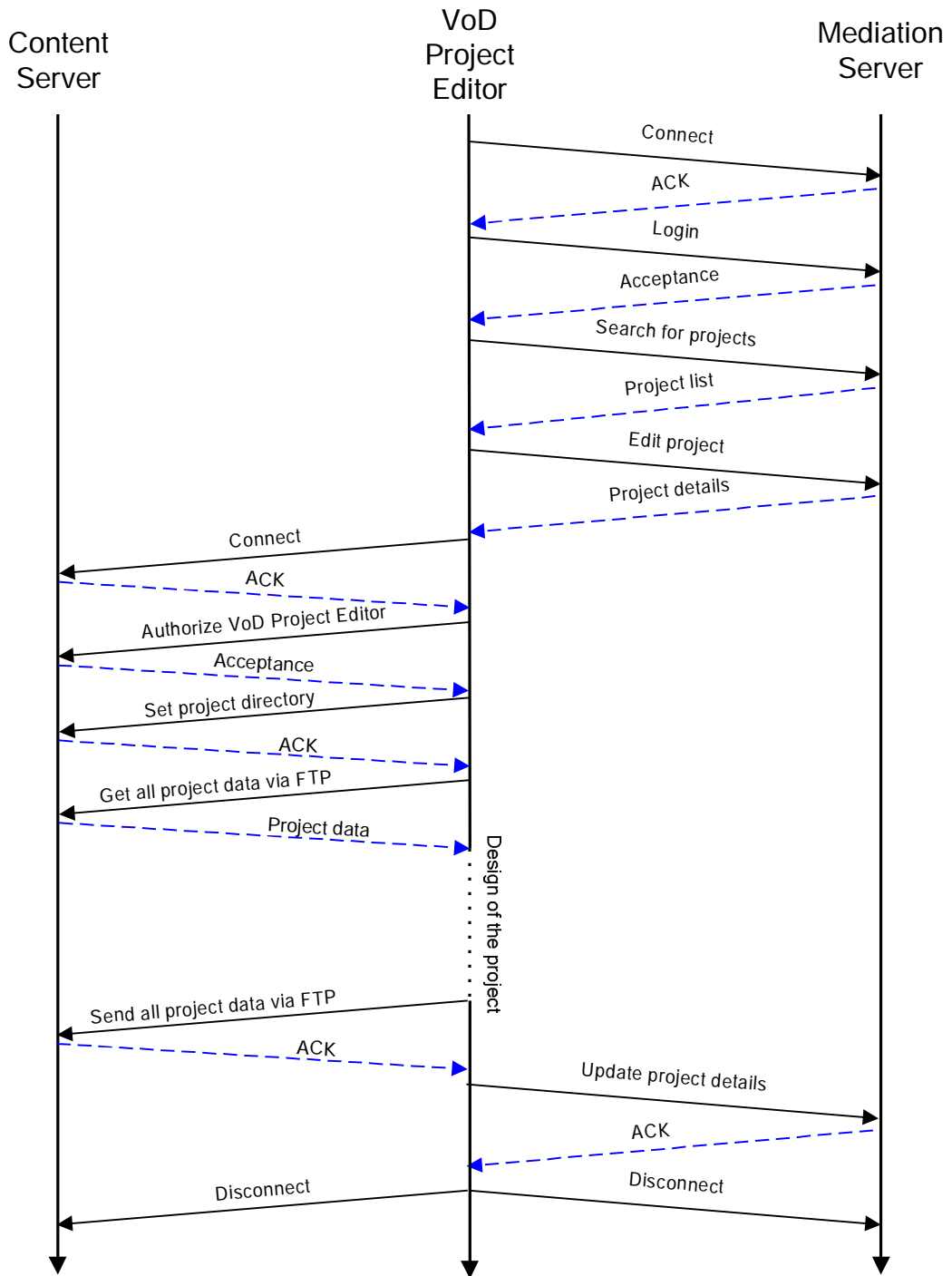


Figure 4-7 VoD Editor – VoD System negotiation (edit mode)

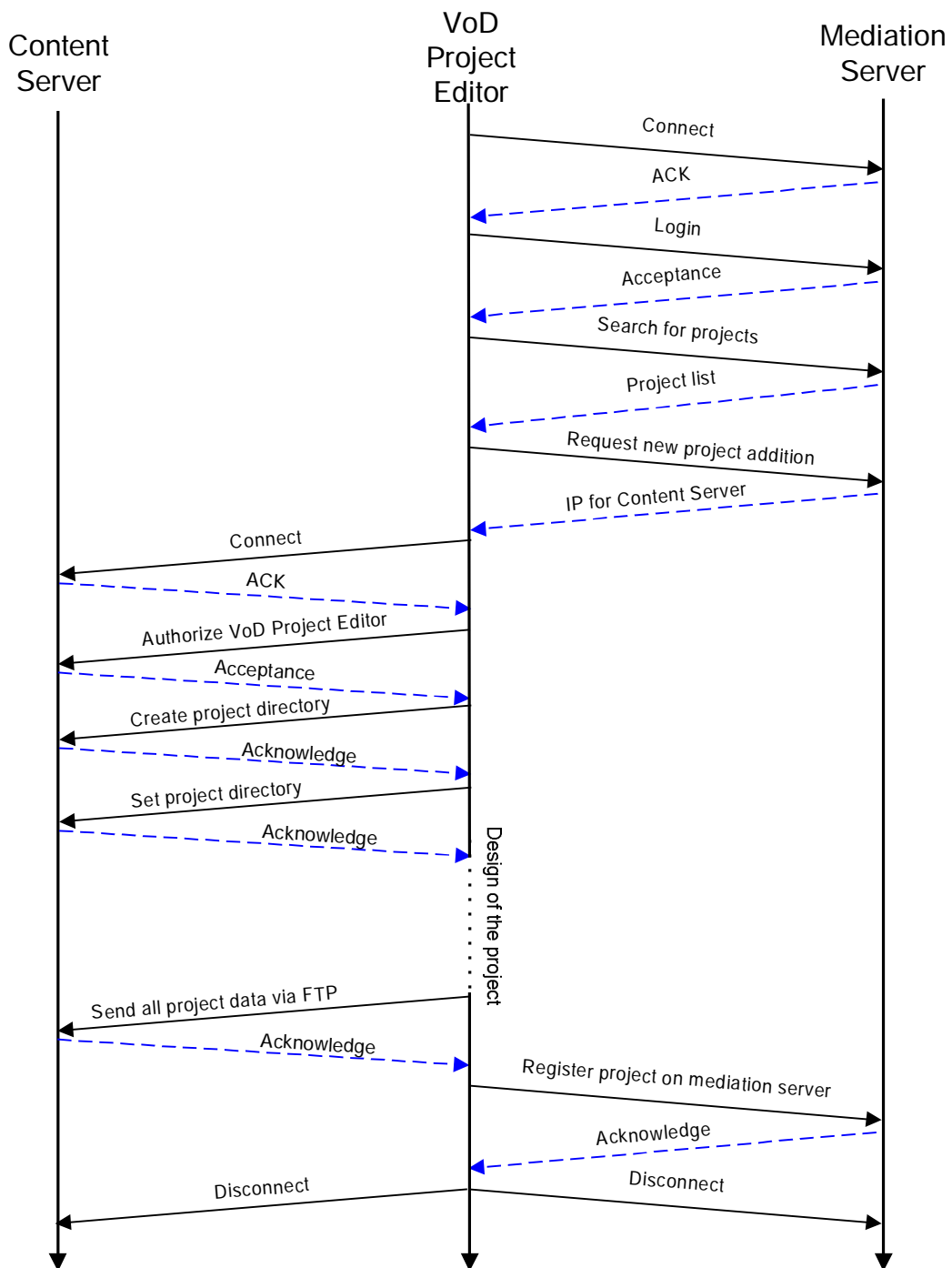


Figure 4-8 VoD Editor – VoD System negotiation

The communications of the VoD client applications to the Mediation Server is handled by the MySQL API, and the Content Server communications are handled by the VoD network Application Protocol (VAP). This layer selects the suitable communication protocol to process the required action in VoD Player as shown in Figure 4-6 but the VoD Project Editor uses only FTP to connect to the Content Server

as shown in Figure 4-7 and Figure 4-8. There are at least two connections are made to the Content Server from VoD Player, one connection is used to download content related materials with FTP and one connection is used to download the video with the preferred connection protocol including FTP, TCP, UDP and RUDP.

A user must authenticate him self through VoD gateway in order to use VoD client side applications as can be seen from the figures Figure 4-6, Figure 4-7 and Figure 4-8. It is impossible to use VoD client applications before the authentication. After the authentication process, the main windows of the applications will appear, and the user will be able to see the interfaces.

Communication between the Content Server and the client could be handled in four transport protocols; TCP/IP and UDP/IP, RUDP working on UDP/IP and FTP working on TCP/IP. TCP, UDP and RUDP server applications designed for this thesis and a third party FTP server must all be installed on the Content Server. VoD server applications use simple un-encrypted password authorization. The third party FTP server users must be authenticated, and authentication systems are also available in all commercial FTP servers.

To gather the existing project information and to make user authentications the VoD client applications make connections to the Mediation Server. By connecting to the server; first of all, the VoD client systems authenticates the end user by querying the user and password data base located in the Mediation server and after authentication the client gather information about the existing projects as shown in Figure 4-6. If the access rights of the users are acceptable, i.e. if the user is a learner, the learner should be able to interact with the projects in VoD Player and if the user is an administrator,

he/she should be able to edit the previous projects or create new ones in VoD Project Editor.

4.3 VoD Servers

The VoD System is designed to allow more than one Content Server to be configured per Mediation Server. There is no restriction to combine both servers in one computer but this will decrease the total system performance for the reasons described below:

- The network switch or the router connected to one server would be over-loaded on too many connections,
- The videos combining the VoD presentations will require enormous amounts of physical storage to be supported by one server,
- Maintenance of the projects must be handled by one administrator to protect other projects from deletion or unauthorized modification or to disallow copyrighted materials to be seen by other faculty members.

Instead of combining the servers, the use of distributed network architecture will increase the overall system performance Rousseau and Duda [1999]. The projects will also be “contained and maintained” in the Content Servers located within the respective faculties by the respective faculties’ administrators. An example of this idea is represented in Figure 4-9.

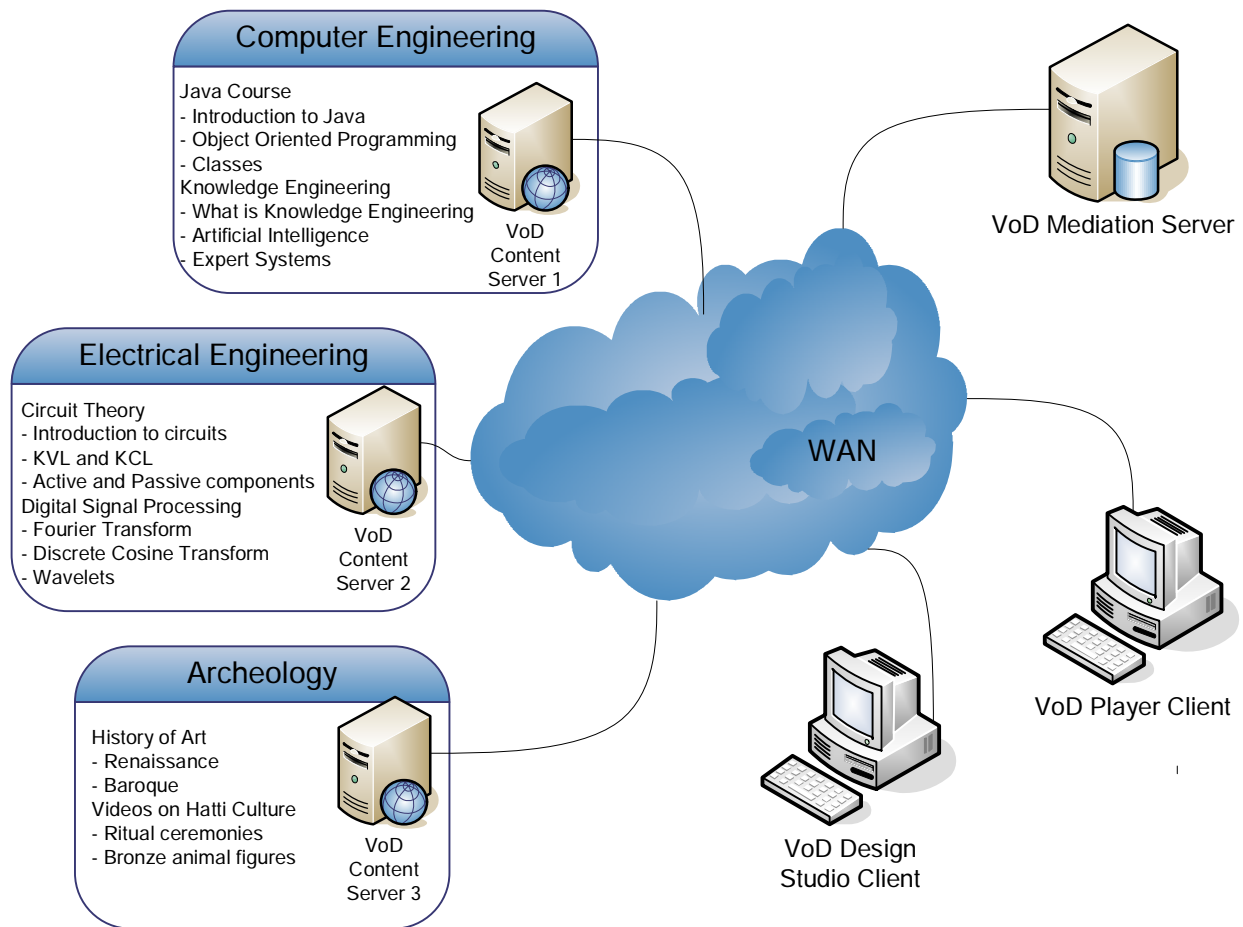


Figure 4-9 Typical VoD Server distribution

4.3.1 Mediation Server

Mediation server is responsible for storing VoD databases and authenticating the users. Database includes user information and other VoD projects related data. User addition to the system is made using the web interface of the VoD system; the passwords are encrypted and stored in the user database. The other additions to the database such as faculties, courses and projects are made using the VoD Project Editor as a compulsory step before adding a video to a project and also the Web interface of the VoD system allows the users to manipulate the database if the user is an admin privileged. Mediation server is independent from any operating systems (OS) because MySQL can run on Linux, FreeBSD, NetBSD, Solaris, SCO, Win32, NT and OS/2 systems. The

existing Mediation Server is built on Red-Hat Linux 9.0 system. Instead of MySQL any SQL based database management system could also be used with the Open Data Base Connectivity (ODBC) system. However this has not been implemented and tested in this thesis and for the sake of simplicity, the VoD system is currently only compatible with the MySQL relational database management system.

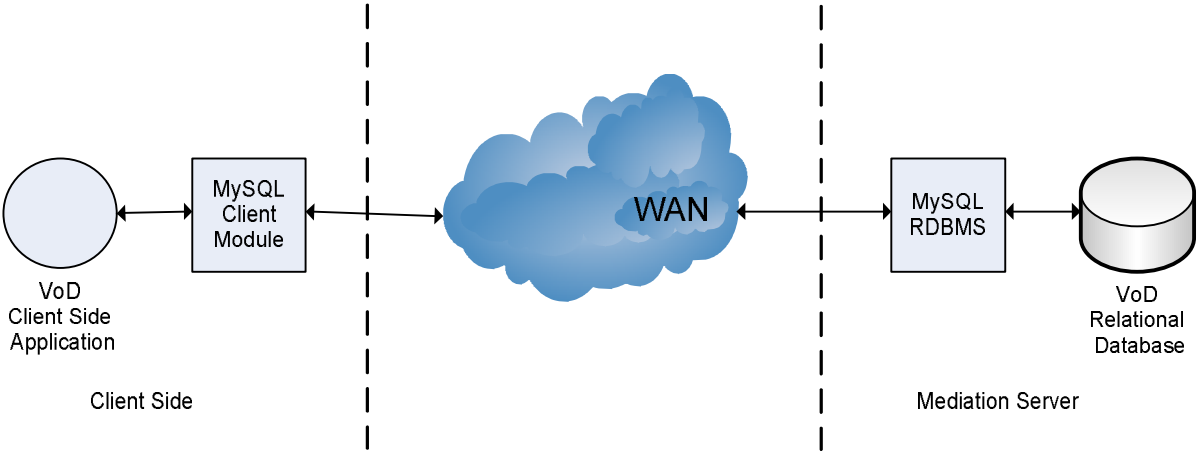


Figure 4-10 VoD Database connection handling

Figure 4-10 shows the database connectivity of the VoD client applications to the Mediation Server. The database connectivity of the client side application is handled by the MySQL library, which is called “libmysql.dll” and is statically attached on compilation of the application. MySQL is an ANSI-ISO-SQL-99 [1999] compliant Relational Database Management System (RDBMS) and VoD client applications use standard ANSI-SQL-99 [1999] commands for sending and receiving data to/from the Mediation Server.

The tables combining the VoD database is shown in Figure 4-11. The figure illustrates the relation between the tables and PK refers to abbreviation for Primary Key and refers to abbreviation for Foreign Key. The details of the tables in Figure 4-11 are

given in the Appendix B.

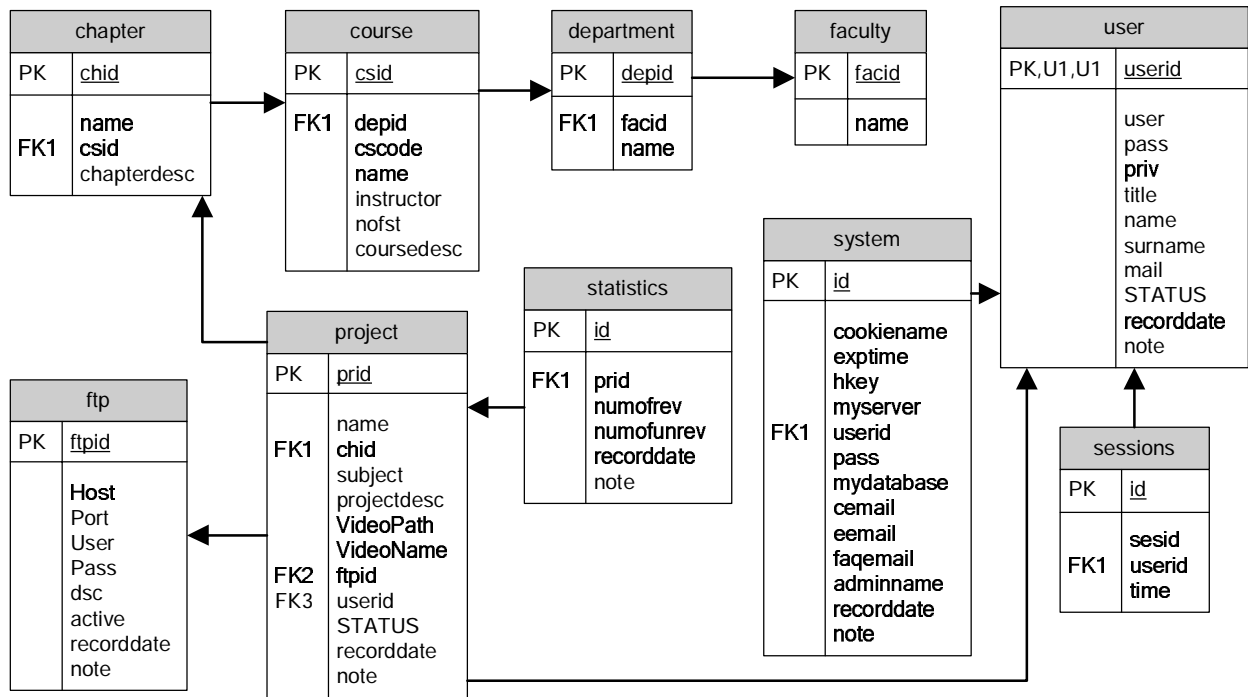


Figure 4-11 VoD relational database organization

Figure 4-11 shows the basic database structure of the VoD system needed on the Mediation Server. The main object for the system is the “Faculties” which encapsulates all the data about the projects by applying the hierarchy. Access to the projects from each faculty is restricted only to their own data, hence making the manipulation of the projects easier.

As it can be noticed from the Figure 4-11, the synchronization data of the content material to the video are not stored in the database; the reason for this is that the size of the synchronization data may vary, and it is more efficient to store these data items in the content servers.

4.3.2 Content Servers

Content servers are designed to serve VoD project videos, the related content materials about those videos and the synchronization data that combines the videos and the content materials as shown in the Fig. 4.12 below.

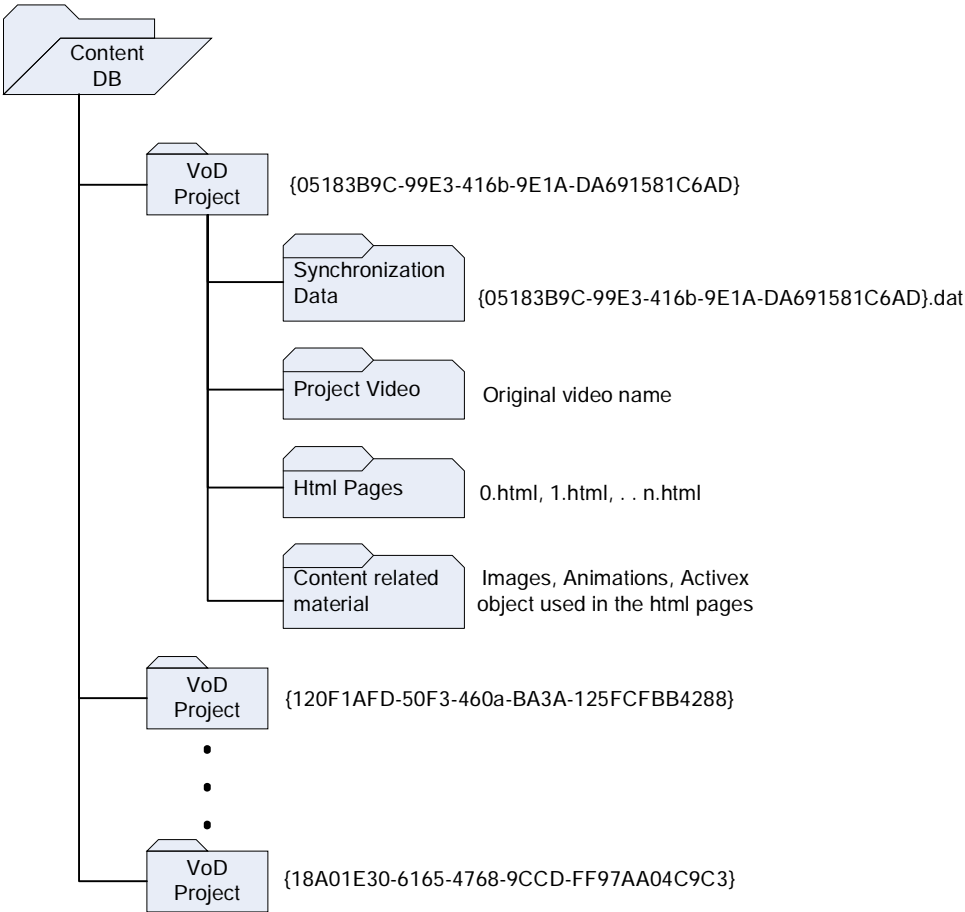


Figure 4-12 VoD Content Database structure

Only VoD Project Editor is authorized to store content to the Content Servers, and the VoD Content database is designed in a structure only understandable by the VoD client and server applications.

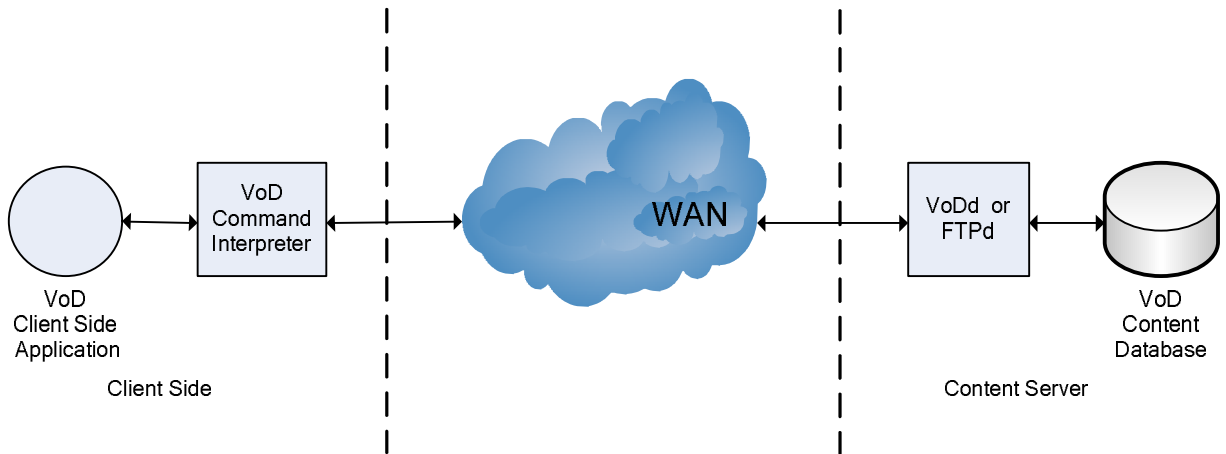


Figure 4-13 VoD Content Database connection handling

The underlying network actions are handled by the VoD Application Protocol (VAP) which is built into the VoD client side applications as shown in

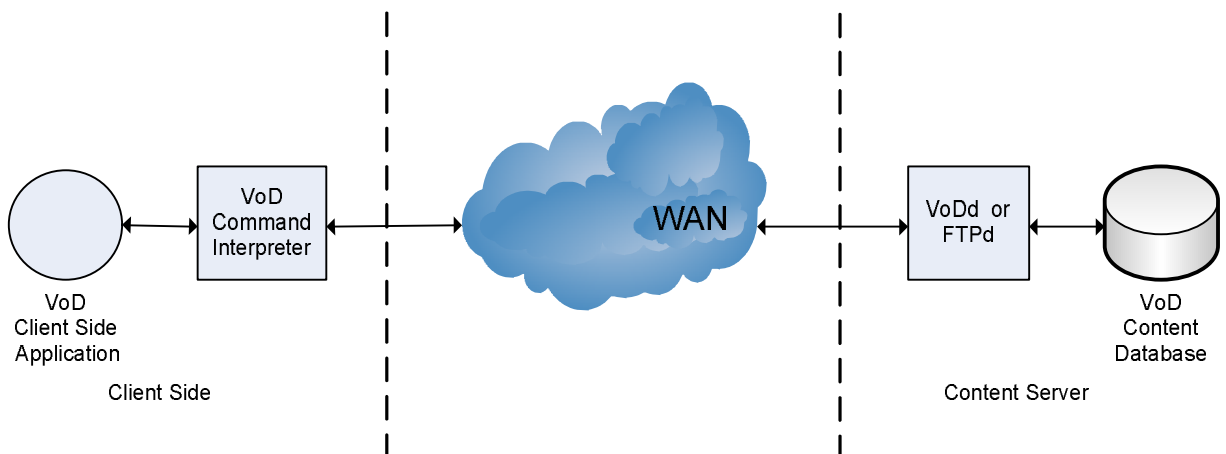


Figure 4-13 VAP translates the high level VoD commands to set of functions to build network connections in a range of communication protocols. On every request for project presentations in VoD Player, the same request is sent to the VAP without taking into account the type of transport connection protocol, the decision of the transport protocol to be selected is handled by VAP. The VAP makes connections, sends FTP commands to download related content material and downloads the video with the preferred connection protocol.

Content servers could be distributed and every server could be dedicated to a faculty: this would improve the total system performance. Nowadays video stream rates vary between 256 kbps to 100 Mbps depending on the quality and compression technique that is used. To fulfill the streaming requirements, a content server should be directly connected to an ATM switch. A typical ATM enabled server could serve its content at up to 155 Mb/s and theoretically this kind of server should support at least 200 connections if we take an average of 0.5 Mb/s for the video streams. But the worst case conditions always define the system performance; hence the worst case condition might be the disk speed, network speed processor speed, bus speed or the overall speed.

Content servers require three server applications to run, these are; a commercial third party FTP server on port 21, VoD TCP Server on port 21000 and VoD UDP Server on port 22000 noting that RUDP services are also handled by the VoD UDP Server application. The FTP server could also be developed in-house, however since there are a large number of proven FTP server applications (FTPd), this is not necessary and it is not undertaken in this thesis project.

The synchronization data stored in the content servers are created by the VoD Project Editor and are sent to the servers when the project is uploaded to the servers. The C structure for the synchronization data is given in the Appendix A.

4.3.2.1 Implementation of the Content Server Applications

The Content Server applications are programmed by using C++ and MFC, but it has no

necessity to mention the content server applications as platform dependant; MFC is only used to simplify the user interface programming and the functional part of the applications are compatible with the *NIX operating systems. The fundamental jobs handled by these applications are to;

- Authorize client applications; VoD Player and VoD Project Editor.
- Serve videos to the clients using FTP, TCP, UDP and RUDP based transfer mechanisms.
- Serve additional content materials and their inbound objects (pictures, animations or the other components added to the html pages) by FTP.
- Store the VoD projects sent by VoD Project Editor.

Since the UDP Protocol is a connectionless, non-reliable protocol, the UDP server does not make connections to the clients; therefore, sending and receiving the data differs from the connection oriented protocols. Instead of making connection or defining a constant path to the client, the UDP server assigns all the calls coming from a specific client to one of its worker threads. Thus, every client application connected from a computer is assigned a thread number in the UDP server application and the authentication of every client in every session is done only once. The threads earmarked to give this service are created as empty handlers when the server application initializes.

In an effort to alleviate the unreliable nature of the UDP protocol, some additional proposals are made for creating a Reliable UDP protocol. The main additions are the connection establishment, acknowledgement and packet numbering. UDP server also handles the requests for Reliable UDP sessions, the most important difference between both is; RUDP is connection oriented and uses a tiny header pasted to the data sent to the clients and the clients send back a one byte (without UDP

header) acknowledgment to complete the process. The packet and header format of the proposed RUDP protocol is given in the Appendix A.

For the VoD system, TCP connection for sending the videos is also handled, the reason for that is to analyze the effects of the FTP protocol and calculate the overhead made by the FTP layer and compare all the protocols that work over the IP layer. The Flowcharts of the servers are given in Fig. 4.14.

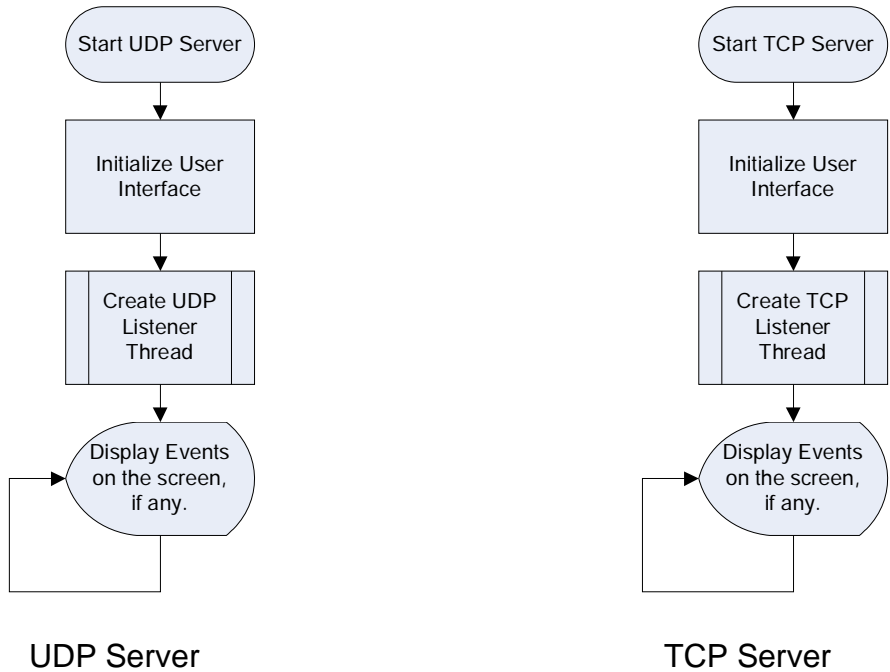


Figure 4-14 Content Server application’s main threads

Figure 4-14 shows the initialization processes of the VoD UDP and TCP server applications. The listener threads on both servers are bound to a specific port where any VoD client response could come. The ports 21000 for TCP traffic and port 22000 for UDP traffic are selected.

The Figure 4-15 is the flowchart of the UDP redirection system, the redirection system is used to avoid many threads to be handled by one client. If the clients’ VoD

application initializes a connection, a unique thread is assigned to that client's application and the further communications are handled by this thread till the end of the session. The advantages are; the threads on the UDP server are created when the UDP server initializes itself and the thread creation time is saved, the other advantage is; when the client is bound to a thread, the thread stores the parameters of the session, such as; authentication of the VoD client application, selected projects folder location, etc. Thus, there is no requirement to authenticate the VoD client application every time in a session and the authentication flag remains set in a thread till the end of the session.

The UDP server may decide to end the session, if the session time outs, (where the time out value is hard coded to the VoD UDP server application as: 600 seconds and could be changed with a new compilation.) or if the VoD client application sends a QUIT command to the working thread.

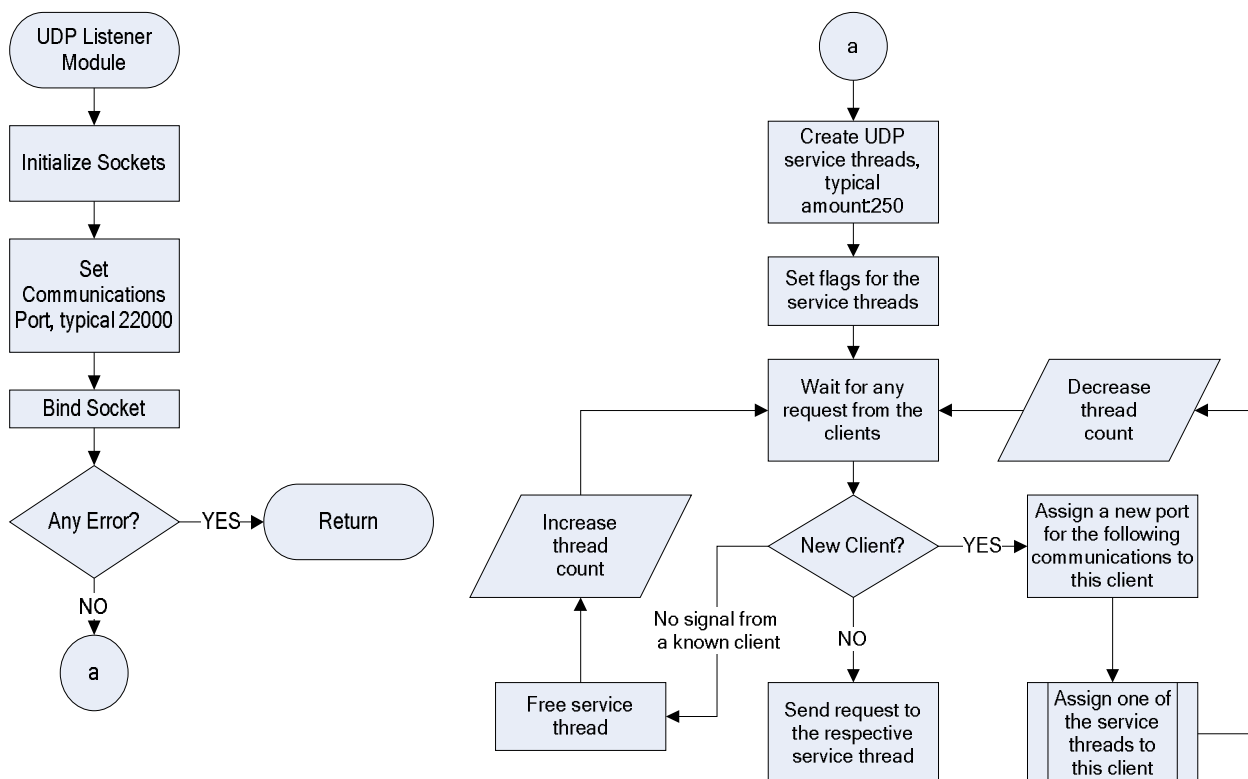


Figure 4-15 UDP Server message redirection system

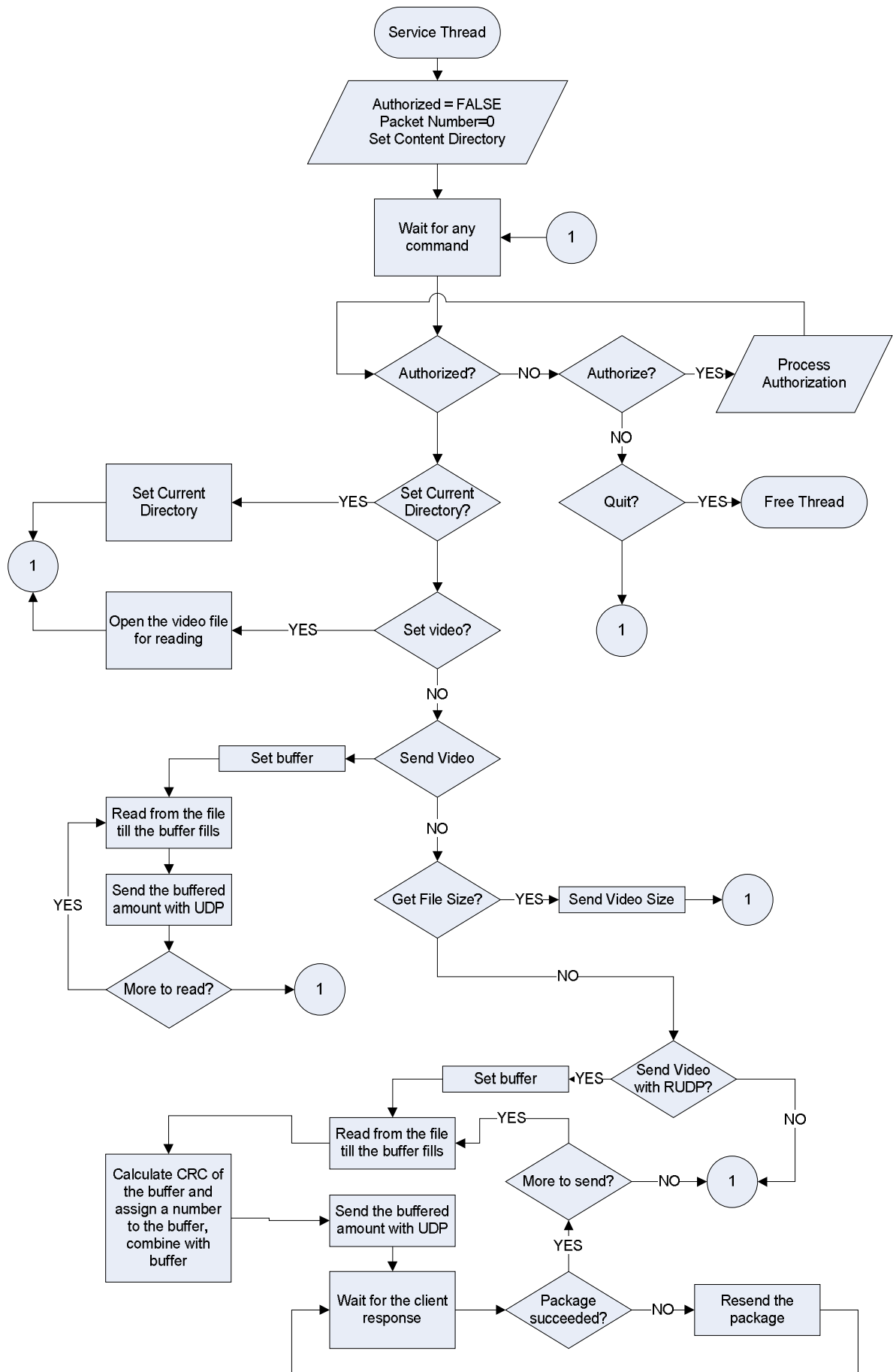


Figure 4-16 UDP and RUDP service thread

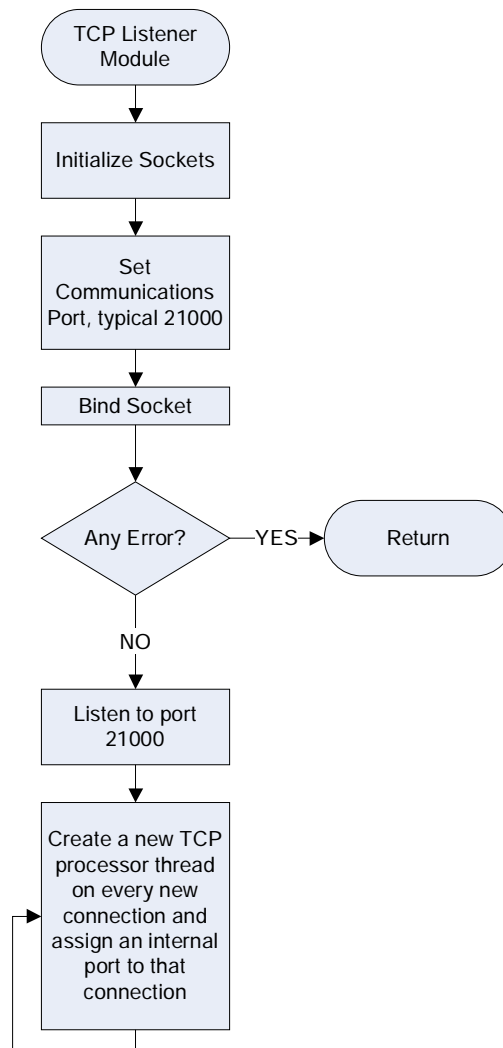


Figure 4-17 TCP listener module

The major difference between the TCP and UDP listener modules is the “listen”ing of a socket. Pathway to listen a socket on UDP differs from TCP; when a client connects to the TCP socket, all the routine tasks are automatically handled by the underlying system but in UDP; all the packets coming to the communications port must be selected by the main thread of the application in order to send them to the respective worker threads of the UDP server.

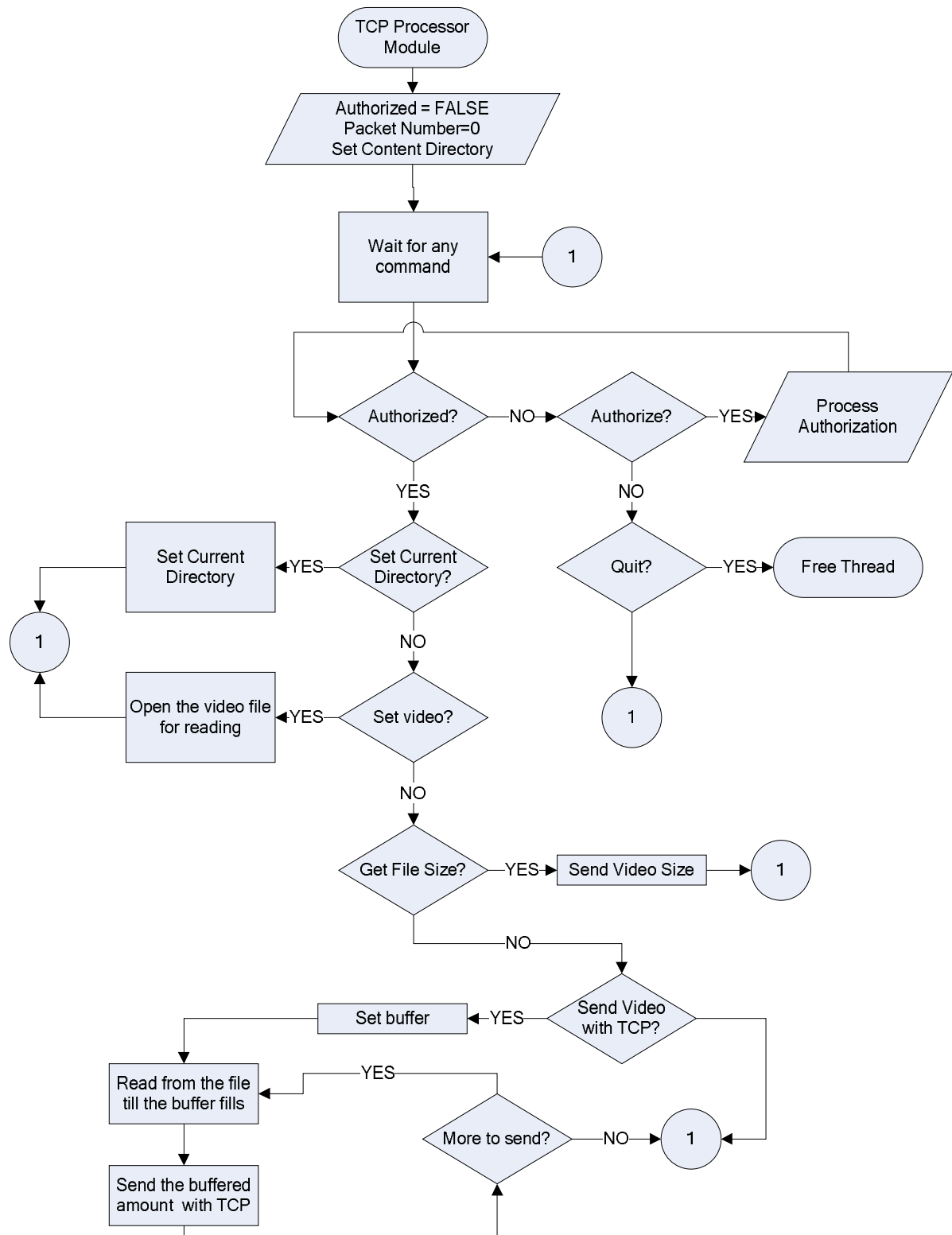
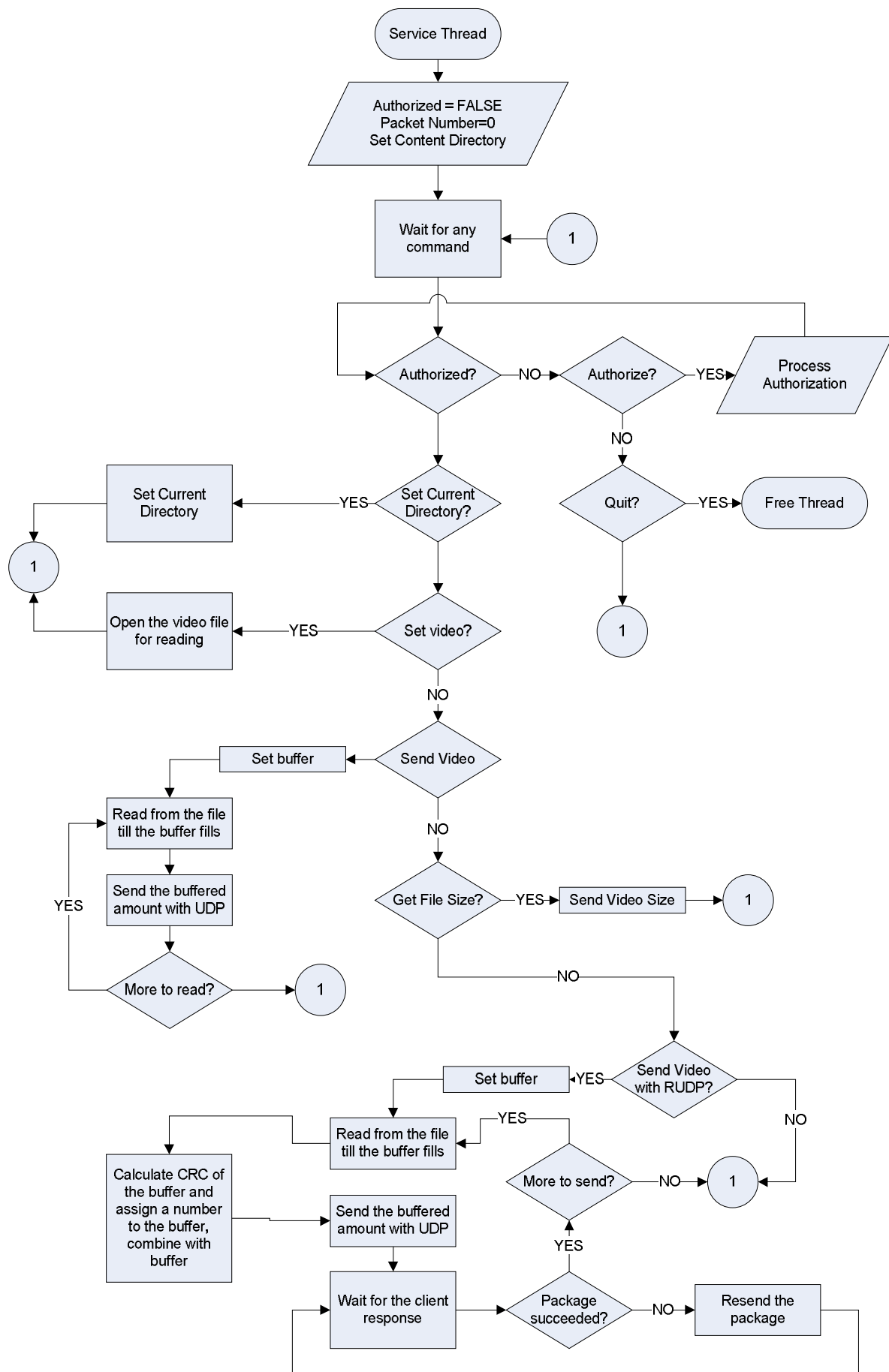


Figure 4-18 TCP processor module

As noticed from the figures, the processes do not show too many differences for different protocols but it is impossible to say that they should give the same guarantee

and the speed for the arrival of the data to be sent.

In



Figur

re 4-16, RUDP sub-routine acts as a simple TCP protocol, and emulating the OSI-RM layer 4 (transport layer) in OSI-RM layer 7 (application layer), TCP/IP protocol in its nature, includes very sophisticated routines such as Nagle Algorithm [1984]. Video transmission over Campus Wide Network does not have any small packet problem. RUDP somehow could be described as a light version of TCP emulated on the application layer.

The screen shot of VoD TCP Server and VoD UDP server are given below in Figure 4-19 and Figure 4-20 respectively. Please note that the UDP server handles both UDP and RUDP requests.

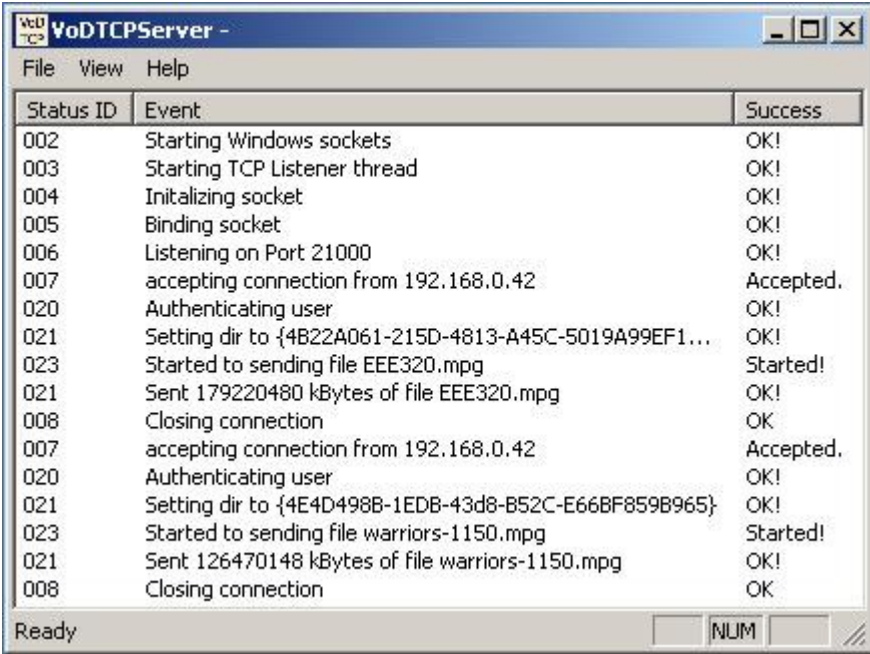


Figure 4-19 VoD TCP Server

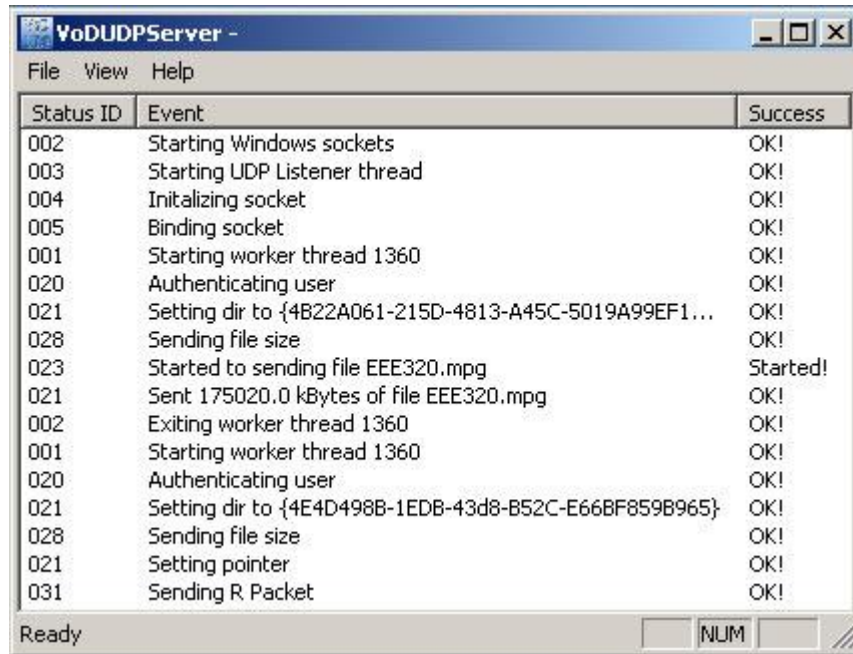


Figure 4-20 VoD UDP Server

4.4 The VoD Clients

The VoD system requires client side applications to make presentations of the designed projects. The clients of VoD system are classified into two:

1. VoD Project Editor
2. VoD Player

The VoD Project Editor is responsible for the editing, designing, sending and registering the projects to the server. The project editor can only be used by system administrators and instructors; other entries to the system are simply discarded. Details of the VoD Project Editor will be covered in Chapter 5, since this relates to technical as well as pedagogical issues.

4.4.1 Implementation of the VoD Player

VoD Player is responsible from user sign-ups, acquiring user and project data form the Mediation Server, streaming the project videos with the available internet protocols and downloading and presenting the content related materials. The properties and the differences between the VoD Player and the Project Editor systems are represented in the Table 4.2 below:

Property	User authentication	Project editing	Video player	Html browser	Html editor	Protocols	Database connection
VoD Project Editor	•	•	•	•	•	FTP, TCP	•
VoD Player	•	X	•	•	X	FTP, TCP, UDP, RUDP	•

Table 4.2 Comparison chart for client side applications

One of the most important issues in a player are that it must be easily understandable by the user and can easily be used even by novices. Therefore, VoD player needs to heavily use the user interface capabilities of the operating system. To implement the typical VoD system facilities, the player includes basic play/pause, stop and project opening and information buttons on its main screen as shown in Figure 4-22. To log into the player system, a user must be authenticated through the VoD Mediation Server. A typical login screen of the player is shown in the Figure 4-21 and is activated by program run. Host IP is the IP of the Mediation Server, username and password are entered for login. The passwords is encrypted and send to the mediation server using MySQL API.



Figure 4-21 VoD Player authentication screen

After the authentication of the user the VoD Player main window shown in Figure 4-22 opens up on user terminal and player gets the user information from the Mediation Server, initializes the system by;

1. DirectX initialization
2. Calculation of the available resources
3. Gathers project information from the mediation server
4. Opens the statistics pane if requested.



Figure 4-22 VoD Player main window

The buttons starting from left to right are; play/pause, stop, project selection and information. In the absence of an open project the play and stop buttons are disabled. When the main screen initializes, the user has only two options: either she can select a project or can configure system options from the information button.

The information button contains configuration, statistics, disconnection and about menus. In the configuration menu, user can select the preferred connection protocol and can select local buffer sizes for video buffering; optionally could connect a different content server to the system. Since the statistics pane takes up resources, it is made an option to enable or disable the pane.

When the project selection button is pressed, a new window is opened and the available project resources are taken from the Mediation Server once again. This new window gives the project information in a hierarchical way where all the projects are categorized by their faculty, department, course and chapters. Figure 4-23 gives an example of this hierarchy.

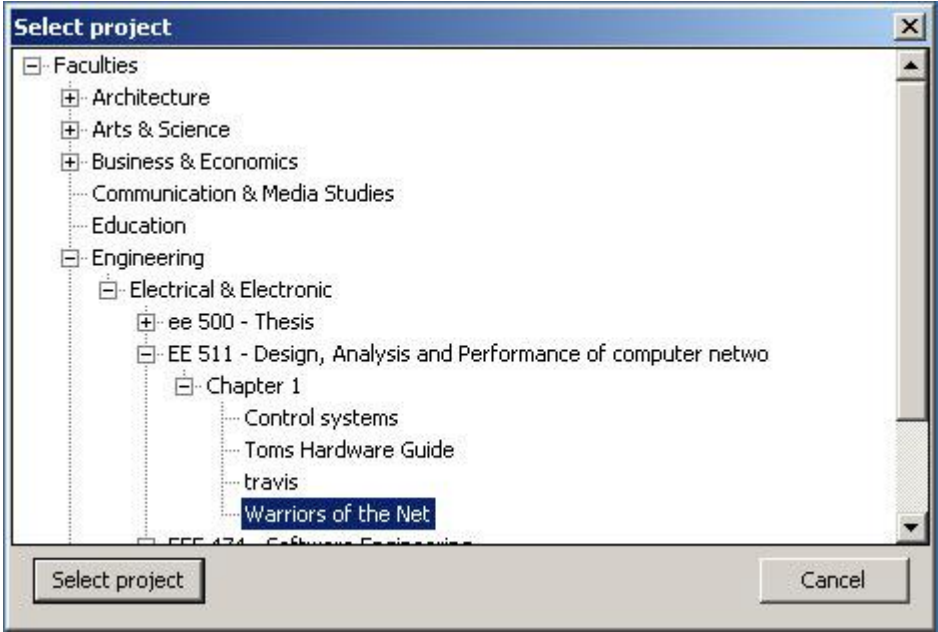


Figure 4-23 Project selection window

After selecting a project, VoD Player gets all the necessary information from the Mediation Server such as project name, video name, Content Server IP address, project folder location in the Content Server and starts downloading the course related content material from the Content Server using an FTP process by these steps; (Detailed diagram shown in the Figure 4-6)

1. FTP process logs in to the Content Server by authenticating itself to the server
2. Sets the remote directory as defined in the Mediation Server database
3. Downloads all the files, i.e. html pages, animations, pictures etc from the server

The screen shot of this process is given in Figure 4-24;

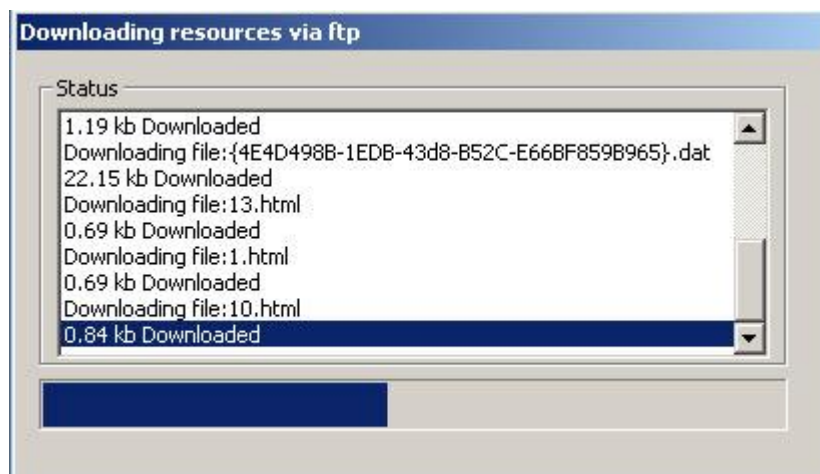


Figure 4-24 Content material download by the FTP process.

The system does not start video application before downloading all the related documents and the synchronization data. The video progress bar uses this synchronization data to put the related materials in their desired timings. The synchronization data is formed automatically by the VoD Project Editor when uploading the projects. After the downloading process finishes, the VoD player selects the preferred media connection protocol and connects to the content servers to request the videos. A screen shot of the main video screen is given in Figure 4-25;

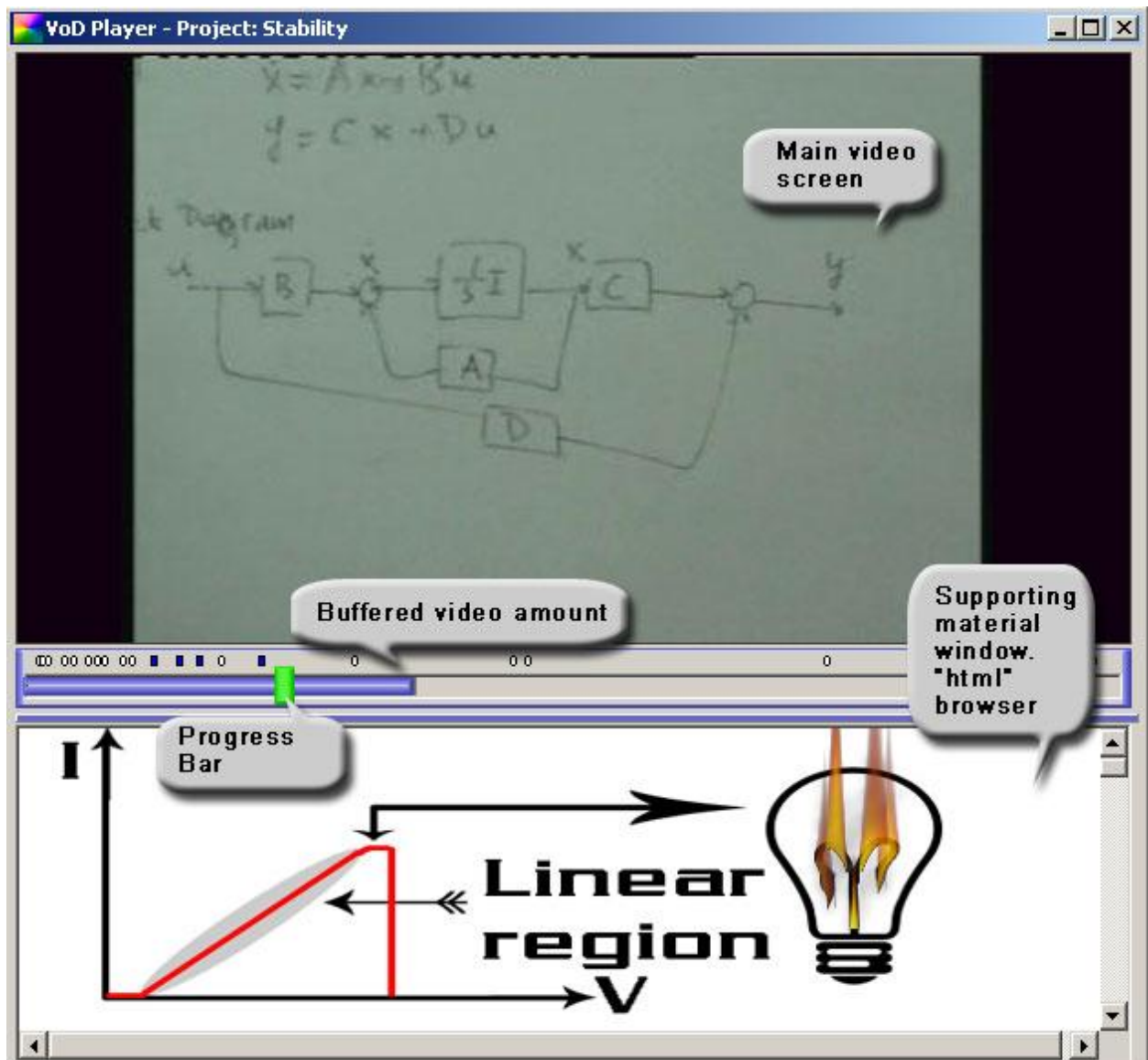


Figure 4-25 VoD Player Video and html browser screen

The VoD Player main video screen consists of a video window, a buffer indicator, an html browser to present video related content material including user contributions and a progress bar whose ticks formed as ellipses are representing the original content related materials which are added by the instructor of the project designed with the VoD Project Editor and the rectangular (blue) tics represents user additions to the project.



Figure 4-26 User contribution drag and drop box

Figure 4-26 shows the pause and drag box of the VoD Player which handles the addition of user contributions by simply dragging the additional users selection of html pages, text files, images, animations even ActiveX objects such as Microsoft PowerPoint slides and Adobe Acrobat pages. The additions are copied from their destination to VoD content folder located inside a special Windows folder called "Application Data". Those user additions will be shown like the content related supporting materials shown in the html window of the VoD Player. The user contributions are stored in the user computer as the same way they stored in the Content Server and in the next openings of the projects, the VoD Player searches for the user contributions in client computer. If any contribution could be found, the player adds them to the project in the user terminal and marks them in the slider bar as rectangular (blue) tics. The user contributions will be described in detailed in Chapter 5 since this subject relates more on pedagogical issues.

Downloading and playing of video clips are done in the streaming mode. That is after a certain amount of buffering (which will be described later), the VoD Player starts to play the (educational) video in its main screen and if there are any additional content materials, these are shown in its html browser. While playing the movie, the end user could rewind, forward, pause or stop the video and if there is any internet link in the content material, this link could be opened inside the application browser or outside with a commercial internet browser.

4.4.2 System Model of the VoD Player

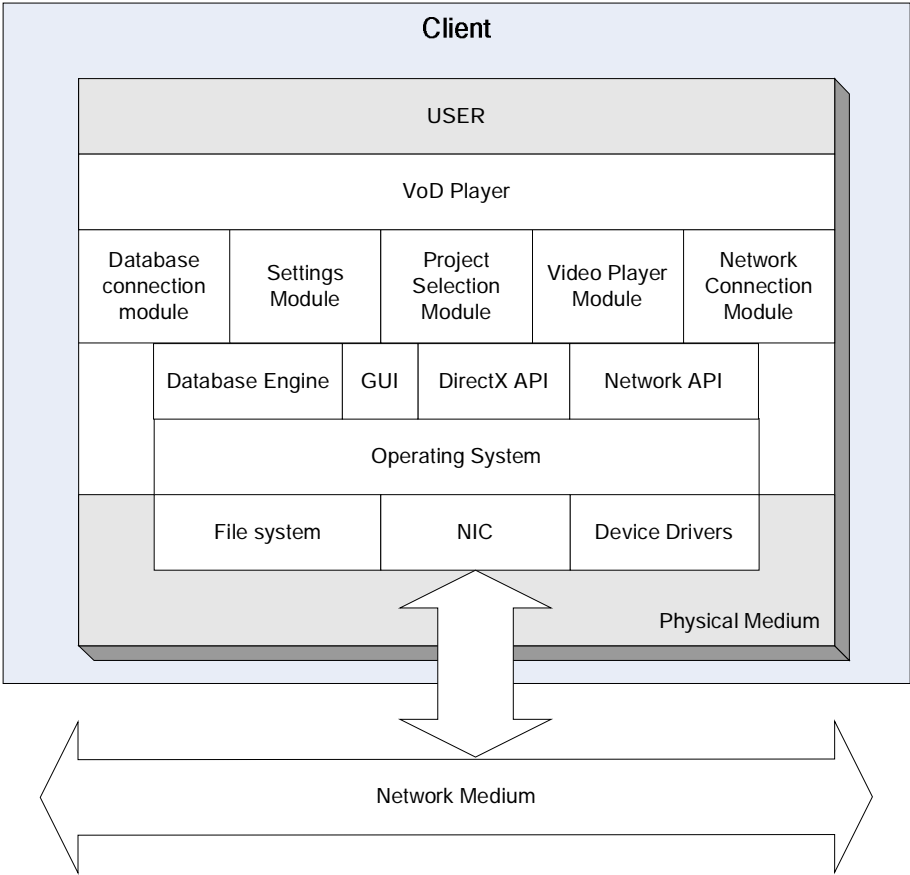


Figure 4-27 Client application diagram

Figure 4-27 depicts the hierarchical diagram of the client side application; the VoD Player.

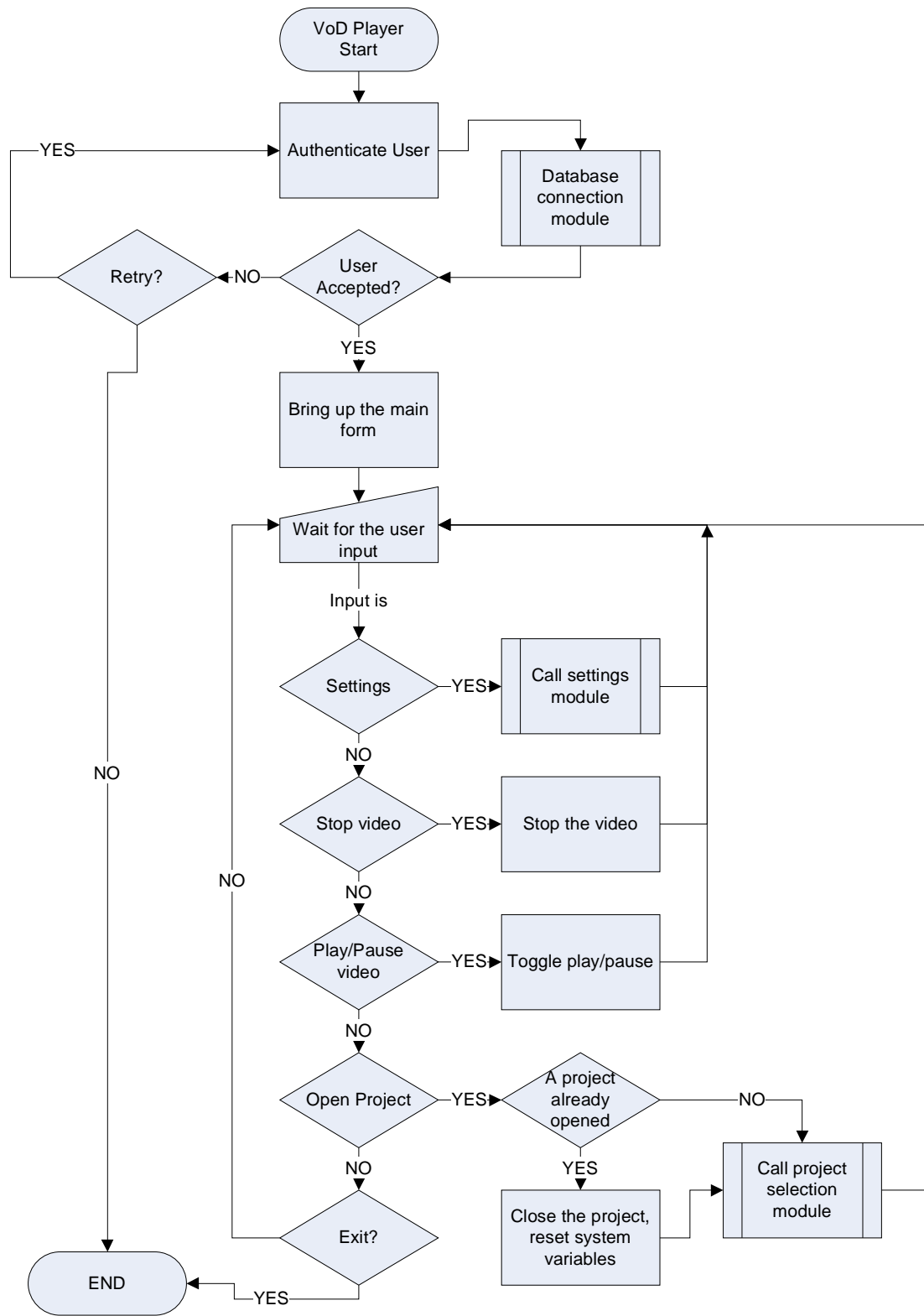


Figure 4-28 Main flowchart for the VoD Player

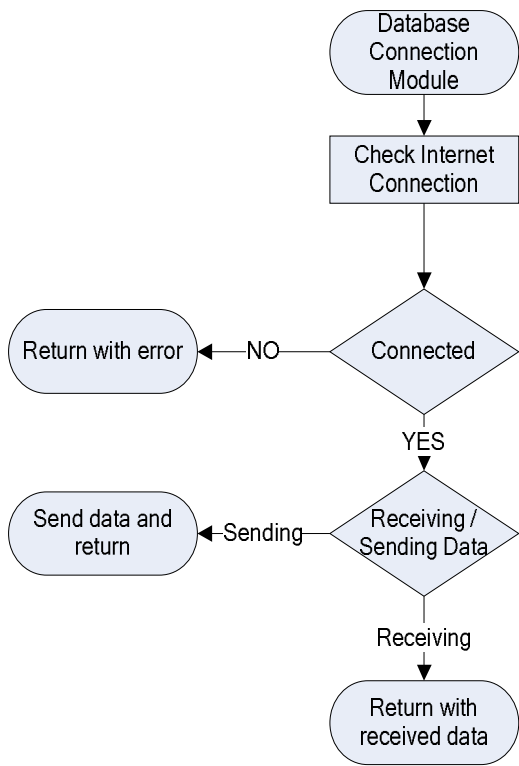


Figure 4-29 Database Module

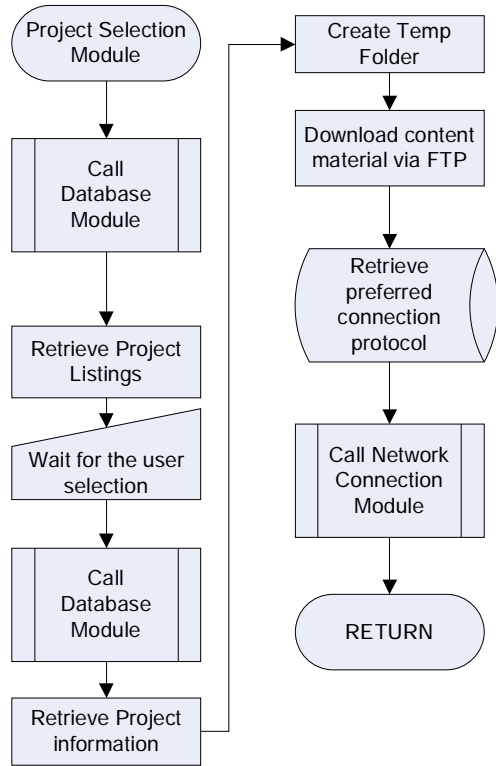


Figure 4-30 Project Selection Module

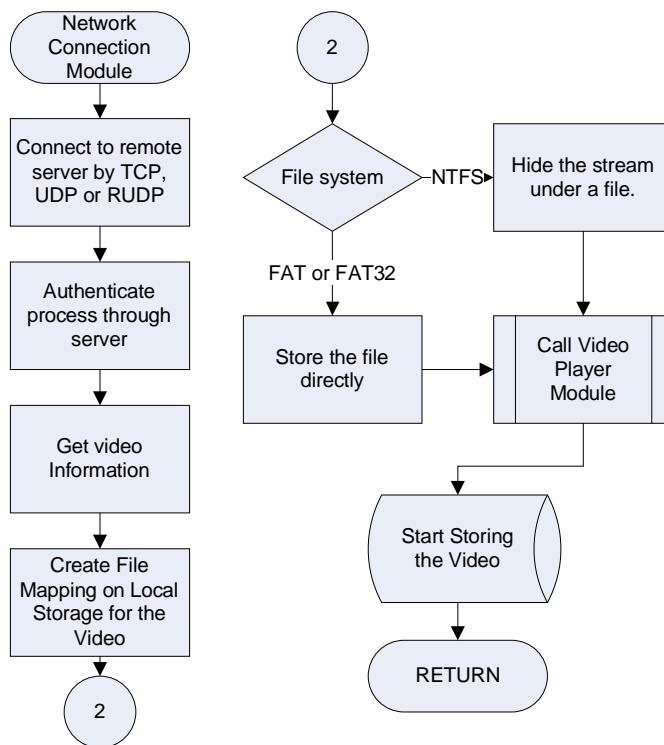


Figure 4-31 Network Connection Module

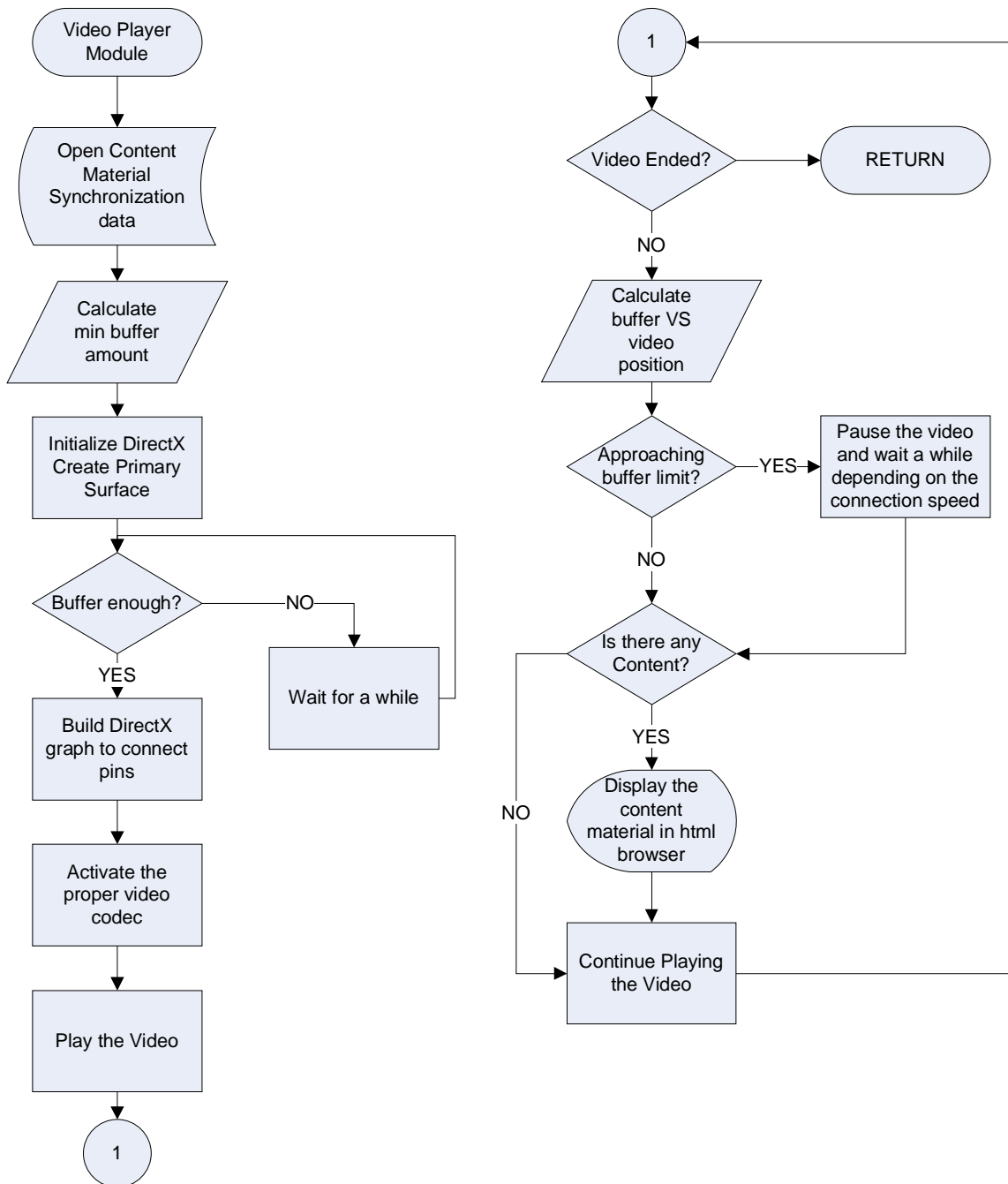


Figure 4.30 Video Player Module

4.5 System Characteristics

4.5.1 Streaming, Buffering and Memory Mapped Files

Streaming refers to the nearly instant presentation of the multimedia content while downloading it over the network. This technique is heavily used because of the large sizes of the multimedia data. In ordinary bulk data transfers through the network, an application must wait for the download to complete. This situation has many reasons;

- The downloading application may not allow file sharing
- The multimedia presentation application could not extract the multimedia header,
- Some multimedia data structures include its header at the end of the file, i.e. for some AVI types, index regeneration is required.

VoD Player includes its sophisticated streaming engine by using the all specialties of the underlying file system. VoD Player could play multimedia streams tested over 2 Mbps while downloading the multimedia data. This ability comes from the parallel usage of the memory and the physical storage at the same time. Memory mapping is the base technology which allows this situation to happen.

Memory mapping of files allows physical storage accessibility with “C” style pointers as made in the dynamic memory allocation and gives the ability to the programmer to divide the physical storage to segments as done in memory arrays. By this technology a file could be handled for reading and writing from different processes of an application at the same time. In normal file operations this action should give file access violation error. More on managing memory-mapped files in Win32 is covered in the MSDN article [1993].

VoD Player, in its nature has to store the video in the client side computer but it is automatically deleted when the user exits the player and the reasons to store the projects are:

1. Since the content includes education videos, it is expected that a project could be viewed several times,
2. Nowadays physical storage is getting less than 1gB/\$,
3. New compression techniques reduced the size of the multimedia files.

The stored project files are automatically deleted when the user closes the project. The deletion process is a required step in order to free the allocated hard-drive space. In the cases where an interruption occurs to VoD Player, the stored files may not be deleted. For this reason the files are stored in the temporary folder of Windows and could be deleted by the system maintenance. If the project files stay resident in the client computer for any reason, they could not be copied by the user, because VoD Player hides the video stream under a file which has absolutely different purpose and the end user will face with a sixteen byte file instead of a huge video data.

Before starting to download a multimedia file, the VoD Player acquires the multimedia data size in order to allocate the whole multimedia space in the physical storage. There are two reasons for this:

1. Memory mapping of files requires full allocation of the required space because, some file systems do not allow expansion of the allocated space,
2. Some AVI type's headers are stored at the end of the file and instead of regeneration of the AVI index, the end octet of file is downloaded first, because index generation takes more time.

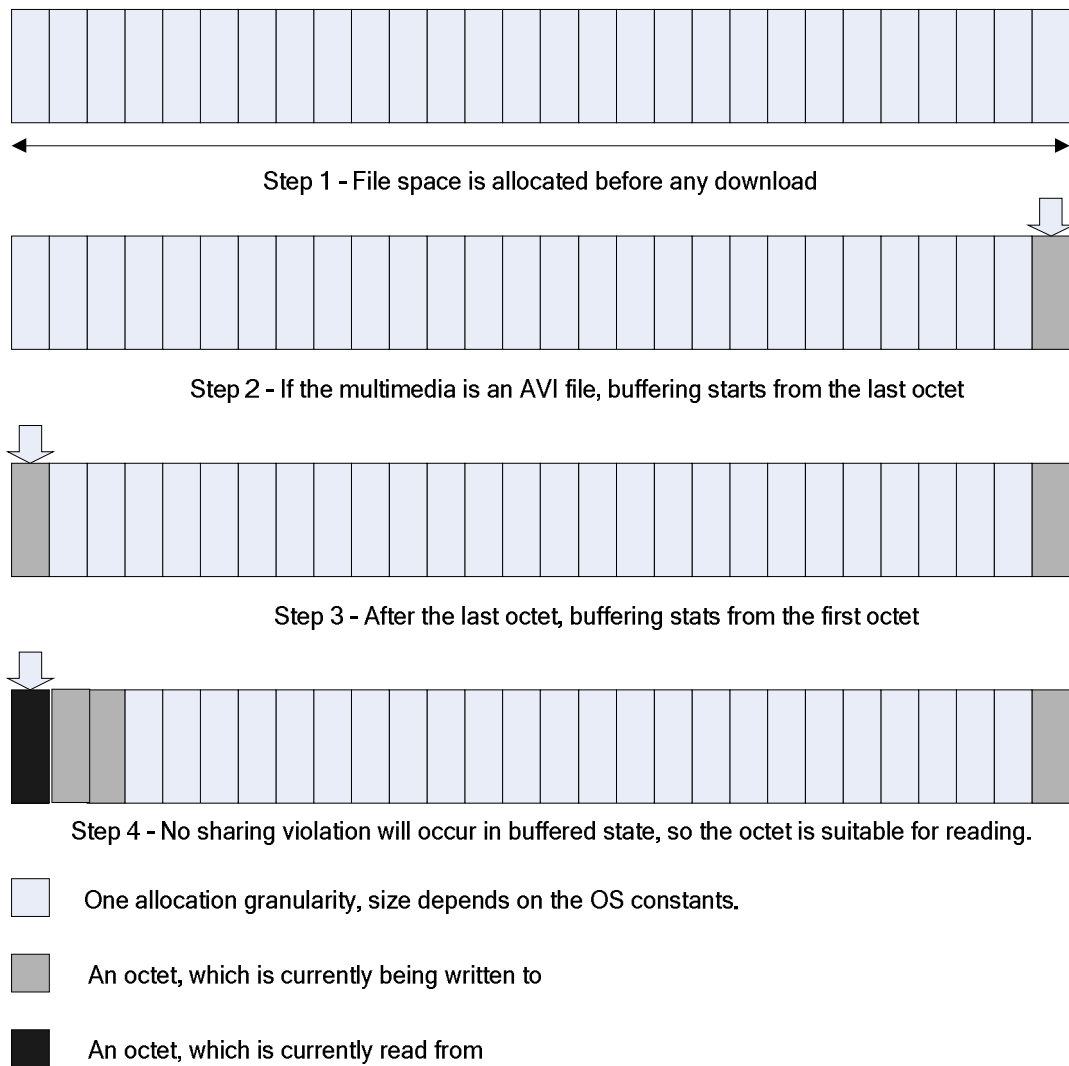


Figure 4-32 Buffering and playing steps of VoD Player

Figure 4-32 shows the steps of buffering, streaming and playing a multimedia file using the VoD Player. Since the buffering window is dependant on the system allocation granularity, the minimum buffer before starting to play the multimedia file will not be less than system allocation granularity. In testing the workstation the granularity was 65535 bytes. This means at least 65 Kbytes of buffer is required for the video playing operation.

The buffering and playback operations are handled independently by two threads in order to maximize the total application performance. The playback thread,

until a signal comes from the buffering thread, waits in its suspended mode. After the retrieval of the permission from the buffering thread, the playback thread initializes the DirectX processes for the video to be played. Those steps could be described as a flow chart as could be seen in Figure 4-33:

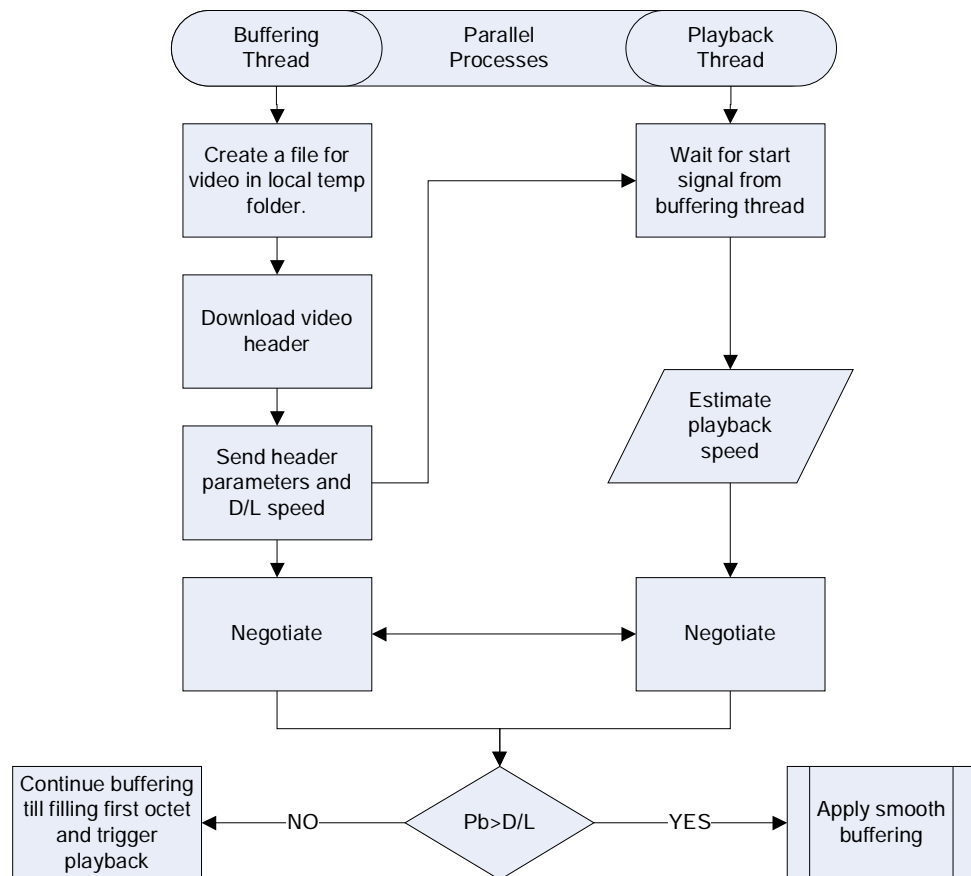


Figure 4-33 Buffering and Playback Processes

D/L: Download

4.5.2 Smooth Buffering

If the video playback speed is greater than the speed of download buffering, the VoD player activates a “Smooth Buffering mechanism” for the presentation. This buffering style gives the end user the chance to watch the presentation without any interruption. The system buffering and playback operation using smooth buffering is shown in Figure

4-34.

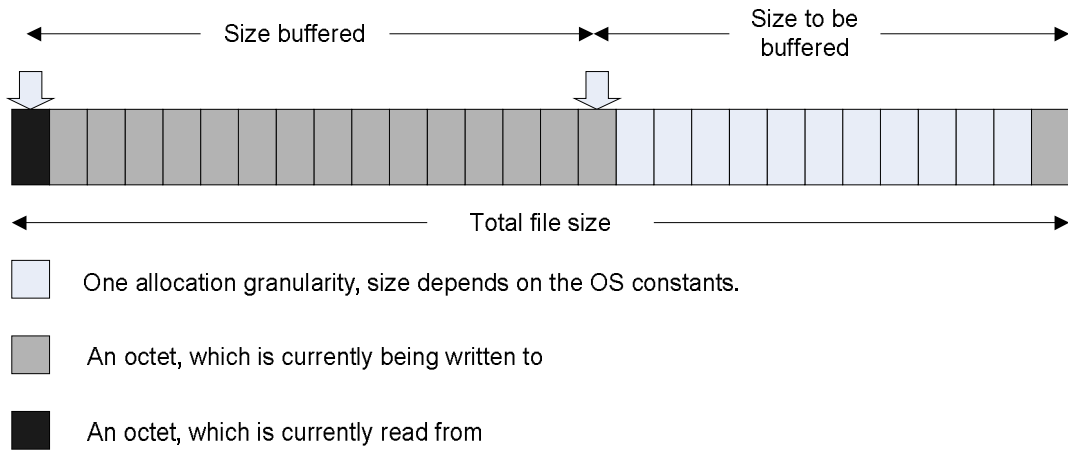


Figure 4-34 Smooth Buffering strategy

The calculation for smooth buffering could be defined in the terms described below:

Parameters	Alias	Description	Unit
Download Speed	S_D	Speed of the network transmission medium	Mbits/s
Playing Speed	S_P	Video frame rate	Mbits/s
Frame Rate	F_R	Number of frames to be displayed per second	Frame/s
Granularity	G	OS file allocation granularity	Bytes
Playing Time	t_P	Total video playing time	s
Waiting Time	t_W	Amount of time to be waited before playback operation	s
Unbuffered Chunk Time	t_U	The estimated time to complete buffering operation	s
File Size	F_S	Size of the video to be presented	Mbits
Total Time	t_t	Required time to complete buffering	s

Table 4.3 Parameters used to calculate initial buffer

The playing time t_P can be expressed as follows:

$$F_S = S_P * t_P$$

$$\begin{aligned}
F_S &= S_D(t_W + t_P) \quad \text{or} \quad F_S = S_D * t_t \\
S_D * t_W + S_D * t_P &= S_P * t_P \\
S_D * t_W &= S_P * t_P - S_D * t_P \\
t_W &= \frac{t_P(S_P - S_D)}{S_D} \quad \text{and} \\
t_t &= t_W + t_P \\
t_t &= \frac{t_P(S_P - S_D)}{S_D} + t_P \\
t_W &= \frac{t_P * S_P}{S_D} - t_P \Leftrightarrow 0 < S_D < S_P \\
t_W &= \begin{cases} 0 & : S_P < S_D \\ \frac{t_P * S_P}{S_D} - t_P & : S_P > S_D \end{cases} \Leftrightarrow 0 < S_D \quad \text{(Eq. 4.1)}
\end{aligned}$$

An example to calculate the waiting time could be as follows, when it is assumed that:

- Media file size: $F_S = 300$ Mbytes which will correspond to: 2517 Mbits, (F_S does not include video header and audio sizes.)
- Download Speed: $S_D = 0.8$ Mbits/s,
- Playing Speed: $S_P = 1.0$ Mbits/s,

The total playing time is given by $t_P = 2517$ seconds and the waiting time t_W should be:

$$t_W = (2517 * 1 / 0.8) - 2517$$

$$t_W = 629 \text{ seconds.}$$

If the playing operation reaches the octet that is currently being buffered unexpectedly because of network connections or any other reason, the multimedia presentation is paused automatically until the buffering process leaves the octet to Free State, i.e. the share protection over the octet must be removed.

5 PEDAGOGICAL ENHANCEMENTS FOR THE VOD SYSTEM

Stored video or VoD systems are useful but are rarely used on their own for Distance Learning (DL). There is a need to provide video as part of a complete DL environment. This environment may have different forms. One approach is to provide video clips when and where necessary for demonstration purposes within a text based learning environment. In this thesis, the approach is taken reverse. It is envisaged that pedagogically enhanced video can be very powerful learning environment. In this approach the video session covers the main event, or an actual presentation on the topic. The instructor responsible from delivery of the DL package utilizes the additional html browser space which is synchronized to the video for adding explanatory learning materials.

The additional learning material that can be added and synchronized to a video session include HTML, XML, DHTML based mark-up text, Virtual Reality Modeling Language (VRML) based visualization material, hyperlinks, ActiveX components such as Microsoft Office components, Macromedia Flash and Adobe Acrobat, sound, etc. These materials can be used to annotate the video based course material in order to bring the related content to the attention of the learner. In this way, the learner gains two advantages:

1. Refer to the additional material on the topic;

Gain an insight to the pre-requisite or more advanced topics related to the issues discussed in the video clip,

2. Provides computer based additional content for visualization and learning.

An extension of this idea is that each learner could also make notes and add further links to the video material in addition to those that are created by the instructor or author. This allows personalization of the video clip. Implementation of this idea necessitates the use of indexing and relating additional content material based on user profiles. VoD Player allows user content additions to a DL project. The user additions are added to the project by blending synchronization cycle of the current project with the user contributed materials as shown in Figure 5-1.

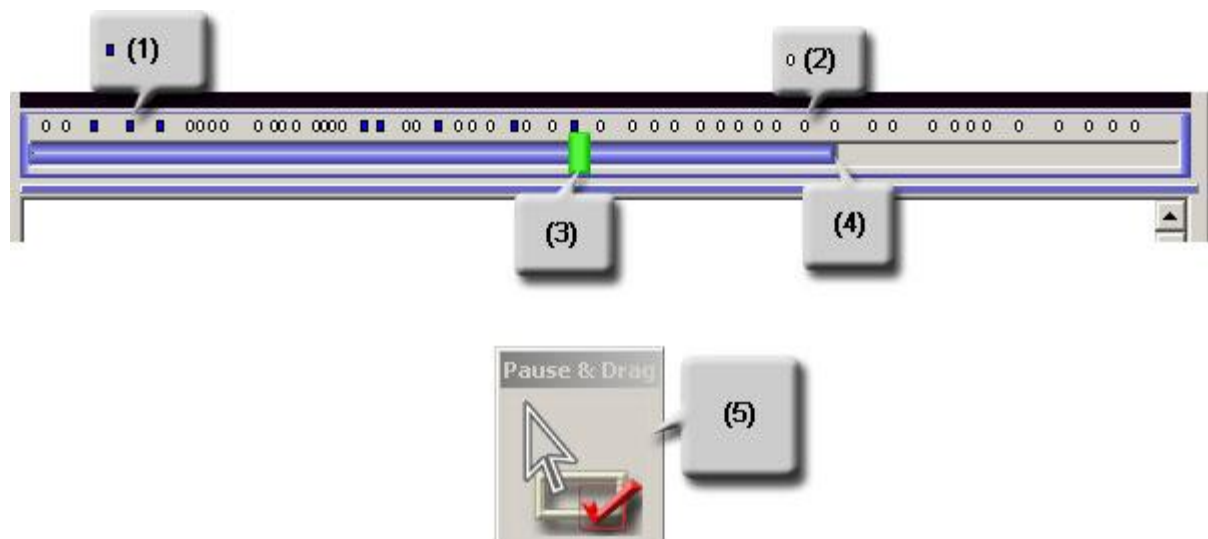




Figure 5-1 Pedagogical enhancements on client side, VoD Player video slider and “Pause and Drag” windows

The descriptions of the fields in Figure 5-1 are:

1. Indicates a user contribution. The user contributions are stored in the client computer the as the same way they are stored in the Content Server. When a project launches, the VoD Player searches for that specific projects user contributions in a special Microsoft Windows folder called Application Data. If the player finds the appropriate project folder, then it searches for the synchronization

data inside that folder and locates the user contributions on the slider bar in their time locations as referred in the synchronization data. The user contributions are marked with “” a picture identifier which has a blue color face and rectangular shape.

2. Indicates a non changeable content material added by the author of the presentation on project building time. These additions are made through “VoD Project Editor” and marked with “” a picture identifier which has a white color and elliptical shape.
3. Indicates the thumb of the slider bar, and shows the current position of the presentation video. Square and ellipse shaped tics over the thumb are points where a new content material will be shown in the html pane of the VoD Player.
4. The progress of the buffering operation is integrated to the slider bar in order to obtain more space for the presentation video and the contents.
5. “Pause and Drag” is a floating on top of all windows, independent window come up with the initialization of the video screen. This window provides “Drag and Drop” facilities to the user to add user contributions easily to the currently opened project. The dragging and dropping cycle works as follows; when the user drags an html page, text, image or any document that could be opened by the internet browser to the Drag and Drop window, VoD Player gets the video frame number corresponding to the drag operation, copies the dragged files to the local project folder and creates a synchronization data for user contributions to the video frames.

One of the fundamental objectives of VoD thesis is to simplify the production tasks of DE projects; for this purpose, a design editor application that would be capable of adding and editing html pages to the projects is implemented. This design editor is also capable of synchronizing html pages to video frames where proper timing of the html pages is compulsory.

VoD Player searches for synchronization data in the Content Server, where the project files are located, and uses that data to show the html pages in their proper timings. The Project Editor produces synchronization data when the projects are uploaded to the servers.

VoD Project Editor is a sophisticated application interface for designing VoD projects for DE. The DL environment in its nature; must be implemented in a way that all the users/students be able to understand the DE material, in this case the DE video cannot be too complicated or too simple. In the sake of not to broke up the video or not to capture a video for more explanation about the topic, we preferred to give additional content material, where this material could include more advanced or more introductory issues in html pages under the video instead of a video with full of non-point of interest materials.

An example of that concept could be given as; if the instructor in the video presentation is covering a topic of Signal Processing, which includes Laplace Transformations calculations hardly, and if the main point of interest is not the transformations, more information, transformation tables and the links about the transformations could be given as content material in the html pages included to the project. Also the instructor could add more advanced topics such as Digital Signal Processing topics, wavelets and image compression to the content related supporting material.

5.1 The Project Editor

The project editor is a tool for building VoD projects. The Project Editor includes a video preview screen, an html viewer, an html editor and an html source code editor. To prepare the VoD presentations, the authors of the presentations should organize the video and supporting content materials before putting them to the VoD Project Editor. The presentation video must be edited or prepared by an external tool before adding to a project.

The functionalities of the editor are:

- Video previewing,
- Html viewing,
- Html creating and editing,
- Html source code editing,
- Creating the synchronization data for html pages,
- Sending the designed projects to the Content Servers via FTP,
- Receiving previously made projects for editing and
- Registering the projects in the Mediation Server.

To build new projects, the users must gain authorization from the Mediation Server. After authorization, the user is presented with an interface as shown in Figure 5-2. The VoD Project Editors main window is composed of four sub (child) windows.

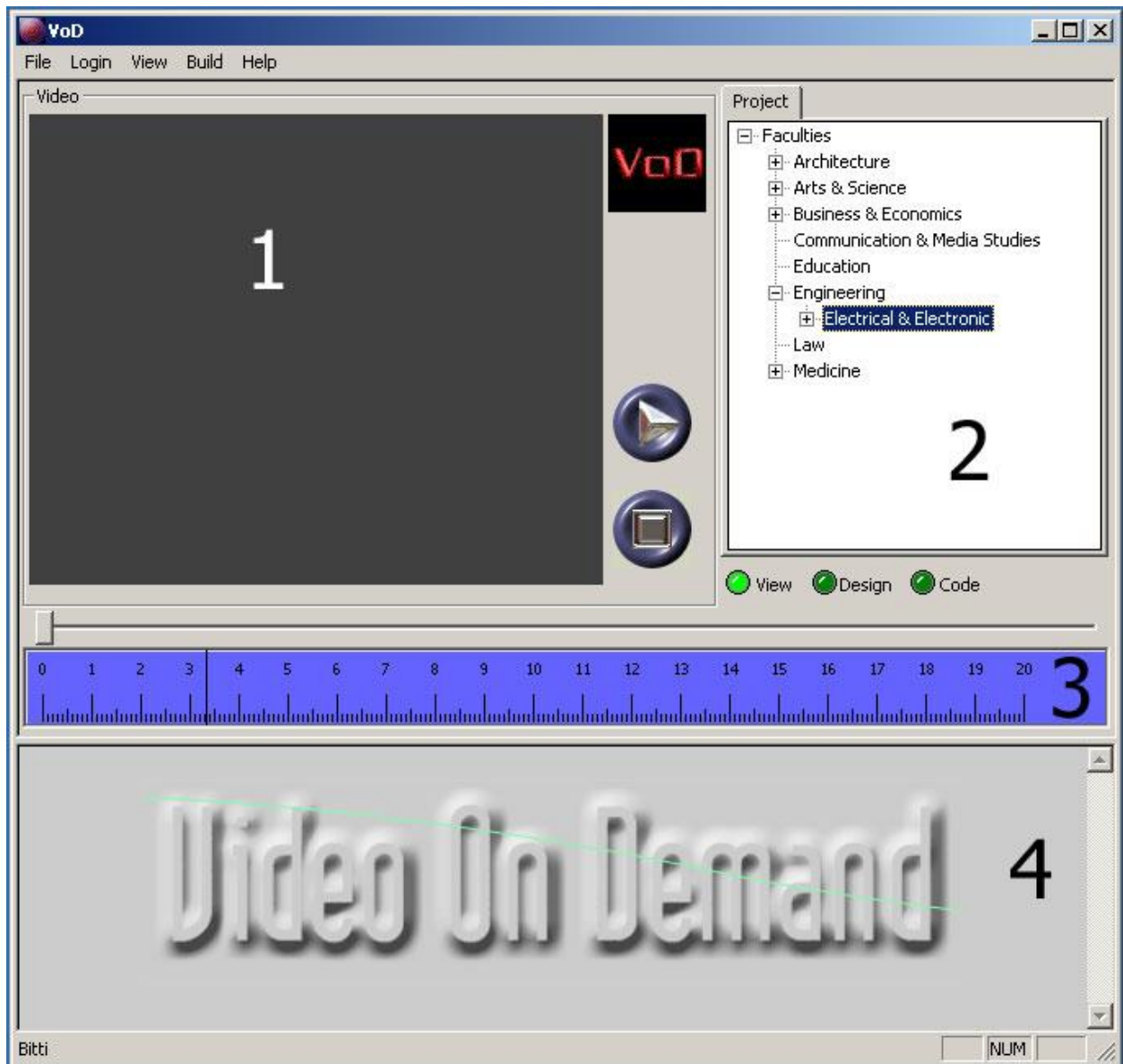


Figure 5-2 VoD Project Editor

All the child windows seen in the figure above have different functionalities. The child windows corresponding to the numbers shown in Figure 5-2 are:

1. Main video preview screen,
2. Available categories to add or edit projects,
3. Content material addition/edition bar,
4. Html viewer/editor/code editor screens.

5.1.1 Project Editing Tree

Project Editing Tree allows the users to add or edit their projects and sorts the project in hierarchical order starting from faculties and end with the projects under course names. The end user is allowed to add new fields to each category such as a new chapter, course, department and even a faculty based on the user access rights defined. A sample screen shot is represented in Figure 5-3:

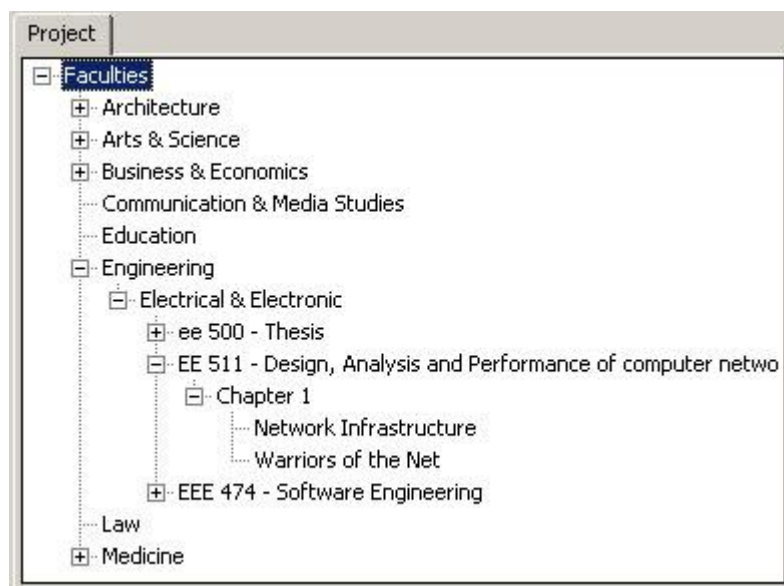


Figure 5-3 Schematic showing the VoD Project Tree

5.1.2 Synchronization of HTML Pages to Video

The synchronization is made through a scrolling ruler^Ψ which has a sensitivity of 10 divisions per second as seen on Figure 5-4. The ruler slides from the right to left direction while the video is in the playback operation. The arrow-heads seen on Figure 5-4 are the points where html pages are added.

^Ψ Based on the Uğur Hamzadayı's CRuler MFC Class code example.

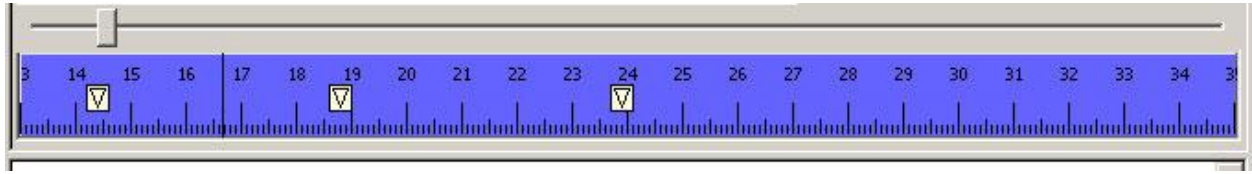


Figure 5-4 Html to video synchronization

The indicators seen in the figure are the points where an html page is already added. When a new project is created, a new html page is created by the system and marked with an indicator in the first second of the scrolling ruler. This first indicator neither cannot be moved to another place nor deleted. The other html pages can be added by simply right clicking the ruler in any position and an empty html page opened while the video automatically switches to pause state. After adding a new html page, a new indicator will show the html page's time position in the scrolling ruler bar. Those new additions can be moved in the time line to any position and the html pages positions can also be changed.

5.1.3 WYSIWYG DHTML Editor

The word WYSIWYG is the synonym for What You See Is What You Get. This name is selected for that editor, because this editor is a high level html page designer and the end editor does not require any html source code knowledge. For any reason, if the end user wants to manipulate the source code, a code browser is also available. For that reason the DHTML Editor built in the VoD Project Editor has three phases that a project designer could use, one for viewing the html pages, one for editing and the last one edits the source code of the html page that is currently available. Those three phases are shown in Fig. 5.5(a, b, c):

```

void CThe::OnAdd()
{
    for(unsigned i=0;i<PageCount;i++)
    {
        if(Pages[i].Frame==(int)m_Ruler.GetScrollPos()/3)
        {
            SetPage(Pages[i].PageName);
            Pages[i].PageSeenBefore=TRUE;
        }
    }
}

```

Figure 5.5 a DHTML viewer

```

void CThe::OnAdd()
{
    for(unsigned i=0;i<PageCount;i++)
    {
        if(Pages[i].Frame==(int)m_Ruler.GetScrollPos()/3)
        {
            SetPage(Pages[i].PageName);
            Pages[i].PageSeenBefore=TRUE;
        }
    }
}

```

Figure 5.5 b DHTML editor

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=windows-1254">
<META content="MSHTML 6.00.2800.1276" name=GENERATOR></HEAD>
<BODY leftMargin=0 topMargin=0>
<TABLE align=center border=1>
  <TBODY>
    <TR>
      <TD><PRE><FONT color=blue>void</FONT> CThe::<STRONG>OnAdd</STRONG>()
{
  <STRONG>for</STRONG>(unsigned i=0;i<PageCount;i++)
  {
    if(Pages[i].Frame==(int)m_Ruler.GetScrollPos()/3)

```

Figure 5-5 a- DHTML viewer, b- DHTML editor, c- Source code editor.

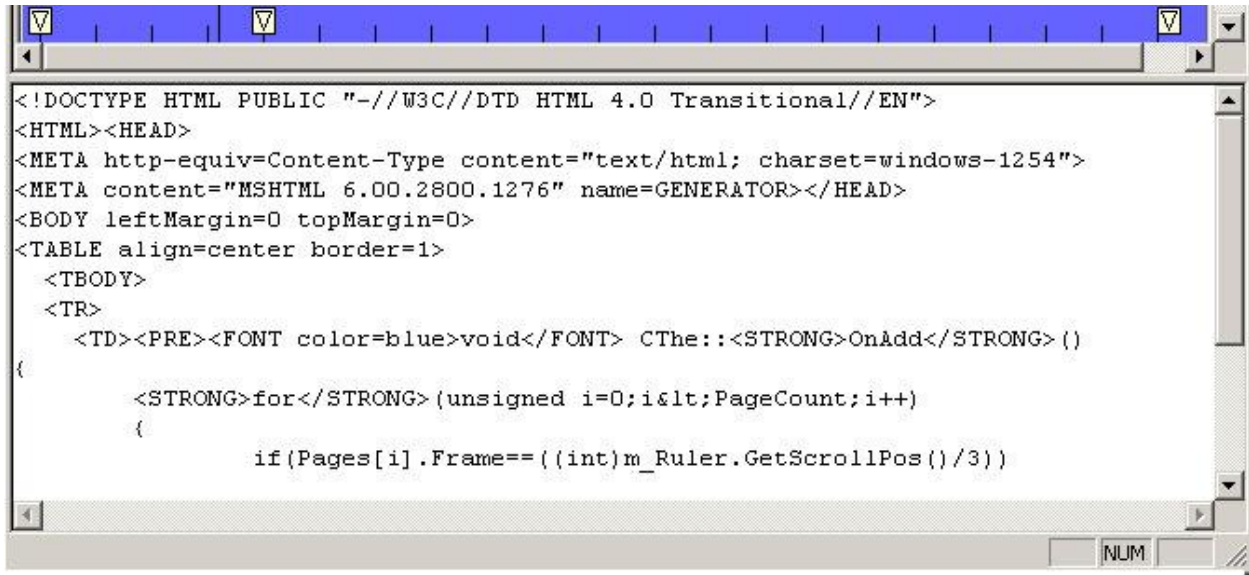


Figure 5-5.a is implemented for previewing of the DHTML pages designed in the DHTML editor. This viewer has all the capabilities of the built-in Microsoft Internet Explorer, because VoD Project Editor uses Internet Explorer server as an ActiveX component.

The DHTML Editor is a simple yet a useful DHTML editing tool for editing web pages in WYSIWYG style editing. This editor cannot handle complex DHTML tasks but should be able to compose understandable web pages. The capabilities of DHTML editor is given in the Table 5.1 as follows:

Command	DHTML Map	Explanation
Copy	IDM_COPY	Copies a selected region
Cut	IDM_CUT	Cuts the selected region
Paste	IDM_PASTE	Pastes the clipboard content to a region
Undo	IDM_UNDO	Undo an action
Underline	IDM_UNDERLINE	Makes the selected text underlined
Italic	IDM_ITALIC	Makes the selected text italic
Bold	IDM_BOLD	Makes the selected text bold
Justify Left	IDM_JUSTIFYLEFT	Justifies the paragraph to left
Justify Center	IDM_JUSTIFYCENTER	Justifies the paragraph to center

Justify Right	IDM_JUSTIFYRIGHT	Justifies the paragraph to right
Hyperlink	IDM_HYPERLINK	Makes an hyperlink to a selected text
Image	IDM_IMAGE	Add an image to the html page
Font	IDM_FONT	Sets font by opening font selection box
Indent	IDM_INDENT	Sets the indentation on a paragraph
Out dent	IDM_OUTDENT	Sets the indentation on a paragraph

Table 5.1 Available DHTML commands in the VoD Project Editor

5.1.4 Synchronization Data Structure

The VoD Project Editor creates synchronization data just before the project is sent to the Content Server. This synchronization data is used when the projects are edited by the Project Editor or used by the VoD Player when presenting the projects. The synchronization data is placed in to the project folder in the Content Server.

The html files edited by the VoD Project Editor are given unique names starting from zero and each html page's position on the scrolling ruler is recorded to a structure given in the Appendix A. For the test purposes, the number of html pages is restricted to 200 and to changing this value requires recompilation of the applications. The simplified structure of the html page is given in the Table 5.2 below,

Page Structure		
Type	Name	Definition
Integer	ID	Identifier of the page
String	PageName	The name of the html page
Integer	Frame	The video frame that the page will be shown
Boolean	PageSeenBefore	Set to disable continuous openings in the same frame

Table 5.2 The html Page Structure

Synchronization Data		
Type	Name	Definition
Integer	Slider Range	The range of the video slider
Integer	Page Count	The number of pages available for the presentation
Pages St.	Pages	A collection of Page Structure
String	Path	Path to project in Content Server
Integer	TotalTime	Total duration of the video file

Table 5.3 Synchronization Data

Table 5.3 depicts the structure of the synchronization data generated for each html page and a sample synchronization data is given below in Table 5.4.

Sample Synchronization Data		
Field	Content	
Slider Range	0 to 2300	
Page Count	2	
Pages	ID	1
	Page Name	0.html
	Frame	513 th Frame
	Page Seen Before	NO
	ID	2
	Page Name	1.html
	Frame	856 th Frame
	Page Seen Before	NO
Path	F:\VoD\Project	
Total Time	1113 seconds	

Table 5.4 A sample synchronization data

5.2 Pedagogically Improved VoD Player

The system includes an improved version of VoD Player in order to supply “chatting”

facilities by the online[♦] users who prefer more interactivity while watching the VoD presentations. This player has three browser windows instead of one window which includes a window for real-time chatting, one for the list of online users and the content browser window as shown in Figure 5-6 on the next page.

Chat enabled VoD Player is made as an additional application and is able to handle all the tasks that standard VoD Player can implement. The reason not to make it the standard VoD Player is the high resources that it requires; the chat windows updates itself on every five seconds and three built in html browsers require more hardware and system resources.

This system requires at least two users to be online at the same project and every user could join to the chat from other VoD Players or directly from the VoD web site. The chat messages are stored in the Mediation server temporarily for twenty minutes.

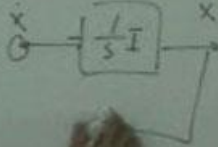
[♦] Chatting facilities also available in the web site of VoD system.

VoD Player - Project: Stability

4 Stability of State Variable Systems

Syst. $\dot{x} = Ax + Bu$
 $y = Cx + Du$

Block Diagram



Project Name: Stability
 Subject: Linear Systems
[Open Stability forum](#)
 Logged as **Student**
 Active Users:
 Std: Cem KARACA
 Inst: Ahmet Turhaner
 Std: Emre Öktem
 Std: Gökhan NAS
 Adm: INTERDEC
 VoD Admin

00 00 00 00 00 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Send

aturhaner: Now the system in the figure below is in its stable region and when flame burns the system will go in an unstable condition.
 cem: Ok
 aturhaner: Now, move your slider to 30th second

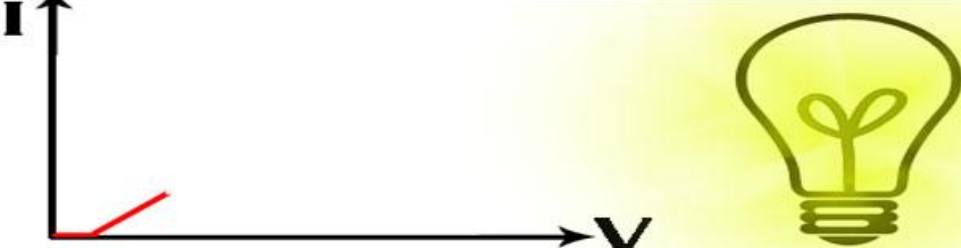


Figure 5-6 Enhanced VoD Player

6 MULTI-PROTOCOL ENHANCEMENTS

The client side VoD Player application designed in this thesis is enriched with a range of available IP protocols for the ease of use for multiplicity of choice and for analyzing the differences and the advantages of these protocols when used for streaming video. The proposed transport and application protocols used in the system are given in the sections below.

6.1 VAP and VoD System in OSI Reference Model

VoD Application Protocol (VAP) acts as a transition layer between the VoD client applications and the transport and datagram protocols. It simply converts the high-level client commands to a set of functions to communicate with the VoD Mediation and the Content servers through appropriate transport protocol.

This section describes the VoD Application Protocol (VAP) its commands, options and the operation. The main duty of the VAP is communication between the client-server (Mediation and Content Servers), retrieve of the necessary database and related information. The VAP is integrated to the client applications core. The VAP process steps on VoD Player are:

1. VoD Player sends a command to VAP in a higher-level language as; "Receive

project from the Content Server where project id is equal to ID and callback when ready". In the query the field "ID" is a variable identifying the projects unique id.

2. VAP queries project specific variables from the Mediation server such as the Content Server IP address, projects folder location as stored in the Content Server, etc.
3. VAP makes an FTP connection to the Content Server to download content related materials and the synchronization data.
4. After downloading the content materials, VAP searches the Windows registry in order to select the preferred transport or datagram protocol for the video transmission.
5. VAP makes a connection to the Content Server to download (stream) project video with the preferred transport protocol.
6. Depending on the playback and download stream rates, VAP decides to make direct or smooth buffering.
7. When the playback time comes (calculations are made in Ch.4), VAP sends a callback to the VoD Player.

In order to process these steps, the VAP makes several network connections and acts as a bridge between the networking interfaces and the VoD Player application. For a graphical representation of the relations between the VoD Player, VAP and the network transport protocols, it is better to attribute to the OSI Reference Model.

OSI is the abbreviation for Open System Interconnection* and designed by International Organization for Standardization (ISO) and International Telecommunications Union (ITU). OSI is designed to enable communication through

* More information about OSI could be found in ISO 7498 standard.

any computer systems regardless of the vendor or manufacturer of the system. The resulting OSI model aimed system programmers to design the systems in more structured way and the result is; any application working on a system, even in a refrigerator could be able to communicate to another computer system such as a washing machine or a central mainframe.

It is better to show the system network connection infrastructure over the ISO-RM model, because the properties of the system could be understood correctly. The connection between VoD clients and the servers are modeled in Fig. 6.1 below.

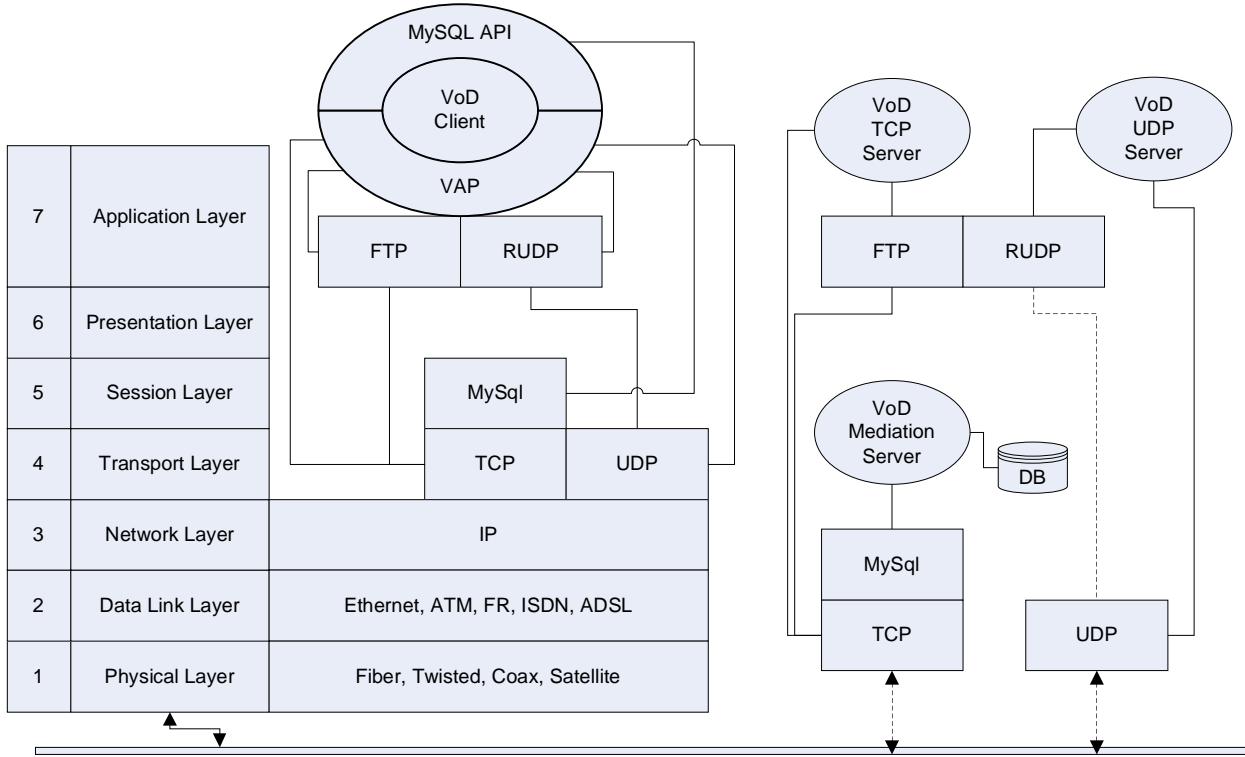


Figure 6-1 Inputs and the outputs to layers of OSI-RM by VoD system

VoD clients make many connections to the servers from different OSI layers by the help of VAP as seen on Figure 6-1. The reason for that is; VoD system is designed to be able to communicate to the servers from different network protocols, besides on of the fundamental reason to make the thesis lies in this idea and obtain the differences

between these protocols in such video application.

VAP uses OSI-RM Layer-7 to communicate through FTP and RUDP protocols and Layer-4 to communicate through TCP and UDP protocols. The abstraction of the database communication stuff left to MySQL in the Session Layer. The advantages of the used protocols are listed in the table 6.1 below:

Protocol	Connection	Buffer	Cost
FTP	Connection oriented	Depends on the FTP server	FTP+TCP headers
RUDP	Connection oriented	65467 Bytes	RUDP header
TCP	Connection oriented	Max available	TCP header
UDP	Connectionless	65467 Bytes	Non-reliable

Table 6.1 Basic comparison chart for the protocols

6.2 Transport Control Protocol (TCP)

Transport Control Protocol, as its name identifies, is used for controlled transmission of data over the network, in other words, TCP aims to provide error free transmission in packet switched computer networks. By its nature TCP is a connection oriented protocol, this means that the client and the server must dedicate a port for any communication through TCP/IP till the end of the session of communication. TCP/IP was designed under the Defense Advanced Research Projects Agency (DARPA) project in September 1981 and standardized by IETF in RFC 793. TCP/IP also forms the infrastructure for many application protocols such as File Transfer Protocol, FTP.

TCP communications for VoD operations are made through a free port that is

not assigned or owned by any other TCP processes such as HTTP or FTP processes. The port number 21000 is selected for TCP communications between the Content Servers and the clients.

6.2.1 VAP commands for handling TCP communications

In order to have the VoD Client and the server can communicate and exchange information VoD Application Protocol (VAP) is needed. Like a TELNET or FTP process, VoD system has its own communication commands for handling negotiation and file related tasks. The VAP commands to handle TCP tasks are;

Command	Definition	Response from VoD Server
CONNECT	Connection request to the server	"0 Server Ready" if server is alive.
QUIT	Sends a quit message to TCP thread of the server	No response
AUTH *	Internal password "*" to identify VoD applications to VoD servers	If true: "User accepted" else "Not logged in!"
SCD *	Sets the remote directory to *	True or False
READ *	Reads the file from remote location	Sends the file "*" if exists else sends back an error back
SETREAD *	Sets the file "*" to be retrieved partially	True or False
SET *	Retrieves the part of a file set by SETREAD starting from "*" bytes	If successful sends the partial file else sends an error back
GFS *	Retrieves the file size of "*"	If successful sends the file size else sends an error back

Table 6.2 VAP internal commands for TCP

All the commands represented in the Table 6.2, have a parameter part except the commands CONNECT and QUIT. The first part of the commands identify the action to take and the rest indicated with “*” are required words to handle the tasks.

The maximum number of clients that could connect to TCP server at the same time is restricted 20 users and this number could object to change depending on the hardware capabilities of the server.

6.3 File Transfer Protocol (FTP)

The File Transfer Protocol is the most heavily used communication protocol in VoD systems because not only the video itself also the content materials are transferred through FTP. There is no other option to transfer content materials from VoD Project editor to Content Server and also from Content Server to VoD Player.

FTP is a designed to transfer bulk data reliably without any loss. FTP inherits reliability from TCP/IP and all the infrastructure for communication over FTP is handled by TCP/IP. Infact FTP is nothing but a group of commands to handle communications between the client and the server machine and the advantage of using FTP is; it is standardized way to exchange files between computers, every FTP server and clients in the globe talks the same language without any dependence on any hardware or software platform.

Additions and enhancements to FTP are updated by the Internet Engineering Task Force (IETF). RFC's related to the FTP are RFC 264[1972], RFC 265[1971], RFC

354[1972], RFC 765[1980] and RFC 959[1985].

6.4 User Datagram Protocol (UDP)

User Datagram Protocol is a non-reliable, connectionless protocol mostly used for one way transmission of the data. Since the protocol is connectionless, the receiver system does not send back any acknowledge and the listening UDP port does not specifically wait for one computer’s messages, hence every computer could easily send message to the listener without any dedication. More on UDP could be found from the internet RFC’s.

The UDP header takes a very small amount of size considered to its maximum message size, hence; UDP could transfer its data through network with a small overhead. The maximum message size for UDP is 65467 Bytes for the IP version IPv4 and the header format for UDP is;

0	1	3
6	6	1
Source Port	Destination Port	
UDP Length	Checksum	
Data		

Table 6.3 UDP Header format

6.4.1 Slow Start Algorithm for UDP

Since the VoD system is built on ATM infrastructure and UDP is a connectionless

protocol, UDP packets exhausted from ATM has the speed of ATM connection and the client side could not catch the whole packages.

Before starting UDP transfers, VoD Player sends the client network connection information to the VoD UDP server. This speed value is not tested by the client application; instead the user must configure the connection speed once when the client VoD application is installed. The configuration screen is given in Fig. 6.2 below;

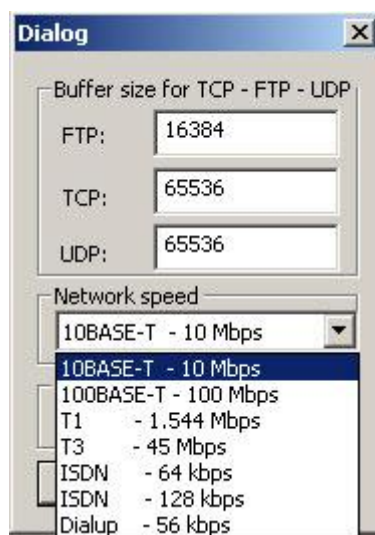


Figure 6-2 Speed selector dialog box

The VoD UDP server sends the selected video file starting with the speed of 10% of the desired speed in order to prevent disconnection and increases the speed to the actual value in seconds, because the client side VoD Player makes initializations of video screen and DirectX at that time and these initializations takes the application resources completely. In connection oriented protocols, VoD Player does not send any acknowledge until the system initializations ends but in UDP, the packets sent through the server without any transmission control. The solution is to receive the first group of packages in slower speeds. The slow start algorithm is given in Figure 6-3;

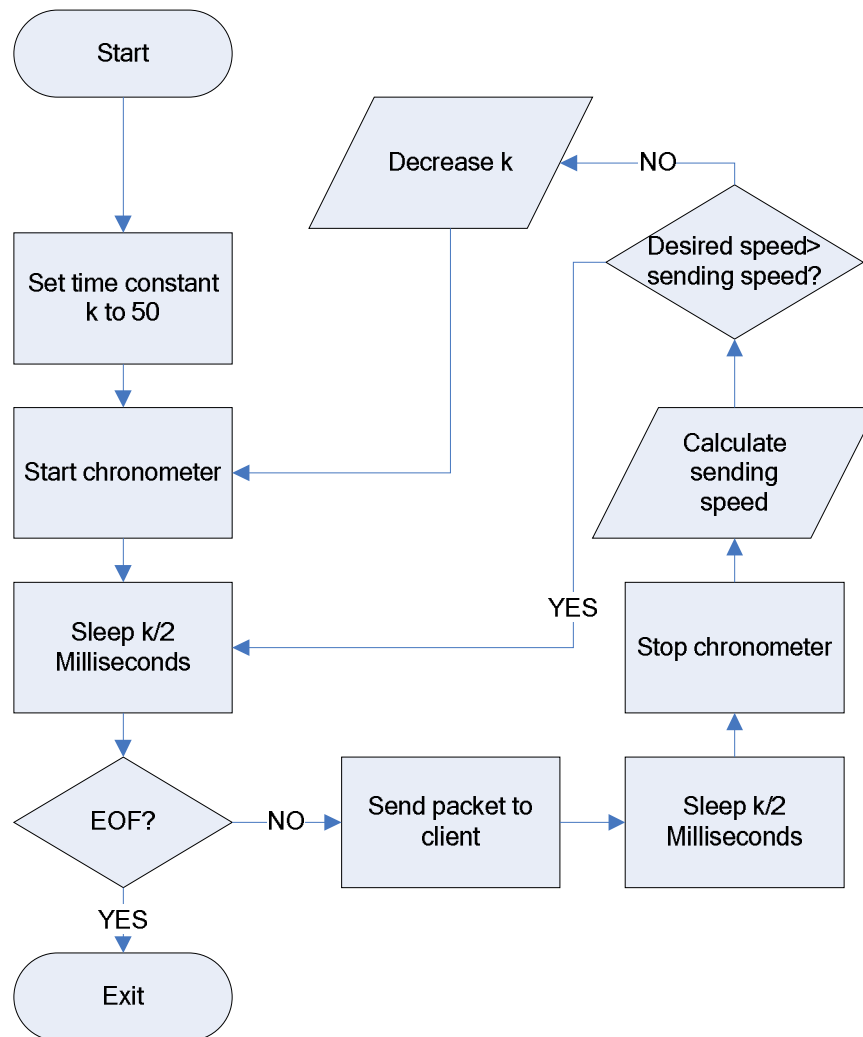


Figure 6-3 Slow start algorithm flowchart

6.5 Reliable UDP

Reliable UDP (RUDP) is a connection oriented a reliable transmission protocol for transferring video data through Campus-wide Area Network (CAN). RUDP is working over the protocol UDP. RUDP has its own data structure and communication control commands. Those commands and data structures are packed in the data segment of UDP and the receiving application unpacks the RUDP data after the kernel strips the UDP header.

RUDP could be classified in a region between UDP and TCP in the senses of speed and reliability. In ideal cases, where the server and the client have the same communication speed with no network traffic, UDP should be considered as the fastest communication protocol because of its minimum overhead. Also UDP transmissions does not require acknowledges to be sent back and the datagram could be sent continuously without waiting for acknowledge. But in real life situations, the datagram could not be caught by the clients in the case where the server's communication speed is greater than the clients receiving speed. The comparison of RUDP to other protocols is given in the Figure 6-4;

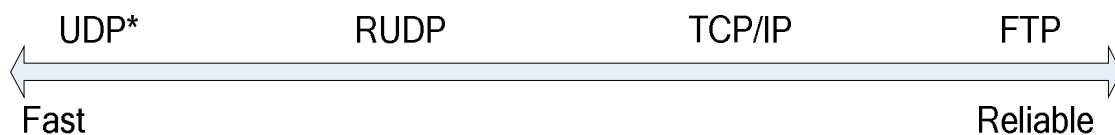


Figure 6-4 RUDP Speed/Reliability comparisons

* in ideal cases, i.e. no network traffic also the sender and the receiver must be in the same connection speed.

6.5.1 RDUP Header and Data Format

RUDP has two types of data formats;

1. RUDP Acknowledge data format,
2. RUDP Bulk data format

The acknowledge data format is designed to be as light as possible in order to decrease the network utilization and decrease the analysis time of the acknowledge data. The acknowledge data format is given in the Table 6.4 below;

0		8		1		6		1	3
Source Port				Destination Port					
UDP Length				Checksum					
Command		Success code							

Table 6.4 RUDP Acknowledge data format

The data format seen in the Table 6.4 has two additional fields added to standard UDP header. The Command field is an 8 bit code including the action to be handled by the RUDP server and the Success code is also an 8 bit code identifying the accomplishment of the previous packages success.

0		8		1		6		1	3
Source Port				Destination Port					
UDP Length				Checksum					
Size									
32 bit CRC									
Packet Number									
Bulk data									

Table 6.5 RUDP Bulk data format

Every packet sent through RUDP server to the clients are 65467 bytes long, which is the maximum available size for an UDP packet in RSVP. RUDP uses the best effort scheme and uses the highest possible bandwidth in the network. After the journey of the UDP packet in the network, the UDP header is striped by the kernel and the resulting RUDP packet header is also striped by the VoD Player. The RUDP header consists of the size of the RUDP header plus the bulk data, a 32 bit CRC data and the received packet's identification number.

The bulk data's CRC number is re-calculated by the VoD Player and compared

with the one came in the RUDP header. If the packet's identification number is the expected packet's and the CRC codes match, the bulk data is written to the buffer and VoD player sends back an acknowledge message "ROK" stating that the packet is successfully received, if the circumstances does not fit, the VoD Player send an acknowledge including the error code "RFAIL".

6.5.2 VAP commands for handling RDUP and UDP Communications

Command	Definition	Response from VoD Server
CONNECT	Connection request to the server	"0 Server Ready" if server is alive.
AUTH *	Internal password "*" to identify VoD applications to VoD servers	If true: "User accepted" else "Not logged in!"
QUIT	Sends a quit message to TCP thread of the server	No response
SCD *	Sets the remote directory to *	True or False
READ *	Reads the file from remote location	Sends the file "*" if exists else sends back an error back
SETREAD *	Sets the file "*" to be retrieved partially	True or False
SET *	Retrieves the part of a file set by SETREAD starting from "*" bytes	If successful sends the partial file else sends an error back
GFS *	Retrieves the file size of "*"	If successful sends the file size else sends an error back
RREAD * (RUDP only)	Applies the command "*" .	Applies the command and sends the desired packet to the client.
CS * (UDP only)	Sets the UDP transmission speed to *	Allows UDP to transmit packets to the clients in their bandwidth limits.

Table 6.6 VoD internal commands for TCP

6.6 ATM LAN Emulation

In Asynchronous Transfer Mode (ATM) Local Area Network Emulation (LANE), ATM device simulates the Ethernet device and works with the same principles as an Ethernet device and supplies ATM speed to Local Area Network and provides entry to ATM network with directly from a computer. More information on LAN Emulation could be gathered from the ATM Forums [1995] paper on LAN Emulation over ATM.

7 SYSTEM PERFORMANCE MEASUREMENTS

VoD system performance measurements are based on the network communication performances of the used protocols, buffering system and VoD system working performances. Yet, for the time being it is impossible to analyze the effects of VoD system on the students.

7.1 Network Protocols Performance Tests

All the performance tests are made in the Eastern Mediterranean University campus wide network. The following tests cover one-to-one and one-to-many connections. The network line especially selected to pass through campus backbone in order to go far away from ideal cases.

The test network configuration includes broadcast, unicast and multicast traffic propagated from ARP, IPX (802.2), IPX (802.3), IPX (Eth II), TCP, UDP, IGMP and ICMP protocols.

The test configuration model is given in the Figure 7-1. The same figure also applies to one-to-many connection tests except, more VoD clients are added to the switches regardless of the location of the switches.

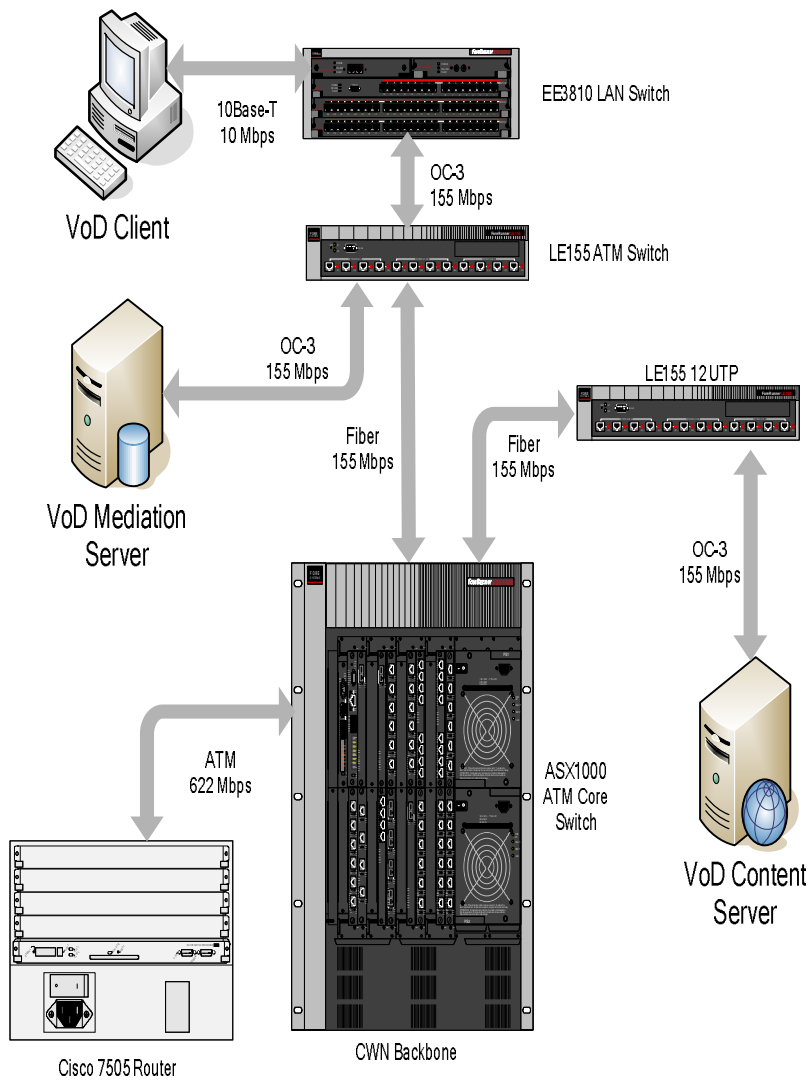


Figure 7-1 Campus Wide Network as a test configuration

The client configuration used for the measurements in Figure 7-1 is:

- Intel Pentium 1.7
- 512 MB RDRAM
- 120 GB HDD
- Cybex 10 Base-T 10 Mbps LAN connection
- 32 MB N-Vidia Ge-Force2 GTS Video graphics card
- Microsoft Window 2000 Pro with SP5 Locale Turkish

The VoD Content Server configuration Figure 7-1 is:

- Intel Pentium 1.7
- 256 MB DDR-RAM

- 20 GB HDD
- ForeRunner PCA-200EPC ATM card 135 Mbps ATM connection
- 32 MB N-Vidia Ge-Force2 MX Video graphics card
- Microsoft Window 2000 Pro with SP5 Locale Turkish

The VoD Mediation Server configuration Fig. 7.1 is:

- Intel Pentium 1.5
- 128 MB DDR-RAM
- 30 GB HDD
- ForeRunner PCA-200EPC ATM card 135 Mbps ATM connection
- 32 MB N-Vidia TNT2 Video graphics card
- Linux 2.4.20-2.41smp #1 Red Hat Linux 9.0
- Apache 2.0.45 with PHP 4.0 web server
- MySql 3.23.54

The tests are made in the crowded network time of the day about 10:00 and 12:00 o'clock and the results are the average of 10 experiments per connection type. The VoD Player in the client side has the buffer sizes set to 65 Kbytes per connection type and the maximum speed is set to 10 Mbps. The downloaded data in the experiments is;

- 94,605,392 bytes long
- Total playing time: 09:02 seconds
- Video size: 352 x 288 pixels
- Bit rate: 1150000 bits/second
- Frame rate: 25.00 frames/second

7.1.1 VoD Network Performance Results

One client tests are made ten times per connection type and the results of the video download times in milliseconds are given in the Table 7.1 Download times in milliseconds.

Test #	FTP	UDP	RUDP	TCP
1	101046	87586	82239	98852
2	103759	87566	81977	98682
3	102197	87576	83310	98622
4	102458	87586	83811	99062
5	104120	87576	82950	98652
6	102177	87786	82969	99904
7	103139	87766	83380	99312
8	104310	87776	82970	98571
9	104961	87787	82428	96068
10	103869	87746	81998	96268
Average	103203.6	87675.1	82803.2	98399.3

Table 7.1 Download times in milliseconds

The speeds of the protocols could be calculated as follows;

Download Speed (DS) = Video Size (Mbits) / Average Download Speed (s) yields to:

$$\begin{aligned} \text{FTP:} \quad \text{DS} &= 94,605,392 * 8 / 103.2036 \\ &= 7.3335 \text{ Mbps} \end{aligned}$$

$$\begin{aligned} \text{UDP:} \quad \text{DS} &= 94,605,392 * 8 / 87.6751 \\ &= 8.6324 \text{ Mbps} \end{aligned}$$

$$\begin{aligned} \text{RUDP:} \quad \text{DS} &= 94,605,392 * 8 / 82.8032 \\ &= 9.1403 \text{ Mbps} \end{aligned}$$

$$\begin{aligned} \text{TCP:} \quad \text{DS} &= 94,605,392 * 8 / 98.3993 \\ &= 7.6916 \text{ Mbps} \end{aligned}$$

RUDP comes out the fastest transmission protocol for video transmissions. In fact in ideal cases, where the server speed and the client speed are the same, surely UDP will be the fastest connection because of its minimum overhead. But in connections where the server speed is much greater than the client speed, as in this configuration, the UDP throughput speed is limited by the server in order to give the clients the chance to catch all the packets. The results are given in the Figure 7-2;

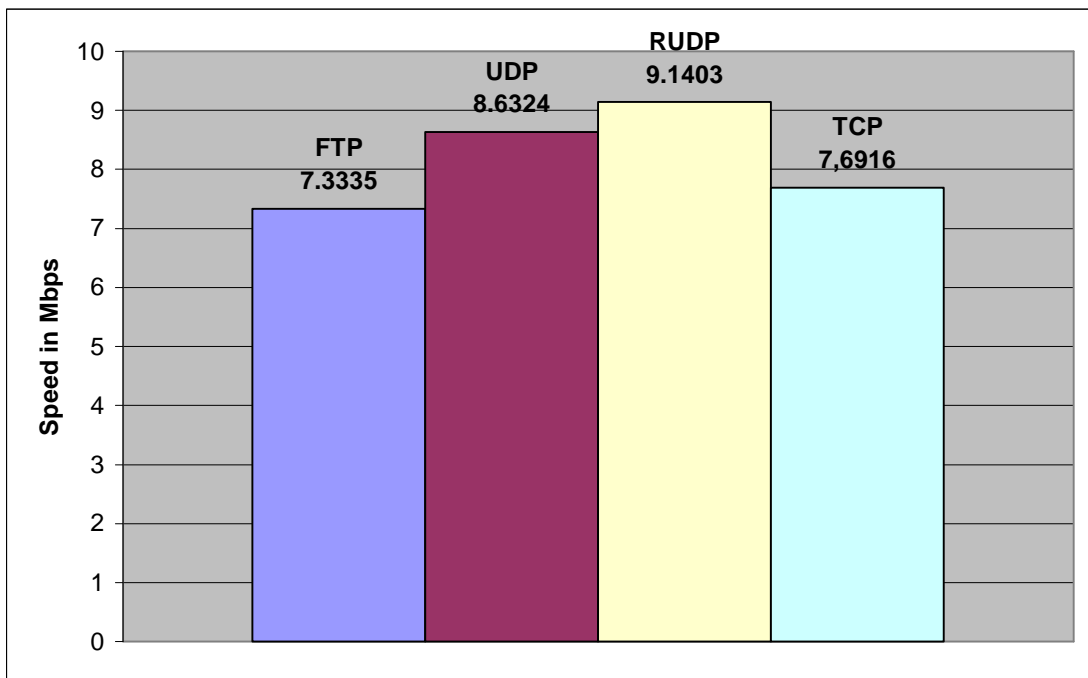


Figure 7-2 Download speeds of the protocols

7.2 Download Speed and Playing Speed Comparisons

The formula for the smooth buffering was calculated in the section 4.5.2. The analysis of both situations where playing speed could be greater than the download speed or vice versa are simulated and the results are given in the figures 7.2 and 7.3 below;

$$t_w = \begin{cases} 0 & : S_p < S_D \\ \frac{t_p * S_p}{S_D} - t_p & : S_p > S_D \end{cases} \Leftrightarrow 0 < S_D$$

Note: The symbol definitions could be found in the Table 4.3

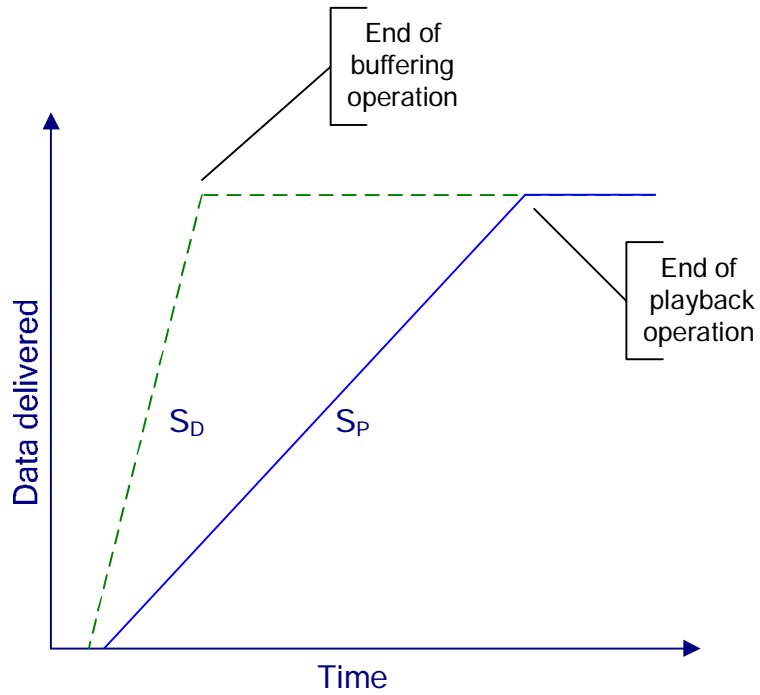


Figure 7-3 Buffering vs. Playback $S_p < S_D$

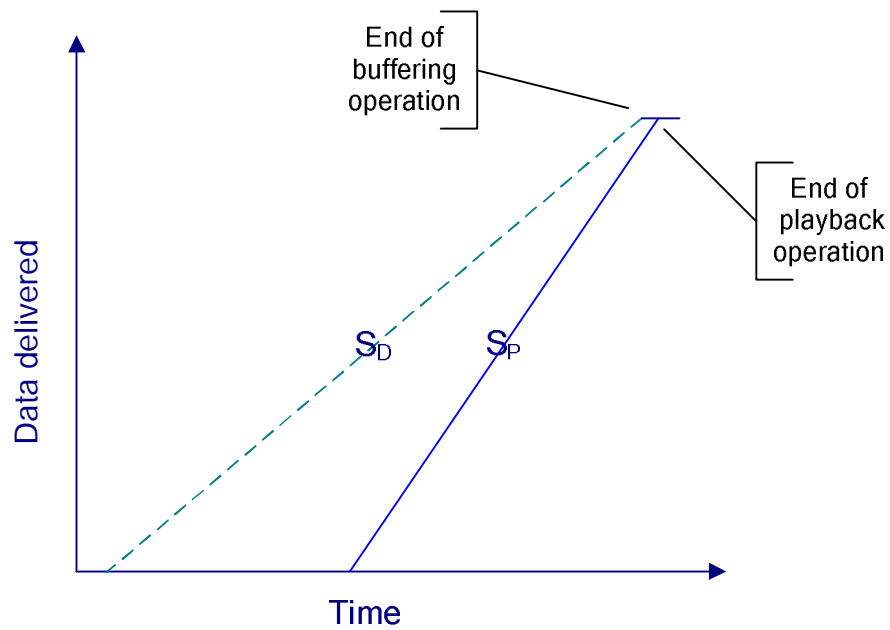


Figure 7-4 Buffering vs. Playback $S_p > S_D$

8 CONCLUSION AND FURTHER WORK

8.1 Conclusion

Computerized Distance Education (DE) is the one of the fields that incorporates many technologies together, including audio-visual, multimedia compressions and networking technologies and the absence of any of the technologies mentioned above could decrease the interactivity and the quality of the DE system dramatically.

The main objective of this thesis is to offer a complete Asynchronous Distance Education (ADE) system which could be accessed on demand by the learners where and when necessary while reducing the disadvantages of ADE systems such as the absence of any interactivity of the online learners with the online instructors. To achieve that goal, DE system is based on Video on Demand (VoD) infrastructure supported with “html” based content related materials which may include animations, text, images besides ActiveX objects. The improved VoD system designed for this thesis has the following properties:

1. VoD Servers:

- VoD Server system uses distributed servers architecture in order to suppress and distribute network load to several LANs while giving the server administrators much flexibility on content arrangements.
- VoD Servers are separated into two parts by their functionalities called as

Content Servers and Mediation Server in order to ease the manipulation of the system. Content Servers store all the DE materials including videos and the supporting materials and there is no limitation on the number of Content Servers. Mediation Server is responsible from storing the user profiles, Content Server addresses, statistics and other system related data.

2. VoD Server Applications:

- Mediation Server supported with MySQL RDBM system in order to support both VoD Client applications and the web based interface of VoD System. Client applications authenticate users, gather DE project information and locate the Content Servers by the help of Mediation Server. Web part of the VoD system also gathers the same information besides stores additional information such as Discussion Board data in the Mediation Server.
- Content Servers are bundled with VoD TCP, UDP, RUDP Server application and a commercial third party FTP Server in order to fulfill the multi-protocol content retrieval requirements. VoD TCP, UDP and RUDP Servers are responsible from sending course video to the clients and authenticating the client applications. FTP server is responsible from storing, sending the course contents and also authenticating the client applications.

3. VoD Clients:

- VoD Client Side is based on the requirements of ADE systems and three main interfaces are designed; a VoD Project Editor to design DE projects, a VoD Player to view DE project and another VoD Player which is an enhanced version that enables chatting with online users and the instructors. VoD Client side also allows the users to facilitate from VoD shared resources such as discussion boards and additional course information available in the VoD web site.

4. VoD Client Applications:

- VoD Project Editor is designed to handle VoD projects design, implementation, editing and sending to the Content Servers by FTP while registering them on the Mediation Server. The project design process for ADE courses includes a video and course related supporting material as “html” pages. VoD Project editor is enriched with the various “html” editors and a synchronization API for the mentioned “html” pages synchronization to the course video.
- The main concerns of VoD Players are authenticating users on Mediation Server, access to project database through Mediation Server, access to project contents through Content Servers and presenting ADE projects. VoD Players are designed to fulfill the requirements of ADE systems while giving the end user the multi-protocol facilities of the underlying VoD System with a user friendly interface. VoD Players are capable of streaming multimedia content and able to browse “html” pages in its Html Browser.
- VoD Players are classified into two categories by their functionalities: A standard VoD Player that can handle the tasks mentioned above and an Improved VoD Player that allows real-time chatting facilities for the users who prefer more interactivity.
- VoD Players allows the learners to add their content material to the VoD projects by using the built-in “Drag and Drop” facility. The user contributions are saved in local computer the same way they stored in the Content Servers therefore in the following openings of the VoD projects, the user contributions are also show in the order they are added.

The current networking technologies are examined and seen that none of the protocols mentioned fulfill the requirements of video transmission in the expected efficiency, some of them are too reliable with high overhead while the others could not

pass the reliability requirements. A need to light weight reliable transmission protocol was investigated. Therefore a new application protocol called Reliable UDP (RUDP) is developed for the transmission of the DE videos from Content Servers to VoD Players.

The strategies, Instant Playing and Smooth Buffering are applied to the streaming system of VoD Players. Before activating the DE video, VoD Player decides whether to apply Instant Playing or Smooth Buffering is described in the thesis.

In order to test the new transmission protocol RUDP, the other network protocol suites such as TCP, FTP and UDP are also made available of the transmission of DE videos. Studies of the protocol speeds are handled and the results are described with the statistical results.

The distributed network architecture made the system applicable over divided university campuses and decreased the effects of limited network bandwidth, huge amounts of storage requirements and fragmentation effects of huge storages.

We believe that, we developed a pedagogically enhanced, more understandable and more interactive DE system powered with video and content related supporting material additivity and the developed underlying VoD system to reduce the disadvantages of the broadband data transmission.

8.2 Further Work

The further work could be summarized as follows:

- The effects of pedagogically enhanced ADE VoD System on learners could not be tested in the mean time. This thesis makes an infrastructure for other studies on the improvements of DE systems.
- The current VoD Player applications are only compatible with Microsoft Windows operating systems and could be adapted to other operating systems.
- Other protocol stack for video transmissions, Real Time Transport Protocol (RTP) could also be added to the system in order to test the protocol behaviors more precisely on video transmissions.
- A tool could be designed to move the user contributions to other computers or the user contributions could also be stored in the Content Servers to ease the information sharing.
- Real time video support could be added to the system in order to cover whole DE scenarios, i.e. synchronous and asynchronous DE.

APPENDIX

APPENDIX A
DATA TYPES AND CUSTOM STRUCTURES

Synchronization Data

```
typedef struct _PAGES
{
    int PageID;                // Page ID
    TCHAR PageName[MAX_PATH]; // Page name
    int Frame;                 // Sync. Frame
    BOOL PageSeenBefore;      // Disallows a page to refresh
} PAGES;                      // Page structure

typedef struct _PAGESDAT
{
    int SliderRange;          // Slider range in frames
    int PageCount;           // Page count
    PAGES Pages[200];        // Page structure
    TCHAR Path[MAX_PATH];    // Path of the html page
    __int64 TotalTime;       // Total video time in 100 nanoseconds
} PAGESDAT;                 // Structure Pagesdat
```

TCP Data Formats for VoD

```
#define QUIT      1    // Quit action
#define AUTH     2    // Authenticate client application
#define SCD      3    // Set Directory *
#define READ     4    // Read file *
#define SET      5    // Set pointer to * location on the file
#define SETREAD  6    // Set file to be read
#define GFS      7    // Get File Size
```


RUDP Data Formats and Constants

```

#define QUIT      1    // Quit action
#define AUTH     2    // Authenticate
#define SCD      3    // Set Directory *
#define READ     4    // Read file *
#define SET      5    // Set file pointer
#define SETREAD  6    // Set file to be read
#define GFS      7    // Get File Size
#define RREAD    8    // Reliable read
#define ROK      9    // Reliable OK ACK
#define RFAIL   10   // Reliable Failed ACK
#define CS      11   // Speed of the connection *

#define BUFFER_SIZE 16384 // UDP buffer size
#define BUFFER_R    65467 // Reliable UDP Buffer Size
typedef struct _RU {
    char ptBuf[BUFFER_R]; // Send buffer
    UINT Size;           // Size of the ptBuf data
    int CRC;             // CRC code of ptBuf
    UINT PacketNumber;  // Packet number
} RU; // Reliable UDP data structure

```

Success – Error codes

Code number	Success – Error text	Reason
001	Starting worker thread %	Shows this code when a UDP/RUDP thread is assigned to a call.
002	Exiting worker thread %	Shown when a thread gets to a free state by reasons QUIT or TIMEOUT
004	Initializing socket	Send before implementing socket() function
005	Binding socket	Send before implementing

		bind() function
010	Sent % kb	The total sent bytes for the session
020	Authenticating user	Send before authenticating the client application
021	Directory % cannot be set	Send when a directory set fails
022	% file cannot be opened	Send when a video file cannot be opened
023	Started to sending file %	Shown when a thread starts sending the video
028	Sending file size	Sending the video size to the client
031	Sending R packet	Shown when sending RUDP packets

APPENDIX B

Explanations of the VoD Database Tables

Table: Faculty

Field	Type	Null	Default	Description
<u>facid</u>	int(1)	No		Internal ID
name	varchar(50)	Yes	NULL	Faculty name

Table: Department

Field	Type	Null	Default	Description
<u>depid</u>	int(1)	No		Internal ID
facid	int(1)	No	0	Faculty ID that the department belongs to
name	varchar(50)	Yes	NULL	Department name

Table: Course

Field	Type	Null	Default	Description
<u>csid</u>	int(1)	No		Internal ID
depid	int(1)	No	0	Department ID that the course belongs to
cscod	varchar(10)	Yes	NULL	Course code
name	varchar(50)	Yes	NULL	Course name
instructor	varchar(60)	Yes	NULL	Course instructor
nofst	int(1)	Yes	NULL	Number of students attending to the course
coursedes	Text	Yes	NULL	Description of the course

Table: Chapter

Field	Type	Null	Default	Description
<u>chid</u>	int(1)	No		Internal ID
name	varchar(100)	Yes	NULL	Chapter name
csid	int(1)	No	0	Course, that the chapter belongs to
chapterdesc	text	Yes	NULL	Chapter description

Table: Project

Field	Type	Null	Default	Description
<u>prid</u>	int(1)	No		Internal ID
name	varchar(100)	Yes	NULL	Name of the project
chid	varchar(10)	Yes	NULL	Chapter ID that the project belongs to
subject	varchar(200)	Yes	NULL	Subject of the project
projectdesc	text	Yes	NULL	Project description
VideoPath	varchar(255)	Yes	NULL	Absolute path of the video located in the content server.
VideoName	varchar(255)	Yes	NULL	Video file name
ftpid	int(1)	No	1	Content server ID
userid	int(1)	Yes	NULL	The creator of the project
STATUS	char(1)	No	1	Status of the project, active or disabled
recorddate	timestamp(12)	Yes	NULL	Project creation time
note	text	Yes	NULL	Additional notes on project

Table: Content Servers, formerly Ftp

Field	Type	Null	Default	Description
<u>ftpid</u>	int(1)	No		Internal ID
Host	varchar(220)	No		Content server host IP
Port	tinyint(1)	No	21	Host port
User	varchar(20)	Yes	NULL	Login name
Pass	varchar(20)	Yes	NULL	Login password
dsc	varchar(64)	Yes	NULL	Description
active	char(1)	No	0	Status, active or disabled
recorddate	timestamp(12)	Yes	NULL	Creation time
note	text	Yes	NULL	Additional notes

Table: User

Field	Type	Null	Default	Description
<u>userid</u>	int(1)	No		Internal ID
user	varchar(50)	Yes	NULL	Login name
pass	varchar(50)	Yes	NULL	Login password
priv	tinyint(1)	No	0	Privilege of the user, i.e. student, instructor or administrator.
title	varchar(50)	Yes	NULL	Title of the user
name	varchar(50)	Yes	NULL	Name
surname	varchar(50)	Yes	NULL	Surname
mail	varchar(50)	Yes	NULL	e-mail address
STATUS	char(1)	Yes	1	Status of the user, active or disabled
recorddate	timestamp(12)	Yes	NULL	User creation time
note	text	Yes	NULL	Additional notes on user

Table: Sessions

Field	Type	Null	Default	Description
<u>id</u>	int(10)	No		Internal ID
sesid	varchar(50)	No		Session ID of the current web session
userid	int(1)	No	0	User ID
time	timestamp(14)	Yes	NULL	Time

Table: System

Field	Type	Null	Default	Description
<u>id</u>	int(1)	No		Internal ID
cookieName	varchar(100)	No		Cookie name stored on the client computer
exptime	varchar(15)	No		Expiration time of the web session
hkey	varchar(100)	No		Hash key used to encrypt the cookie using MD5 algorithm
myserver	varchar(50)	No		Mediation server IP address for web applications of the VoD system
myuser	varchar(50)	No		Login user name
mypass	varchar(50)	No		User password
mydatabase	varchar(50)	No		Database name
cemail	varchar(100)	No		Contact e-mail
eemail	varchar(100)	No		Error e-mail
faqemail	varchar(100)	No		FAQ e-mail
adminname	varchar(100)	No		Administrator name
recorddate	timestamp(14)	Yes	NULL	Creation time
note	text	No		Additional notes

Table: Statistics

Field	Type	Null	Default	Description
<u>id</u>	int(1)	No		Internal ID
prid	int(1)	No	0	Project ID
numofrev	int(11)	No	0	Number of reviews
numofunrev	int(11)	No	0	Number of partially reviews
recorddate	timestamp(12)	Yes	NULL	Creation date
note	text	Yes	NULL	Additional notes

REFERENCES

- [1] ActiveX, <http://www.microsoft.com/com/tech/activex.asp>
- [2] Aggarwal, C., C., Wolf, J., L., and Yu, P., S., On Optimal Batching Policies for Video-on-Demand Storage Servers., IEEE Int. Conf. on Multimedia Systems'96, Hiroshima, Japan, pp. 253 - 258, June 1996.
- [3] Aleksic-Maslac, K., and Jeren, B., Asynchronous Distance Learning Model, Int. Conf. on Engineering Education (ICEE'01), Oslo, Norway, pp. 7B2-13-16, 2001.
- [4] Allcock, B., Bester, J., et al, Data Management and Transfer in High-Performance Computational Grid Environments, Parallel Computing Journal, Vol. 28 (5), pp. 749-771, May 2002.
- [5] ANSI-ISO-9075, Database Language SQL, 1995.
- [6] Apple QuickTime, <http://developer.apple.com/quicktime/>.
- [7] Bhushan, A., The File Transfer Protocol, Internet Engineering Task Force, RFC 354, 1972.
- [8] Box, D., Essential COM, Addison-Wesley, 1997.
- [9] Chan, G., Distributed Servers Architecture for Networked Video Services, IEEE/ACM Transactions on Networking, Vol. 9, Issue 2, pp. 125 - 136, April 2001.
- [10] Chiariglione, L., MPEG-4 FAQs, Technical Report, ISO/IEQ JTC1/SC29/WG11, July 1997.
- [11] Comer, D., E., Computer Networks and Internets, Prentice Hall, New Jersey, 1999.
- [12] Component Object Model, <http://www.microsoft.com/com/default.asp>

- [13] Dan, A., and Sitaram, D., A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads, Proc. of Multimedia Computing and Networking Conf., San Jose, CA, United States, pp. 344 - 351, 1996.
- [14] Dan, A., Sitaram, D., and Shahabuddin, P., Scheduling Policies for an On-demand Video Server with Batching, Proc. of the 2nd ACM Int. Conf. on Multimedia, San Francisco, California, United States, pp. 15 – 23, 1994.
- [15] Feamster, N., and Balakrishnan, H., Packet Loss Recovery for Streaming Video, Proc. of 12th Int. Packet Video Workshop, Pittsburgh, PA, USA, 2002.
- [16] Furht, B., Interactive Television Systems, Proc. of the ACM Symp. Applied Computing, Pennsylvania, United States, pp. 7-11, 1996.
- [17] Gottschalk, T., H., Distance Education at a Glance,
<http://www.uidaho.edu/eo/distglan.html>, May 2003.
- [18] Gramoll, K., An Internet Portal For Statics And Dynamics Engineering Courses, Int. Conf. on Engineering Education, Oslo, Norway, pp. 7B1-1-6, August 6 – 10, 2001.
- [19] Hall, B., Beej's Guide to Network Programming,
<http://www.ecst.csuchico.edu/~beej/guide/net/html/>, 2001.
- [20] He, E., Leigh, J., Yu, O., and DeFanti, T., A., Reliable Blast UDP : Predictable High Performance Bulk Data Transfer, IEEE Int. Conf. on Cluster Computing, pp. 317, Chicago, Illinois, United States, September 23 - 26, 2002.
- [21] Hyper Text Markup Language, World Wide Web Consortium,
<http://www.w3.org/MarkUp/>, 2003.
- [22] International Standard: ISO/IEC IS 11172-2, Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, 1993.
- [23] Jones, A., and Ohlund, J., Network Programming for Microsoft Windows 2nd Edition, Microsoft Press, Redmond, Washington, 2002.
- [24] Kath, R., Managing Memory-Mapped Files in Win32,

- http://msdn.microsoft.com/library/techart/msdn_manamemo.htm, Microsoft Developer Network Technology Group, 1993.
- [25] Kuo, F., et al, Multimedia Communications, Prentice Hall, Upper Saddle River, NJ, United States, 1997.
- [26] Lee, H., and Du, D., H., C., The Effect of Disk Scheduling Schemes on a Video Server for Supporting Quality MPEG Video Accesses , Int. Conf. on Multimedia Computing and Systems (ICMCS '97), IEEE, Ottawa, Ontario, CANADA, pp. 194, June 03 - 06, 1997.
- [27] Lee, W., Srivastava, J., Adaptive Disk Scheduling Algorithms for Video Servers, Int. Conf. on Parallel Processing, IEEE, pp. 363, Wakamatsu, Japan, 1999.
- [28] Leweler, D., A., and Hirschberg, D., S., Data Compression, ACM Computing Surveys, Vol. 19, Issue 3, pp. 261 – 296, New York, USA, 1987.
- [29] Loeser, C., Distributed Video on Demand Services on Peer to Peer Basis, Proc. of the Int. Workshop on Real-Time LANs in the Internet Age, Vienna, Austria, June 2002.
- [30] Ma, H., and Shin, K., G., Multicast Video-on-Demand services, ACM SIGCOMM Computer Communication Review, Vol. 32, Issue 1, pp. 31 - 43, January 2002.
- [31] Merabti, M., Pereira, R., & Sumari, P., Video on Demand Server: Strategies for Improving Performance, IEE Proc. Software, Vol. 146, No 2, March 1999.
- [32] Microsoft Developer Network (MSDN), <http://msdn.microsoft.com/>, 2003.
- [33] Microsoft DirectX, <http://msdn.microsoft.com/DirectX>, 2003.
- [34] Microsoft Foundation Classes, http://msdn.microsoft.com/archive/en-us/dnarmfc/html/msdn_mfc4tech.asp, 1995.
- [35] Microsoft Windows Media Software Development Kit, <http://msdn.microsoft.com/av/>.
- [36] Moving Picture Experts Group, <http://www.chiariglione.org/mpeg/index.htm>

- [37] MySQL Open Source Database, <http://mysql.com>
- [38] Nagle, J., Congestion Control in IP/TCP Internetworks, Internet Engineering Task Force, RFC 896, 1984.
- [39] NTFS Multiple File Streams, <http://msdn.microsoft.com/library/en-us/dnfiles/html/ntfs5.asp>, March 2000.
- [40] Open System Interconnection Model, ISO 7498, ISO/ITU, http://www.acm.org/sigcomm/standards/iso_stds/OSI_MODEL/, 1994.
- [41] Postel, J., and Reynolds, J., File Transfer Protocol, Internet Engineering Task Force, RFC 959, 1985.
- [42] Postel, J., User Datagram Protocol, Internet Engineering Task Force, RFC 768, 28 August 1980.
- [43] Project Mayo, <http://projectmayo.com/>, 2003.
- [44] Prosise, J., Programming Windows with MFC 2nd edition, Microsoft Press, Redmond, Washington, May 13, 1999.
- [45] R., A., Ellingson, 32 bit Cyclic Redundancy Check Source Code for C++, <http://www.createwindow.com/programming/crc32/>, 2003.
- [46] Real Networks, <http://real.com>.
- [47] Rousseau, F., and Duda, A., An Advanced Multimedia Infrastructure for WWW-Based Information Systems, IEEE Int. Workshop on Advance Issues of E-Commerce and Web-Based Information Systems, p. 108, Santa Clara, California, United States, April 08 - 09, 1999.
- [48] Sarwate, D., V., Computation of cyclic redundancy checks via table look-up, Communications of the ACM, Vol. 31, Issue 8, pp. 1008 – 1013, New York, NY, USA, 1998.
- [49] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., RTP: A Transport Protocol for Real-Time Applications, Internet Engineering Task Force, RFC 1889,

January 1996.

- [50] Schulzrinne, H., Rao, A., and Lanphier, R., Real Time Streaming Protocol, Internet Engineering Task Force, RFC 2326, April 1998.
- [51] Sollins, K., The TFTP Protocol Revision 2, Internet Engineering Task Force, RFC 1350, July 1992.
- [52] Stevens, W., R., UNIX Network Programming 2nd Edition, Prentice Hall, 1998.
- [53] Technical Committee, LAN Emulation over ATM, The ATM Forum, 1995.
- [54] Windows NT File System, <http://msdn.microsoft.com/library/en-us/fileio/base/ntfs.asp>, February 2003.
- [55] WinInet API, <http://msdn.microsoft.com/library/en-us/wininet/wininet/portal.asp>, 2003.