Scalability Improvization of VoD Systems by Optimal Profiteering

Anil Kumar Sharma[1], Harsh Bardhan[2]
Final Year Students ,
Department of CSE,
Bharath University,Tamilnadu, India

K.Rajakumari[3]
Assistant Professor,
Department of CSE,
Bharath University,Tamilnadu, India

***Abstract*- Video-on-Demand (VOD) focus more on mending service architectures and optimizing overlays but do not carefully consider the user behavior and the benefit of perfetching strategies. As a result, they cannot better support, we propose a network coding equivalent content distribution scheme to efficiently handle interactive video-on-demand (VoD) operations in peer-to-peer systems and VOD, a -oriented VOD for P2P networks. In , videos are divided into segments that are then further divided into blocks. These blocks are encoded into different peers and distributed over a local storage . Together with a hybrid caching strategy, a collaborative prefetching scheme is designed to optimize resource distribution among neighboring peers. A new streaming algorithm designed to combine random network coding with a randomized push**

Keywords*-Batching,VOD,RTP, Gossip, Hybrid Cahing,VOD Peers.*

I. INTRODUCTION

Multimedia is media and content that uses a combination of different content forms. Multimedia system consists of audio file,text file, still pictures, video,animation and interactivity content forms. Multimedia system consists of audio file,text file, still pictures, video,animation and interactivity content forms. Hypermedia can be considered one particular multimedia application. Streaming media area unit multimedia system that area unit perpetually received by and delivered to finish user by streaming supplier. The excellence is sometimes applied to media that area unit distributed over telecommunications networks, as most different delivery systems area unit either inherently .VOD is an interactive multimedia service, which delivers video content to the users. Differing live streaming, a VOD user expects to enjoy the video with completely free choices. In this work, we keep the focus on VOD to transfer the peer it currently playing.

For ease of expression, in the rest of the paper, we use the terms "client," "peer," and "node" interchangeably.In order to provide "play-as-download" VOD services, stream request techniques such as batching, patching, and chaining are proposed. Generally, the overlay construction with those techniques is tightly coupled with the peers' playback progresses.The stream reusability will be underutilized unless partnering peers keep persistent connections with each other. Consequently, user experiences are seriously degraded when they take frequent controls. To address this problem, prefetching is also employed.VOD users can be greatly improved with a better prefetching strategy. Bearing this in mind, we propose VOD, a system for large scale p2p network,that can be characterized as follows:

ISSN: 2278 – 7798

1. Scalability: We adopt the combination of batching and patching. VOD is able to serve many more concurrent clients than the original capacity of the source server.

2. Flexibility: Used hybrid caching strategy and patching, VOD provides abundant backup resource for asynchronous clients and frequent requests.

3. Short latency: VOD adopts an efficient gossip protocol and a collaborative prefetching strategy. Both the requests of joining in and the dynamic controls can be responded with very short latencies.

II. SYSTEM ANALYSIS

Existing System: Existing research in VOD, mainly focus on improving the client-server model.In the model the dedicated server get the video segments by segments.Users in asynchronously receive the streams from different channels.

In order to provide uninterrupted streaming for all users, a track of all the channels is been kept by server and ensures no interruptions exist between playing segments, which causes heavy load.Then the request is sended to the server,the user in the channnels are limited, and some channels are underutilized. VOD serves asynchronous peers with the combination of patching and batching .NO buffer record is kept by server. The requests are resolved in a distributed manner. Moreover, VOD employs a behavior-content-based scheme of prefetching.

III. SOFTWARE DESCRIPTION

1. Introduction To Java: The Java programming language and environment is designed to solve a number of problems in modern programming practice.Some problem was faced in c++ while starting of project.

2. Object-Oriented Programming: Object-Oriented Programming is at the core of Java. In fact, all Java programs are object-oriented—this isn't an option the way that it is in C++.

Two Paradigms

A program can be organized around its data and around its code.The process-oriented model can be thought of as *code acting on data.*

2.1. Abstraction: A powerful way to manage abstraction is through the use of hierarchical classifications.

2.2. Encapsulation: It binds together the data and code it manipulates, and keeps both safe from outside interference and misuse.

2.3. Inheritance*:* Inheritance is the process by which one object acquires the properties of another object.

3. JMF: The Java Media Framework (JMF) provides the object-oriented programming tools necessary to facilitate the construction of software, which delivers this content to the end-viewer.It provides functionality for capturing, processing and viewing time-based media. By abstracting these elements through the JMF the programmer can construct a program that will not only have better functionality but will also be cross platform compatible. JMF Application Programmer's Interface (JMF API) utilizes four managers, which make it easier to create new interfaces with less modification to existing code. The four managers are as follows:

3.1. Manager: Coordinates construction of Players, Processors, DataSources and DataSinks.

ISSN: 2278 – 7798

3.2. PackageManager: Contains a listing of packages that utilize JMF classes.This includes custom Players, Processors, DataSources and DataSinks.

3.3.CaptureDeviceManager: Contains a listing of available capture devices.

3.4. PlugInManager: Contains a listing of available plug-ins such as Multiplexers, Demultiplexers, Codecs, Effects and Renderers.

Demultiplexer: Extracts multiple tracks from data streams, which were multiplexed when created.

Effect: Handles processing of special effects on a da ta stream.

Codec: These handle the encoding and decoding of content types.

Multiplexer: Combines multiple audio/video tracks into a single stream for delivery.

Renderer processes data in a track and outputs it to an output device such as a screen or speaker.When designing a media player, the first step in utilizing the JMF is to acquire a data source.The difference between using a Player or Processor is in the format of the captured data. If the DataSource is to be played than a Player is used.The filename is generally a URL object.By using a Processor the programmer can, for example, capture audio from a microphone, encode it in the MP3 file format, and write it to disk for later access.The JMF can be extended through the use of plug-ins by doing additional processing on a track or through constructing new DataSources and MediaHandlers.Multiplexers handle combined one or more tracks into a single data source, which can then in turn be turned into multiple tracks by a Demultiplexer when being prepared for playback.Once the library is downloaded and installed the plug-in is referenced by the following:This provides the functionality of the MP3 decoder to the program and will allow the Player object to decode and playback music encoded with the MP3 encoding. The Player object goes through a series of steps, before starting playback of the media stream: 'Unrealized', 'Realizing', 'Realized', 'Prefetching', 'Prefetched', 'Started'.

4. JMF Player State Diagram: In addition to the two stages used by a Player, the Processor introduces a third stage called 'Configuring' during which the Processor determines if and how it will process the data being provided.The method getControlPanelComponent can be called to get the Component, which can then be inserted, into the program's interface. Streaming media means that the program must be able to keep up with the incoming data, prepare it, process it and deliver it to the end-viewer without dropping any of the information. RTPControl provides support for dynamic media data and a media Format. An RTP media locator uses the following format: When sending streaming media content the data is received from a Processor. To open a file, the menu Player Open is accessed. The player filters the file list for only the three supported file types. As "mov" files can also be video.When the file is opened, the JMF Manager creates a new Player object with the filename as the DataSource. The default controls provide the following functions:

- play
- pause
- mute & volume adjustment
- speed of playback adjustment
- slider control over position of playback

When playback is finished the player stops. The JMF provided a very easy to

ISSN: 2278 – 7798

learn and use functionality which made creating this media player.

5. JMF RTP: JMF uses API that is outlined within the package that allows transmission of RTP stream.

for example, the RTP arthropod genus is employed for telecommunication app that is employed to associate degreeswers decision and records messages like an electronic device. Outgoing RTP streams originate from capture device. The outgoing streams may also be contend regionally, saved to a file, or both.

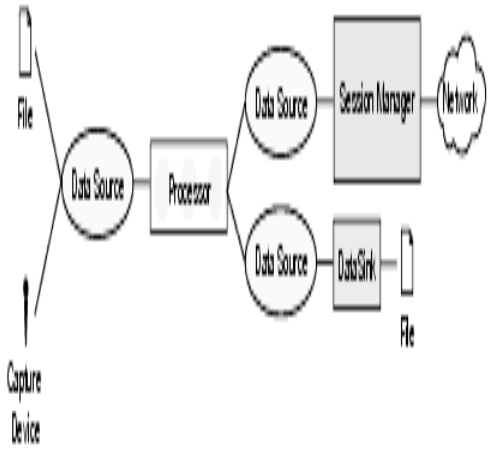


Figure: 1. RTP transmission.

6. RTP Architecture: The JMF APIs and RTP APIs are made to work with the capture, presentation, and processing capabilities of JMF**.**

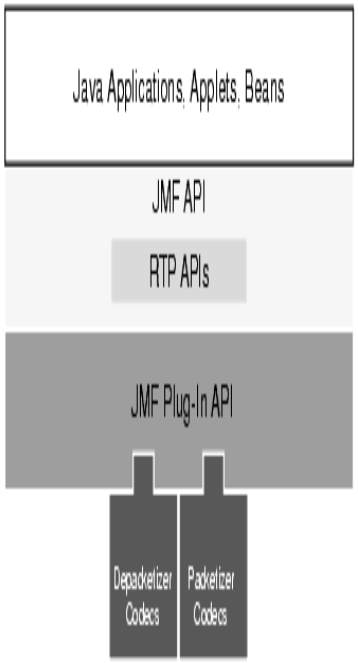


Figure:2. High-level JMF RTP architecture.

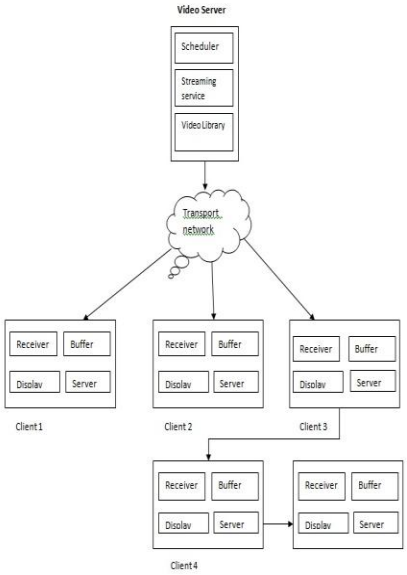


Figure: 3.Architecture Diagram

ISSN: 2278 – 7798

7. Session Manager: In JMF, a Session Manager is used to coordinate an RTP session. The session manager helps RTCP for both senders and receivers**.**

8. Session Statistics: The session manager provides access to global reception and transmission statistics:

8.1.GlobalReceptionStats: Maintains global reception statistics for the session. **8.2.GlobalTransmissionStat**s**:**Maintains cumulative transmission statistics for all local senders.

Session Participants Session Managers create a Participant whenever an RTCP packet arrives that contain SDES with CNAME that has not been seen before in the session (or has timed-out since its last use).

9.Session Streams: The Session Manager maintains an RTPStream object for each stream of RTP data packets in the session.

10. Send Stream Listener: You can implement SendStreamListener to receive notification whenever: New send streams are created by the local participant. The transfer of data from the DataSource used to create the send stream has started or stopped. The send stream's format or payload changes.

10.1.ReceiveStream Listener: There are seven types of events associated with a Receive Stream

10.1.1.New ReceiveStreamEvent : Indicates that the session manager has created a new receive stream for a newly-detected source.

10.1.2.ActiveReceiveStreamEvent: Indicates that the transfer of data has started.

10.1.3.InactiveReceiveStreamEvent: Indicates that the transfer of data has stopped.

10.1.4.TimeoutEvent: Indicates that the data transfer has timed out.

10.1.5.RemotePayloadChangeEvent: Indicates that the format or payload of the receive stream has changed.

10.1.6.StreamMappedEvent: Indicates that a previously orphaned receive stream has been associated with a participant.

10.1.7.ApplicationEvent: Indicates that an RTCP APP packet has been received.

11. Remote Listener: To receive notification of events or RTP control messages received from remote participants and to monitor the session--it enables you to receive RTCP reports and monitor the quality of the session reception without having to receive data or information on each stream.

IV. MODULES

1.VOD Server:

→Batching Plus Patching

2.VOD :

→Patching

→Hybrid Caching

→Gossip.

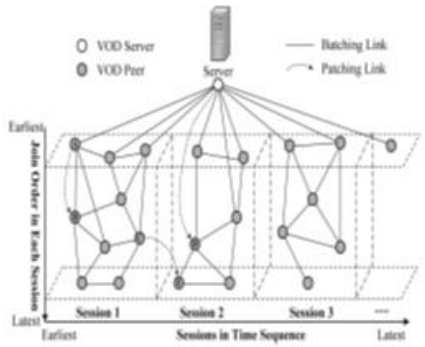
3. Modules Description:

ISSN: 2278 – 7798

3.1. VOD Server: VOD Server adopts the combination of batching and patching, as shown . Such a design is flexible with substantive concurrent and asynchronous clients.

3.1.1. Batching Plus Patching: The VOD server uses batching to serve asynchronous peers. The server allocates a certain amount of dedicated outgoing bandwidth for each batching session. In each session, early joining peers directly become the children of the server.

Figure 4. Batching and Patching



After the allocated bandwidth is fully occupied, late peers are redirected by the server and become the descendants of the early ones. Since the peers in a session transfer same video content currently bro-adcast by the server, the streaming mechanism inside a batching session is similar with P2P live streaming.Moreover,

VOD reinforces the batching scheme with patching. The server sends a list of randomly selected peers to each joining peer.When a peer joins in a session late and misses the initial part of the video, it pi cks up a few peers from the random list as patching sources and immediately starts to download the missing part from them.

3.3.VOD Peer:

3.3.1. Patching: Peers are clustered according to their arrival times and from sessions.Each sessions ,together with the server,construcys an application multicast.

Later,peers can retrive the missing parts from the server or other peers.VOD divides the peers into the generation according to there arrival time.

3.3.2. Hybrid Caching: To support asynchronous accesses to the video content. When the server starts a new batching session, it need not wait any late peers.

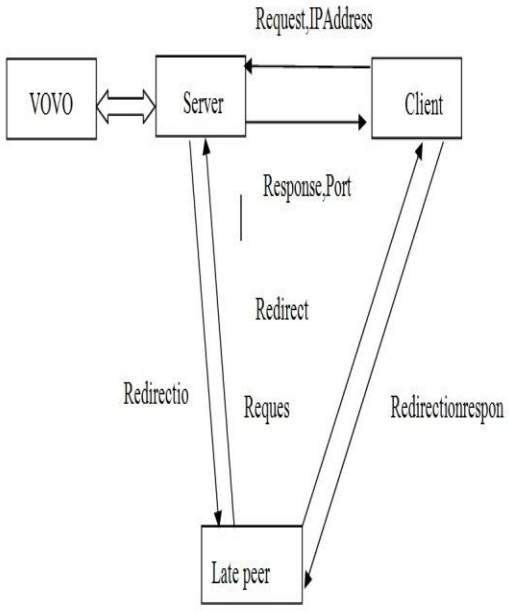


Figure.7: Hybrid Cache

The early peers in a batching session obtain the video content and become the substitute video sources. Late peers can make up the missing content using patching from the early ones. Meanwhile, patching improves

ISSN: 2278 – 7798

the system flexibility.VOD provides abundant backup stream sources for patching by adopting the hybrid caching strategy, where all the peers keep both the initial 5 minutes and the latest 5 minutes of the video played.Any late peer can instantly find patching sources and make up the missing part immediately after join, no matter if it starts from the beginning or any other offset of the video.Because th patching sources are selected randomly, load balance is kept among the peers.

3.3.3. Gossip: Peers in VOD conduct periodical gossips to exchange their state information. During each period, a peer generates a state message, including its latest state information.The format of the state message is IP, Incremental playback record, Time stamp, where IP is the peer's IP address, Incremental playback record refers to the string of segments the peer plays after it generates last state-message last time, Time stamp is the time since the peer joins in.On the other hand, each peer maintains a list of records. Each entry in the list corresponds to a peer and records its latest state.On receiving a state message, a peer performs relevant operations before it forwards the message to its neighbors:If the time stamp of the state message is greater than that in the entry, the incremental playback record is inserted into the tail of the playback record in the entry, and the time stamp in the entry is updated.Using the gossip-based state propagation, a peer is able to accumulate the information of playback history of all the peers.Furthermore, since the hybrid caching strategy is well known to all, through periodical gossips every peer can keep aware of the global distribution of video data on all the peers.

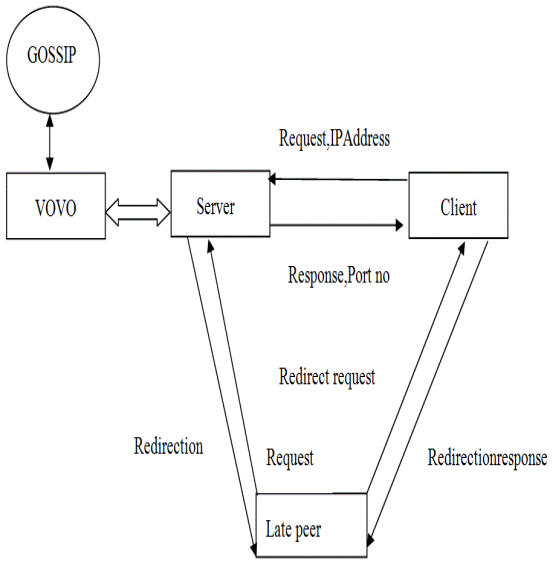


Figure 8: Gossip

3.3.4. Playback Record: Every minute of video is called a segment. A peer in VOD maintains its playback record while streaming and playing the video. The playback record is a string of segment indices, which is initially empty.When a segment is played, its index is inserted into the tail of the string.For example, suppose the current playback record of a peer is (1, 2, 3,4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 7, 8), it depicts a playback history as follows:The peer first plays the video from the first to the eighth minute, fast searches to the 11th minute, plays until the 15th minute.The reverse searches to the seventh minute, and plays the eighth minute before the playback record is last updated.

3.3.5.Association Rules: Peers in VOD take the state information collected through gossips as the input of mining.The goal of mining is to find the segments most associated to the segment currently played.First, it is observed that the user behavior in the next few minutes is closely related with his/her experience during the last few minutes.The preconfigured sizes of

ISSN: 2278 – 7798

item sets in the association rules have apparent impact on the efficiency and accuracy of mining. Thus, we propose to mine all the rules Second, the playback history of a VOD user actually forms a sequence of segments. Input and output of predictions based on the history information ought to be order sensitive.For example, a user who plays the segments (4, 5, 1) will probably continue to watch the second and the third segments, while a user who plays the segments (1, 4, 5) will probably go on with the sixth segment.Third, according to the theory of association rule mining, we find all the association rules that have a support and a confidence greater than the specified thresholds.Setting the thresholds of support and confidence helps to avoid the impact of random coincidence and improves the precision and accuracy of mining. For a particular peer A, its playback history in the last 3 minutes is denoted as an ordinal string (a1, a2, a3). Let L be the local list of records kept by peer A.Then, peer A extracts all those segments which simultaneously satisfy the following requirements: From each record in the list, three segments (if exist) at most are extracted. They do not contain the immediate next two segments after the current progress of playback, because these segments are downloaded through urgent downloading. They are the closest segments following the a3th segment in the record.

VI. TESTING

6.1. Unit Testing: Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specification and contains clearly defined inputs and expected results.

The first module had been tested that whether the data packets have been send correctly to the ingress router.The second module is tested that it performs the prevention mechanism for congestion collapse for the data packets that flows in the network.The third module is tested whether the router is successfully passing the data packets to egress router.The fourth module is tested so that to determine how rapidly each flow packets are leaving the network.

6.2. Integrated Testing: Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent.

6.3. Validation Testing: Validation testing provides final assurance that software meets all functional behavioral and performance requirements, software once validated must be combined with other system elements. At the end of integration testing, software is completely assembled as package, interfacing error have been uncovered and correction testing begins.

6.4. Verification Testing: The standard definition of Verification goes like this: "Are we building the product RIGHT?" i.e.

ISSN: 2278 – 7798

Verification is a process that makes it sure that the software product is developed the right way. The software should confirm to its predefined specifications, as the product development goes through different stages, an analysis is done to ensure that all required specifications are met. Methods and techniques used in the Verification and Validation shall be designed carefully, the planning of which starts right from the beginning of the development process. The Verification part of 'Verification and Validation Model' comes before Validation, which incorporates Software inspections, reviews, audits, walkthroughs, buddy checks etc. in each phase of verification During the Verification, the work product (the ready part of the Software being developed and various documentations) reviewed/examined personally by one or more persons in order to find and point out the defects in it. This process helps in prevention of potential bugs, which may cause in failure of the project.

VII. SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users, which it will work efficiently and effectively. It involves careful planning, investigation of the current System and its constraints on implementation, design of methods to achieve the change over, an evaluation, of change over methods. An implementation co-ordination committee based on policies of individual organization has been appointed. The implementation process begins with preparing a plan for the implementation of the system.

According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system. Implementation is the final and important phase, the most critical stage in achieving a successful new system and in giving the users confidence. That the new system will work is effective. The system can be implemented only after through testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system.

VII. CONCLUSION

Enabling large-scale VOD service in wide area Internet is crucial for many commercial applications. In order to provide a -oriented VOD service in large scale P2P networks, we propose VOD scheme. VOD combines batching and patching as the basic service architecture and reinforces it with a hybrid caching strategy. Based on the observations on user behavior and interactivities, we adopt the technique of association rule mining to exploit the associations within videos. The segments requested in interactivities are thus accurately predicted according to the information collected through gossips among peers. Moreover, a collaborative perfecting strategy is designed to optimize the resource distribution on the neighboring peers.

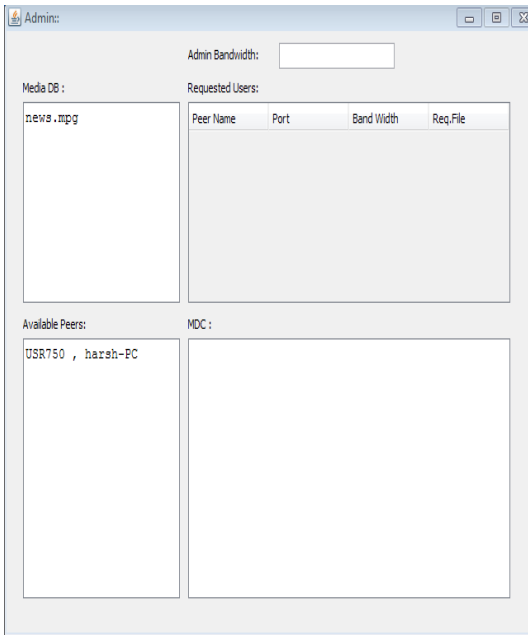
VIII. FUTURE ENHANCEMENTS

In proposed system, the segments requested in interactivities are thus accurately predicted according to the information collected through gossips among peers. In the Future, the project will

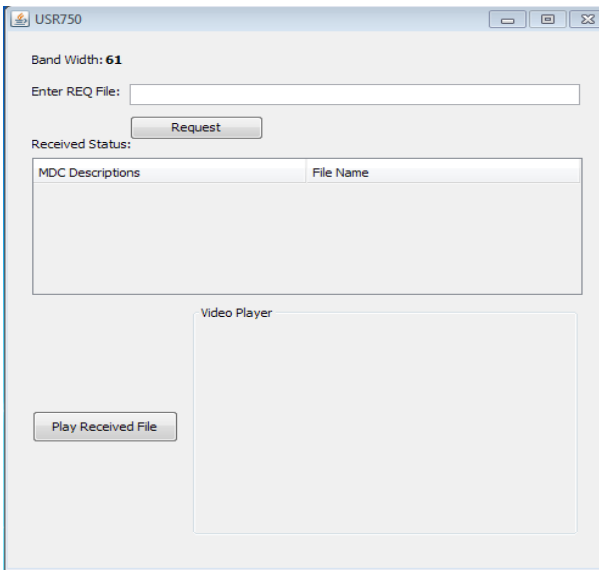
ISSN: 2278 – 7798

be developed on the in-session topology optimization with VOD for more flexibility.

APPENDIX A



A. Admin page



B.User page

REFERENCES

1. Cisco Visual Networking Index: Forecast and Methodology, 2011–2016 (2012, May 30) [Online]. Available: www.cisco.com.

2. M. Huadong and G. S. Kang, "Multicast video-on-demand services," ACM SIGCOMM Comput. Commun. Rev., vol. 32, no. 1, pp. 31–43, Jan. 2002.

3. F. Thouin and M. Coates, "Video-on-demand networks: Design approaches and future challenges," IEEE Netw., vol. 21, no. 2, pp. 42–48, Mar.–Apr. 2007.

4. K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," in Proc. WWW '01, 2001, pp. 36–44.

5. A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an ondemand video server with batching," in Proc. ACM Multimedia, 1994, pp. 15–23.

6. W. K. S. Tang, E. W. M. Wong, S. Chan, and K. T. Ko, "Optimal video placement scheme for batching VoD services," IEEE Trans. Broadcast., vol. 50, no. 1, pp. 16–25, Mar. 2004

7. K.Rajakumari, C. Nalini, "Improvement of Image Quality Based on Fractal Image Compression" in Middle-East Journal of Scientific Research 20 (10): 1213-1217, 2014, ISSN 1990-9233, © IDOSI Publications, 2014.

ISSN: 2278 – 7798