CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

The main aim of this project is to handle multiple video requests with help of super source point in the network. Mobile video traffic is expected to increase by 16folds in the next five years, accounting for over 66 percent of global data traffic in mobile networks by 2017. Evolved Packet Core is an all IP core network, consisting of packet data network gateways and serving gateways. Content providers host a large amount of online videos in their content distribution networks. Video request is initiated by the user to a CP and receives the response from the same. This requires extremely high data rate, stable connection and low experienced latency. So transparent caching in EPC moves video clips close to user device, so that it can effectively improve the quality of experience of customers. Hence a cache is built in mobile networks, where selected popular videos are stored therein. Serving gateways are now interconnected in a flat topology with built in caching. A video request received by a serving gateway will be parsed and checked whether the requested video is available at its local cache. If so, the corresponding serving gateway serves the request directly. When multiple users request the same video, multiple concurrent flow problem occurs. To solve this problem Super source point is created for each video. It handles the multiple requests for respected video and selects the server. The server selection minimizes the maximum link utilization and minimizes the cost. When the traffic intensity is low, the super source point will adopt maximum link utilization, which is an optimal solution. If traffic is high, super source point aims to minimize the cost.

After gateway selection, the server responds to the video request and transmits the data. The data is transferred via the intermediate. So the super source point distributes flow splitting table to each intermediate node. The table contains receiving node, destination address, next hop neighboring node.

CHAPTER 2

LITERATURE SURVEY

Project Title: Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery

Author Name: Vijay Kumar Adhikari

Published in: 2012

Netflix is one of the leading internet subscription service companies. Understanding the Netflix architecture and its performance can shed light on how to best optimize its design as well as on the design of similar on-demand streaming services. In this paper, they performed a measurement study of Netflix to uncover its architecture and service strategy. Netflix employs a blend of data centers and Content Delivery Networks (CDNs) for content distribution. A periodic computation of the three CDN's employed by Netflix is done, in order to measure the delivery bandwidth. Finally, as improvements to Netflix's current CDN assignment strategy, they proposed a measurement-based adaptive CDN selection strategy and video delivery strategy, and demonstrated their potentials in significantly increasing user's average bandwidth.

Project Title: Optimal Content Placement for a Large-Scale VoD System

Author Name: David Applegate, Aaron Archer.

Published in: 2014

PTV service providers offering Video-on-Demand (VoD) typically have many servers at each metropolitan office to store all the videos in the library. It will soon become infeasible to replicate the entire library at each office as the video on demand library size is increasing. Hence they have presented an approach for intelligent content placement that scales to large VoD library sizes. The problem is formulated as a mixed integer program (MIP) that takes into account constraints such as disk space, link bandwidth, and the skew in content popularity. To overcome the challenges of scale, it is observed that a Lagrangian relaxation-based decomposition technique can find a near-optimal solution (e.g., within 1-2%) with orders of magnitude speedup. Simple strategies are adopted to address practical issues such as popularity estimation, content updates, short-term popularity fluctuation, and frequency of placement updates. Using traces from an operational system, it is seen that this approach significantly outperforms simpler placement strategies. For instance, the MIP-based solution can serve all requests using only half the link bandwidth used by LRU cache replacement policy. The tradeoff between disk space and network bandwidth is also investigated.

Project Title: A Collaborative Framework for In-network Video Caching in Mobile Networks

Author Name: Jun He, Honghai Zhang

Published in: 2015

The need to place frequently accessed information close to the requestor is becoming progressively important. However, this is a difficult problem due to the large number of online videos and video requests, limited capacity of caching nodes, and limited bandwidth of in-network links. . In this paper, they propose a dynamic collaborative video caching framework to be deployed in mobile networks. The caching problem is decomposed into a content placement sub-problem and a source-selection sub-problem. SRS (System capacity Reservation Strategy) is then developed to solve the content placement sub-problem, and LinkShare, an adaptive traffic-aware algorithm to solve the source selection. Our framework supports congestion avoidance and allows merging multiple requests for the same video into one request. We carry extensive simulations to validate the proposed schemes. Simulation results show that the SRS algorithm achieves performance within $1 - 3\%$ of the optimal values and LinkShare significantly outperforms existing solutions.

Project Title: Coordinating In-Network Caching in Content-Centric Networks: Model and Analysis

Author Name: Yanhua Li

Published in: 2015

In-network content storage has become an inherent capability of routers in the content-centric networking architecture. This raises new challenges in utilizing and provisioning the in-network caching capability, for instance, how to optimally provision individual routers' storage to cache contents, so as to balance the trade-offs between the network performance and the provisioning cost. To address this problem, they proposed a holistic model to characterize the network performance of routing contents to clients and the network cost incurred by globally coordinating the in-network storage capability. An optimal strategy is derived for provisioning the storage capability that optimizes the overall network performance and cost, and analyze the performance gains via numerical evaluations on real network topologies and is also demonstrated which has achieved a significant gain on the load reduction at origin servers.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The core network of the system namely Evolved Packet Core is an all IP core network, consisting of packet data network (PDN) gateways (P-GWs) and serving gateways. Content providers host a large amount of online videos in their content distribution networks. A video request is initiated by the user to a CP and receives response from the CP.

Problem Definition:

- Data delivery is inefficient
- Requires extremely high data rate and stable connection
- Long Distance relay path also increase the complexity

3.2 PROPOSED SYSTEM

Strategically route the video requests, so as to optimize the distribution of network traffic. Proposed system jointly considers the server selection problem and the flow routing problem. In order to facilitate deployment of solutions in a real system, a complete routing protocol is developed using the output from the solutions. It also addresses the video request routing problem with simultaneous server selection and flow routing in cache-enabled networks, in which multiple sources and multiple paths for each request are considered. Proposed solutions include hop-by-hop forwarding routing decisions, which lead to a complete request routing protocol.

3.2.1 SHORTEST PATH ALGORITHM

The shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. In the following algorithm, the code u ← vertex in *Q* with min dist[u], searches for the vertex *u* in the vertex set *Q* that has the least dist[u] value. length (u, v) returns the length of the edge joining two nodes u and v.

ALGORITHM

```
Function Djikstra (Graph, source):
create vertex set Q
for each vertex  v in graph:
    dist[v]← Infinity
    add v to Q
    dist [source] ← 0
while Q is not empty:
    u ← vertex in Q with min dist[u]
remove u from Q
for each neighbour v of u:
    alt  ← dist[u] +length(u,v)
if alt < dist[v]:
    dist[v]←alt
    prev←u
return dist[],prev[]
```

The router computes the shortest path as the sum of the weights on the link. This information helps in forwarding the content to the next hop. The super source point contains information regarding traffic conditions and video availability.

3.2.2 BINARY SEARCH TREE

A binary search tree is a binary tree data structure that works based on the principle of binary search: the nodes of the tree are arranged in sorted order, and traversal of the tree is performed using a logarithmic time binary search-like algorithm. Insertion and deletion also require logarithmic time in binary search trees. This is faster than the linear time insertion and deletion of sorted arrays, and binary trees retain the ability to perform all the operations possible on a sorted array, including range and approximate queries.

However, binary search is usually more efficient for searching as binary search trees which will most likely be imperfectly balanced, resulting in slightly worse performance than binary search. This applies even to balanced binary search trees, binary search trees that balance their own nodes—as they rarely produce optimally-balanced trees—but to a lesser extent. Although unlikely, the tree may be severely imbalanced with few internal nodes with two children, resulting in the average and worst-case search time approaching comparisons.

PSEUDOCODE

```
Current node =root;
while $(current \to data != data)$
if $(current != NULL)$
        print $current \to data$
if $(current \to data \geq data)$
```

```
      $current = current \rightarrow leftchild$ ;
else
      $current = current \rightarrow rightchild$ ;
if $(current == NULL)$
 $return$ NULL;
 return current;
```

3.2.3 HOP BY HOP FORWARDING

Routing decisions are made at every router, depending on the destination of the incoming traffic. Based on the destination, the incoming flow is split and forwarded to the next hop. Flow splitting table is created that consists of the destination, next node and source.

CHAPTER 4
REQUIREMENT SPECIFICATIONS

4.1 INTRODUCTION

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process. It lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

4.2 HARDWARE AND SOFTWARE SPECIFICATION

4.2.1 HARDWARE REQUIREMENTS

• Hard Disk	:	250GB and Above
• RAM	:	4GB and Above
• Processor	:	Intel i3 and Above

4.2.2 SOFTWARE REQUIREMENTS

- Windows 7 and above
- JDK 1.8
- Tomcat 6.0
- Net beans

CHAPTER 5

SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM

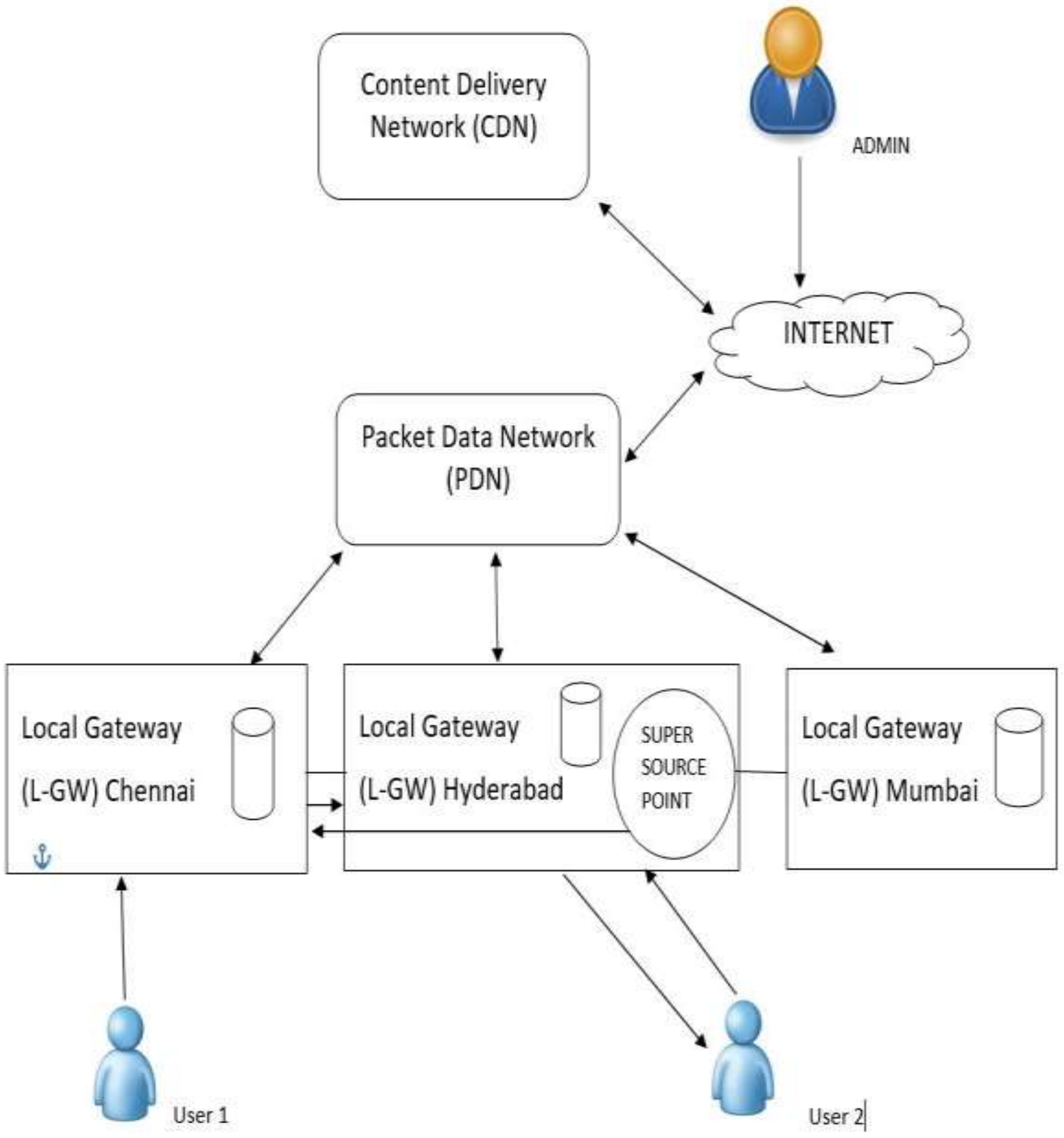

Content Delivery Network (CDN)
ADMIN
INTERNET
Packet Data Network (PDN)
Local Gateway (L-GW) Chennai
Local Gateway (L-GW) Hyderabad
SUPER SOURCE POINT
Local Gateway (L-GW) Mumbai
User 1
User 2

Fig 5.1 Video Request Routing Architecture

Traditionally, User 1 requests are directed to the local gateways with respect to one's location. The request is forwarded to the packet switched data network by the serving gateway which ultimately reaches the Content Distribution Network as illustrated in Fig 5.1.When the same video is requested by User 2 from the other location, it reaches one's respective local gateway and gets forwarded to the gateway where the video is available in its cache with the help of Super Source point.

5.2 SEQUENCE DIAGRAM

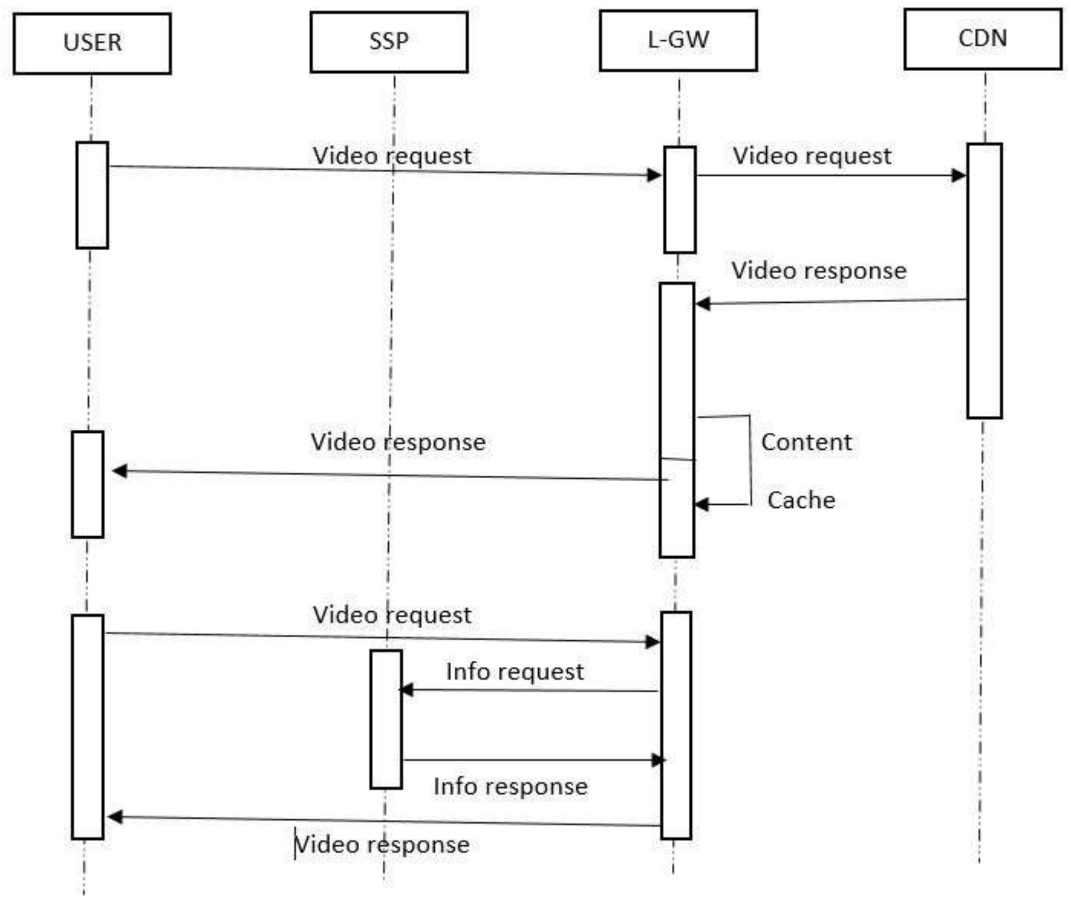


Fig 5.2 Sequence of Actions Performed by Actors

The following events are illustrated in Fig 5.2

Event 1: User initiates the video request and sends it to the CDN via local gateway.

Event 2: CDN responds to the user request by delivering the content to the local gateway. Local gateway stores the content in the local cache and sends the contents to the user.

Event 3: Another user requests for the same video is directed to the local gateway which is redirected to the super source point.

Event 4: The super source point routes the requested video to the other local gateway and reaches the user end.

5.3 USE CASE DIAGRAM

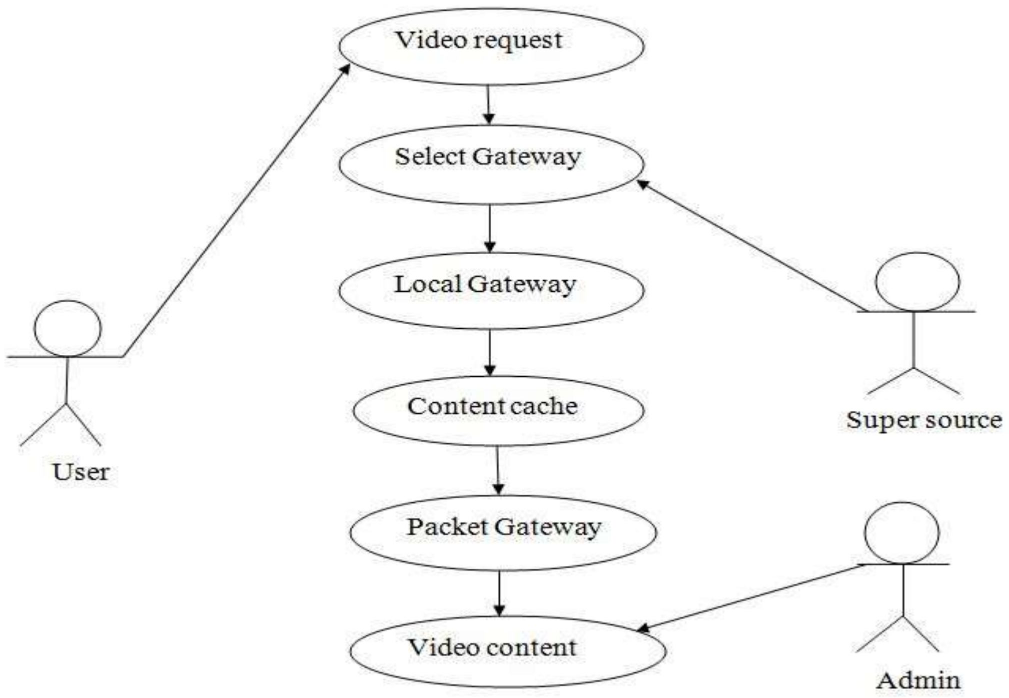


Fig 5.3 Actors and Use Case Involved in the System

The actors involved are User, Super Source point, Administrator. Administrator uploads the videos in the server. As shown in Fig 5.3, User initiates the video request by selecting the videos available in the server. The super source point directs the request to the appropriate local gateway with respect to video availability and traffic.

5.4 FLOW CHART

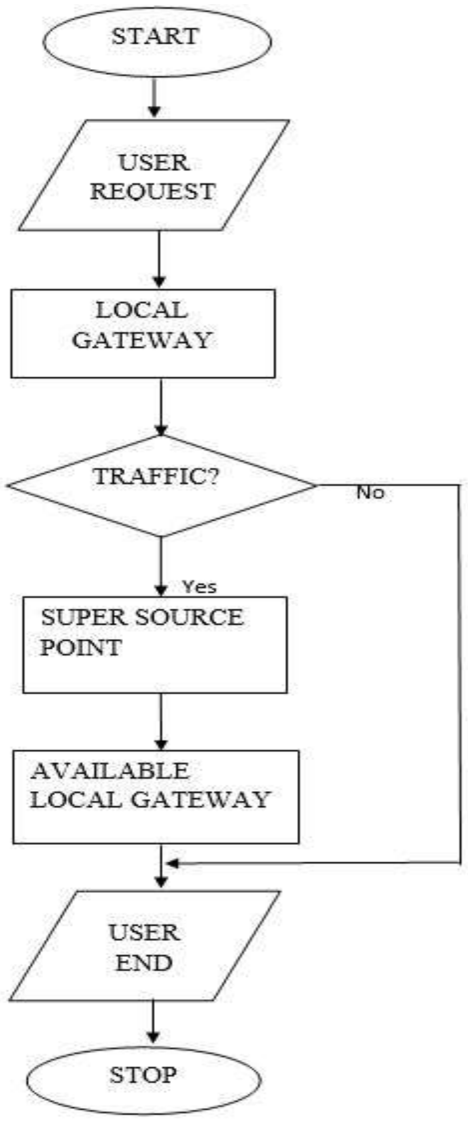


Fig 5.4 Flow of Events and Decision Making

Initially, user requests the video to the local gateway. Local gateway checks for the traffic conditions, as shown in Fig 5.4. If yes, it is forwarded to the super source point which will allocate the local gateway that has low traffic. If no, the request is served by the local gateway itself by sending the video from its cache as response.

5.5 COLLABORATION DIAGRAM

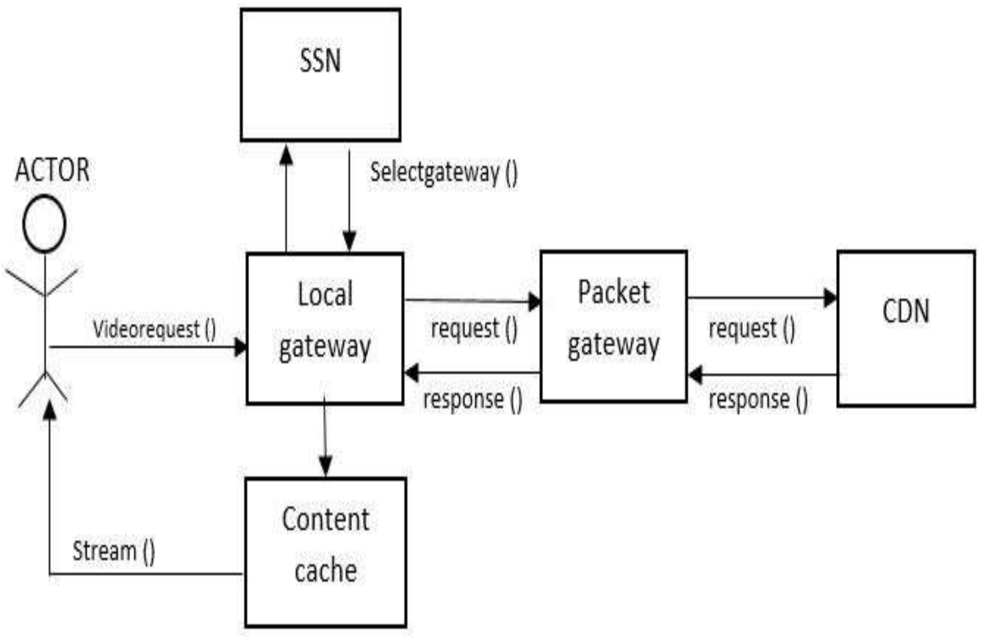


Fig 5.5 Collaboration of Methods Invoked by Actors

The user initiates the video request by invoking videorequest() method that is directed to the local gateway which ultimately reaches the CDN via PDN using request() method. The response() method is invoked by CDN to deliver the contents to the user. The method call is illustrated in Fig 5.5. The super source point invokes Selectgateway() method to determine the appropriate gateway. The local gateway streams the video content at the user end from its cache by invoking stream() method.

5.6 CLASS DIAGRAM

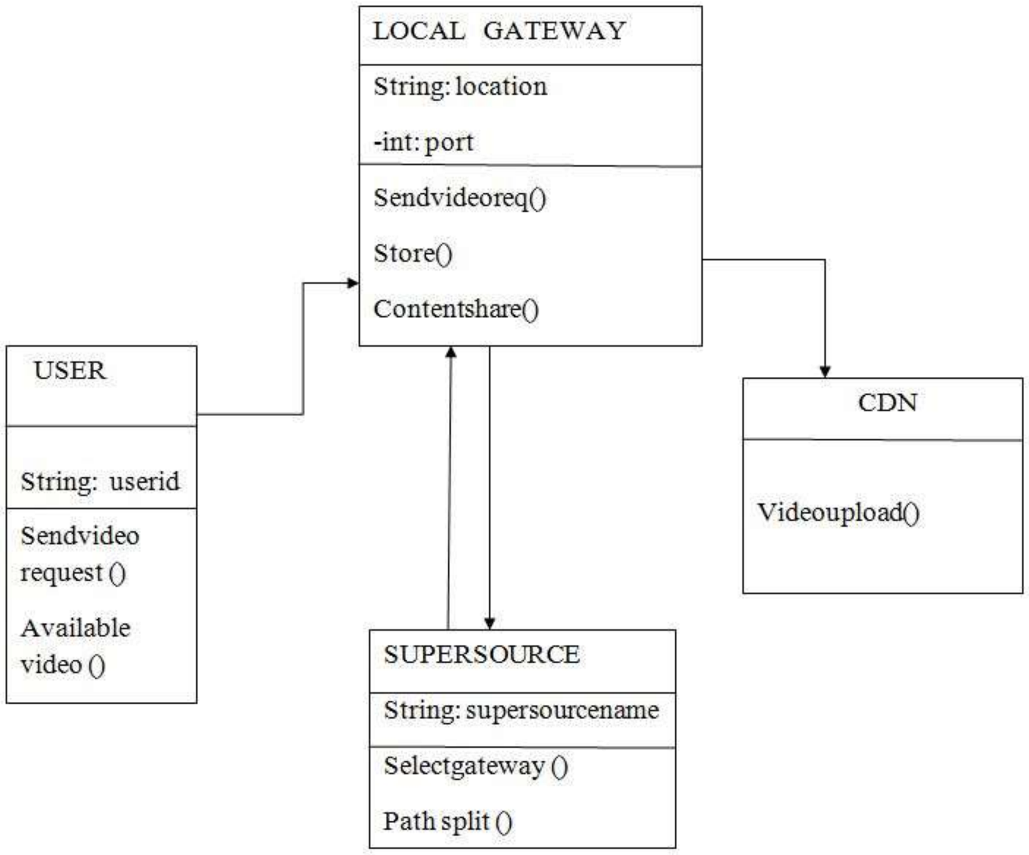


Fig 5.6 Attributes and Methods of Classes

Fig 5.6 depicts the classes namely User, Local gateway, CDN, Super source. Classes are referred with their attributes and invoked using methods. The user invokes Sendvideorequest() method to the local gateway for delivery. Local gateway will invoke the contentshare() method to responds back to the user. The Local gateway invokes the store() method to store the contents received from the server. The administrator invokes the videoupload() method to upload videos in the server. The Super source point uses Selectgateway() method and pathsplit() method to efficiently route the request.

5.7 MODULE DESCRIPTION

The entire process is divided into four modules namely,

- Upload video contents
- Network formation
- Gateway selection
- Traffic engineering

5.7.1 UPLOAD VIDEO CONTENTS

In this module admin will upload the videos as illustrated in Fig 5.7. The list of videos is available for request in the user end. Admin can access the server. Any change made in the server will reflect instantly on the user end. In this process admin is responsible to maintain videos in the server.

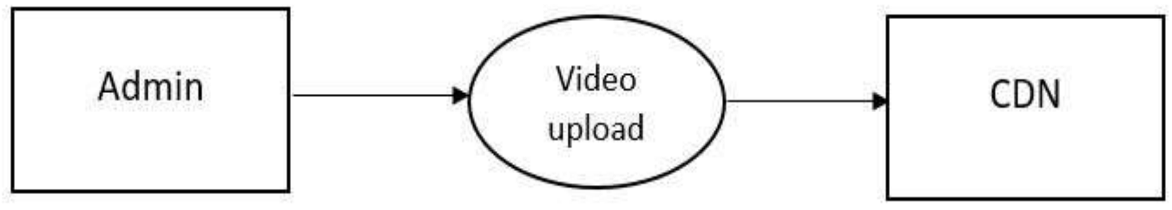


Fig 5.7 Data Flow Diagram of Upload Video Contents in the Server

5.7.2 NETWORK FORMATION

In this module local gateways are created, as shown in Fig 5.8. The gateways are differentiated based on their location. User sends the video request to the local gateway which is redirected to the CDN. The CDN responds the users by sending the requested video to the local gateway. The local gateway forwards the packet to the user and it stores the video in its cache. When another user requests the same video, the local gateway serves the video to the user. Gateways, intermediate nodes and users are created. The network is formed with the help of nodes and edges.

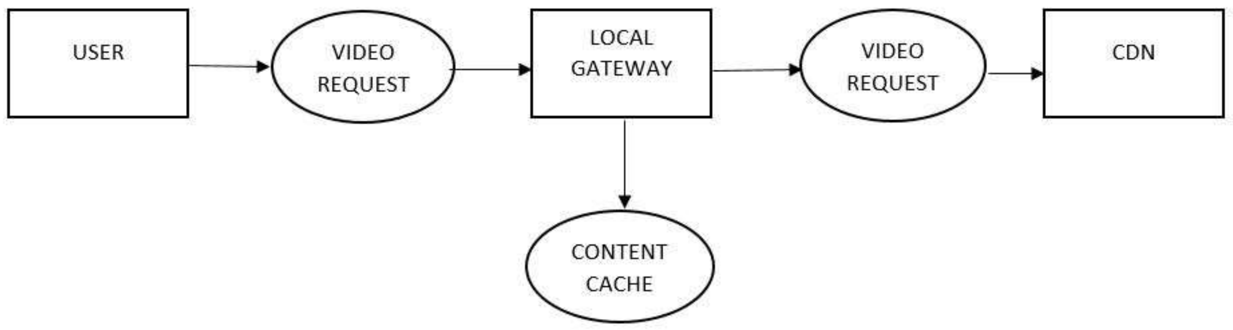


Fig 5.8 Data Flow Diagram of Creation of Gateways and Built in Content Cache

5.7.3 GATEWAY SELECTION

After building a content cache, the super source node is created for each video request. The super source node handles every respective video request, as shown in Fig 5.9. When another user requests the same video, the video request is sent to the respective super source node. The super source node receives the request from the user and then finds the serving gateway. The server selection is done based on two algorithms. They are shortest path algorithm and binary search algorithm. The shortest path algorithm reduces maximum link utilization and binary search algorithm reduces the cost. The super source node receives the request and check with their local gateway traffic. If it is high the super source node implements binary search algorithm to find the local gateway, and if the traffic is low the super source node selects the shortest path algorithm.

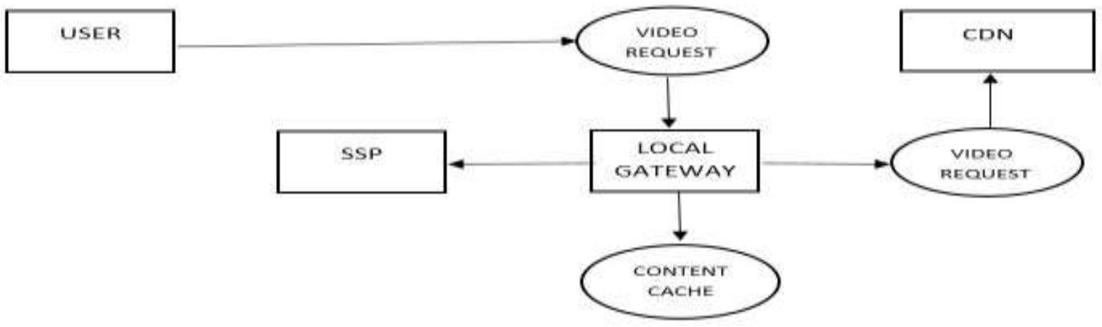


Fig 5.9 Data Flow Diagram of Selection of Gateways using Super Source Point

5.7.4 TRAFFIC ENGINEERING

The local gateways have large number of video clips and each video clip may have a set of source nodes, as shown in Fig 5.10. So Data is transmitted in hop by hop fashion which creates flow splitting table for each node. The table contains the receiving node, destination address and next hop neighboring node. The flow splitting table will be used by the routing protocol.

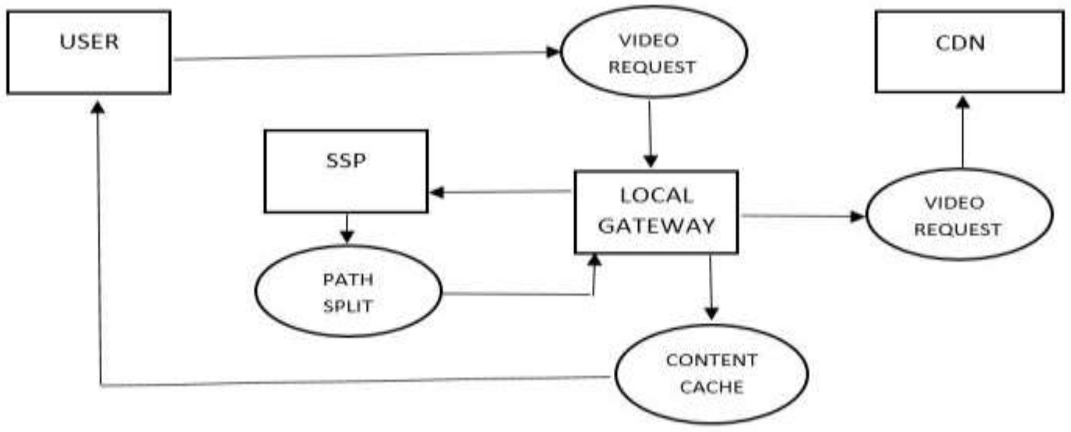


Fig 5.10 Data Flow Diagram of Path Splitting and Routing Contents to the Gateways

CHAPTER 6

IMPLEMENTATION AND RESULTS

6.1 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

- Program should be simple, clear and easy to understand.
- Naming conventions
- Value conventions
- Script and comment procedure
- Message box format
- Exception and error handling

6.1.1 NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be self-descriptive. One should even get the meaning and scope of the variable by its name. The conventions are adopted for easy understanding of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

6.1.2 VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

6.1.3 SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

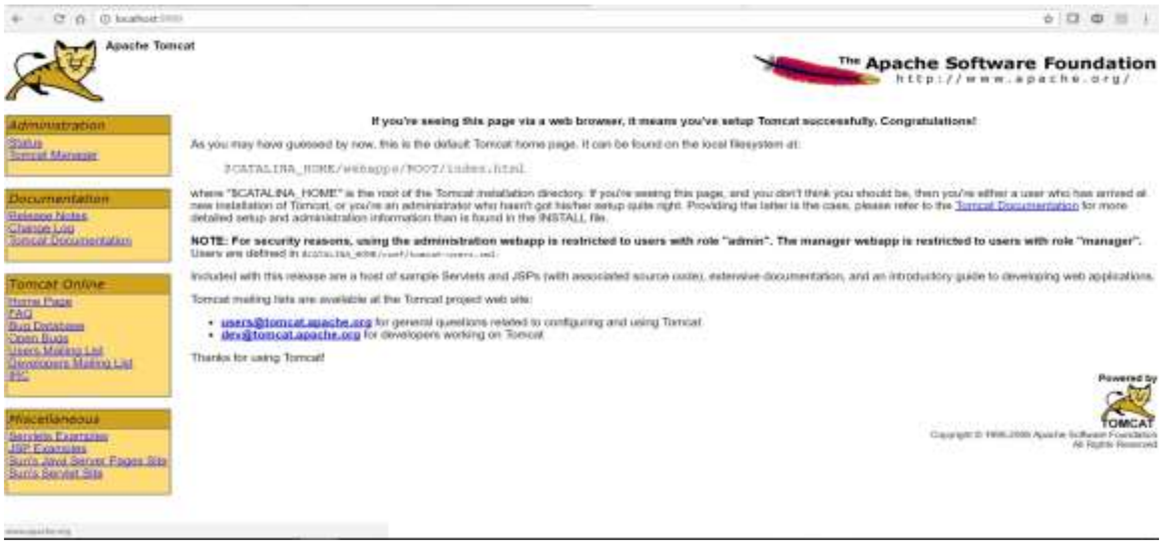
6.1.4 MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

6.2 SERVER PROCESS

Tomcat runner is executed for Catalina service. The service is initiated and brought to a runnable state with the corresponding Ipaddress and port number,as shown in Fig 6.1. The Ipaddress is entered in the URL of the web browser in order to communicate with the tomcat server. This enables the http request and response to and from the server for executing JSP files that are stored in the webapps folder present in the server.



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in $CATALINA_HOME/conf/tomcat-users.xml.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

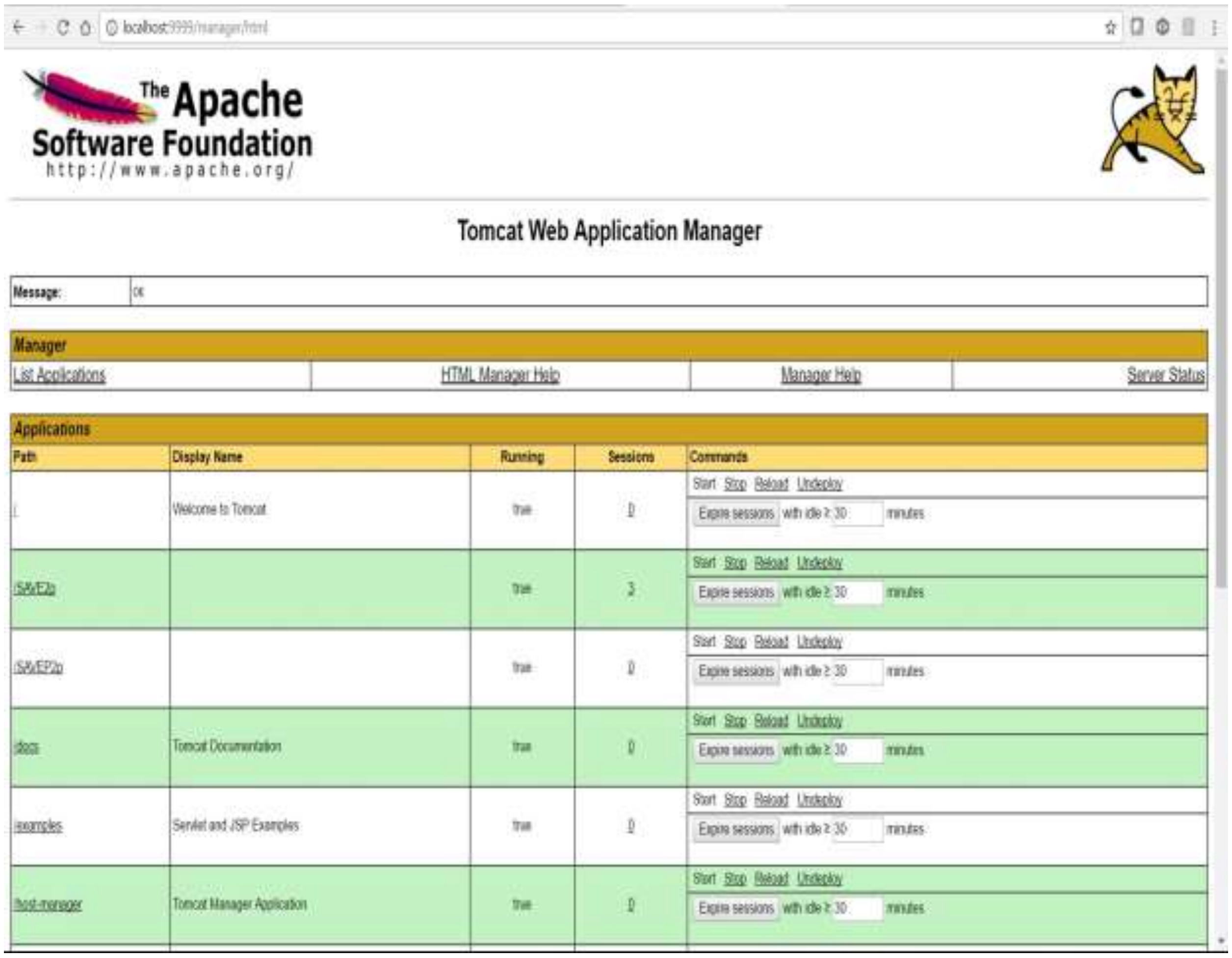
- users@tomcat.apache.org for general questions related to configuring and using Tomcat
- dev@tomcat.apache.org for developers working on Tomcat

Thanks for using Tomcat!

Copyright © 1999-2006 Apache Software Foundation
All Rights Reserved

Fig 6.1 Screenshot of Apache Homepage

The administrator selects the file manager in tomcat webpage which displays all the existing folders stored in webapps, as shown in Fig 6.2. The administrator selects the appropriate file for execution.



localhost:9999/manager/html

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/SAVE2p		true	3	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/SAVEP2p		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Fig 6.2 Screenshot of List of Folders Present in the Webapps

This directs the administrator to the login page, as shown in Fig 6.3. The administrator accesses the server console by logging in with the credentials. The validation is done with the help of a database stored using Mysql, as shown in Fig 6.4.

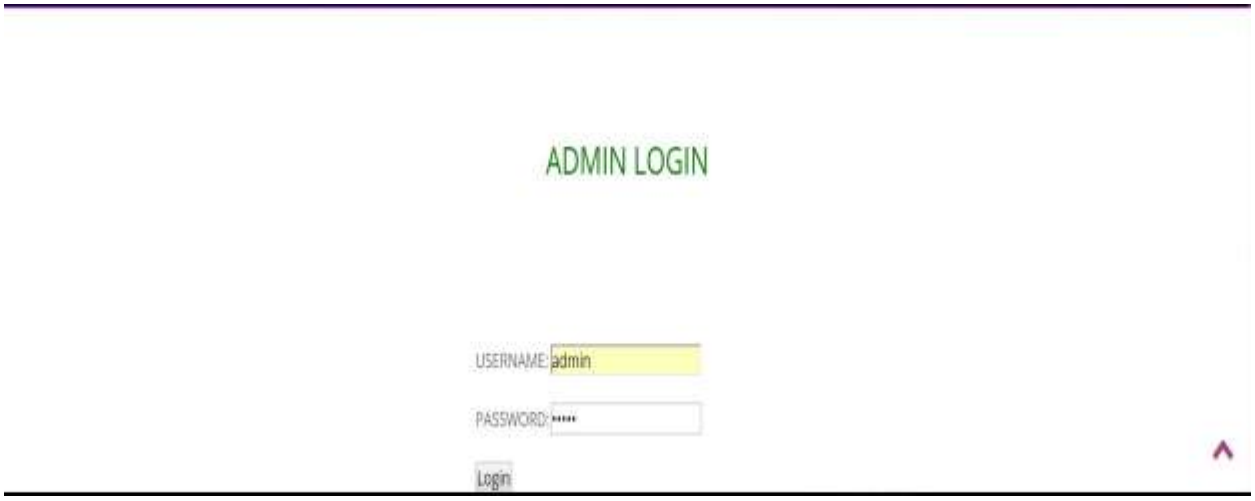
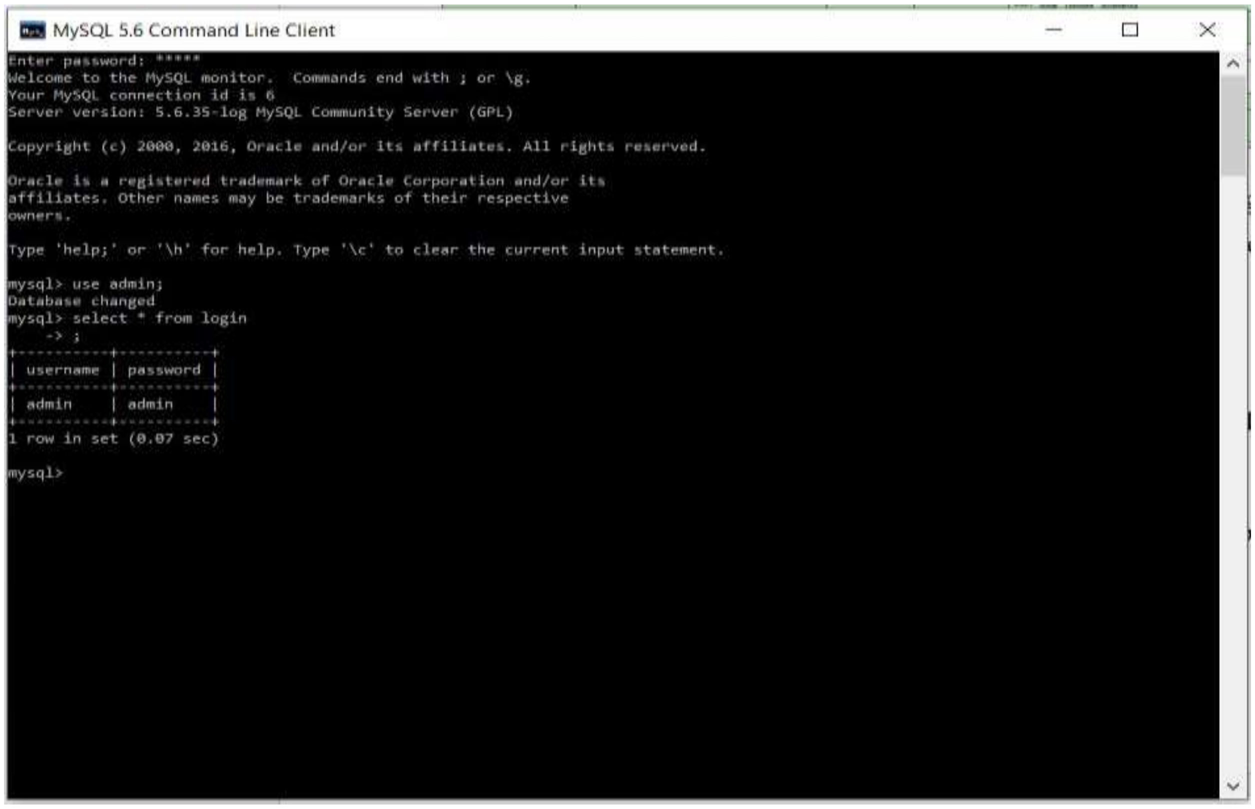


Fig 6.3 Screenshot of Administrator Login Form



```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.6.35-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use admin;
Database changed
mysql> select * from login
    -> ;
+----------+----------+
| username | password |
+----------+----------+
| admin    | admin    |
+----------+----------+
1 row in set (0.07 sec)

mysql>
```

Fig 6.4 Screenshot of MySQL Database with Validation Contents

The administrator creates a category for storing videos with same kind, as shown in Fig 6.5. The videos are chosen from the system memory and are uploaded in the server. The list of categories and the videos are displayed in the webpage.

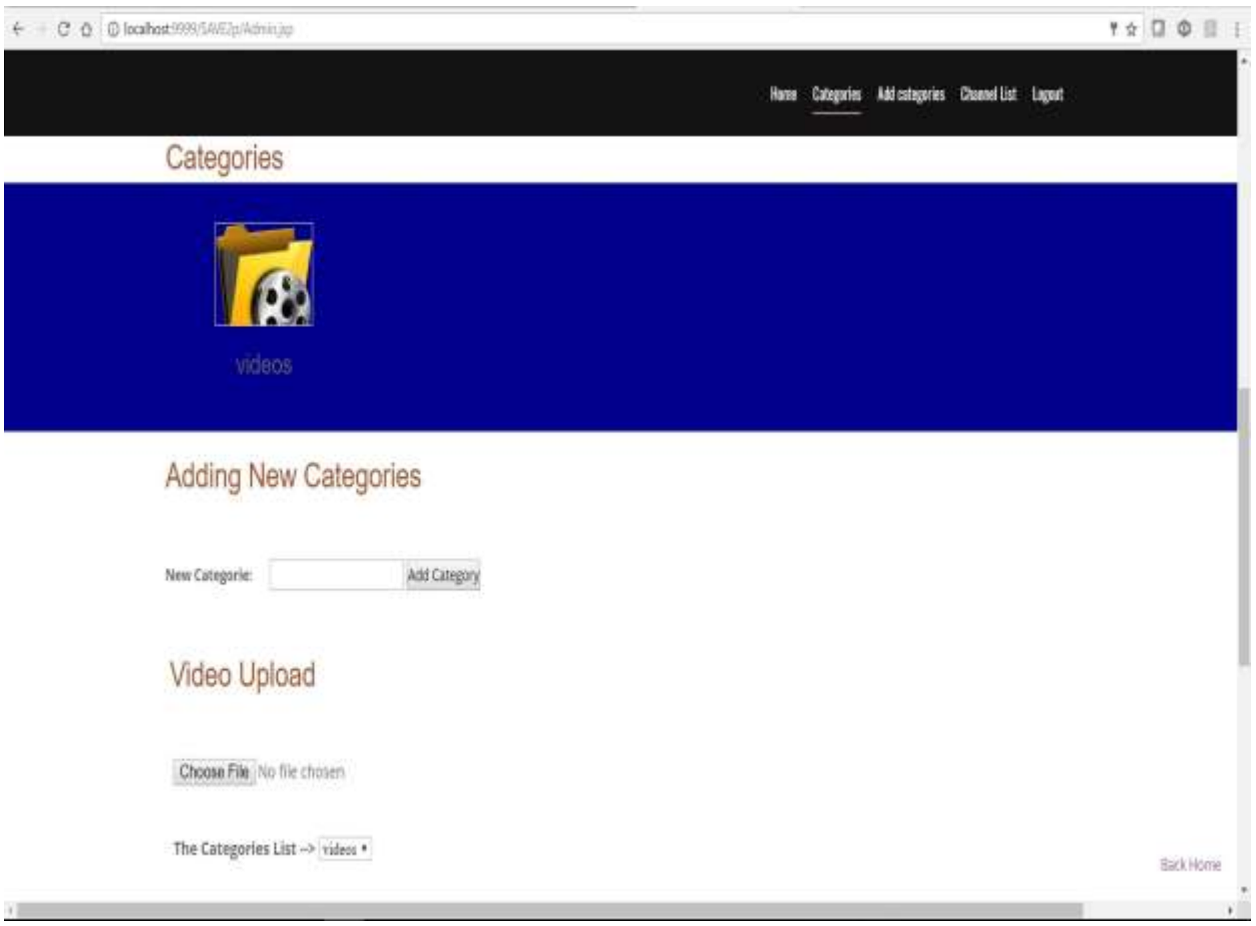


Fig 6.5 Screenshot of Administrator Home Page for Video Upload

The uploaded videos and the created categories can be viewed in the server console for verification, as shown in Fig 6.6. The server console shows all the operations done on the web pages. The data and the entries are stored in the server end.

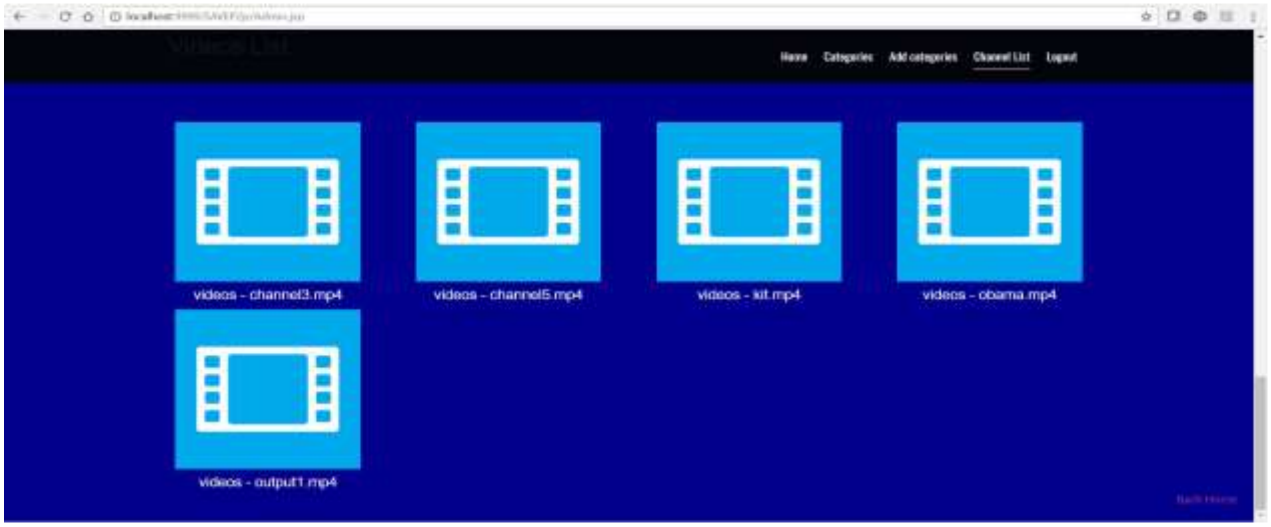


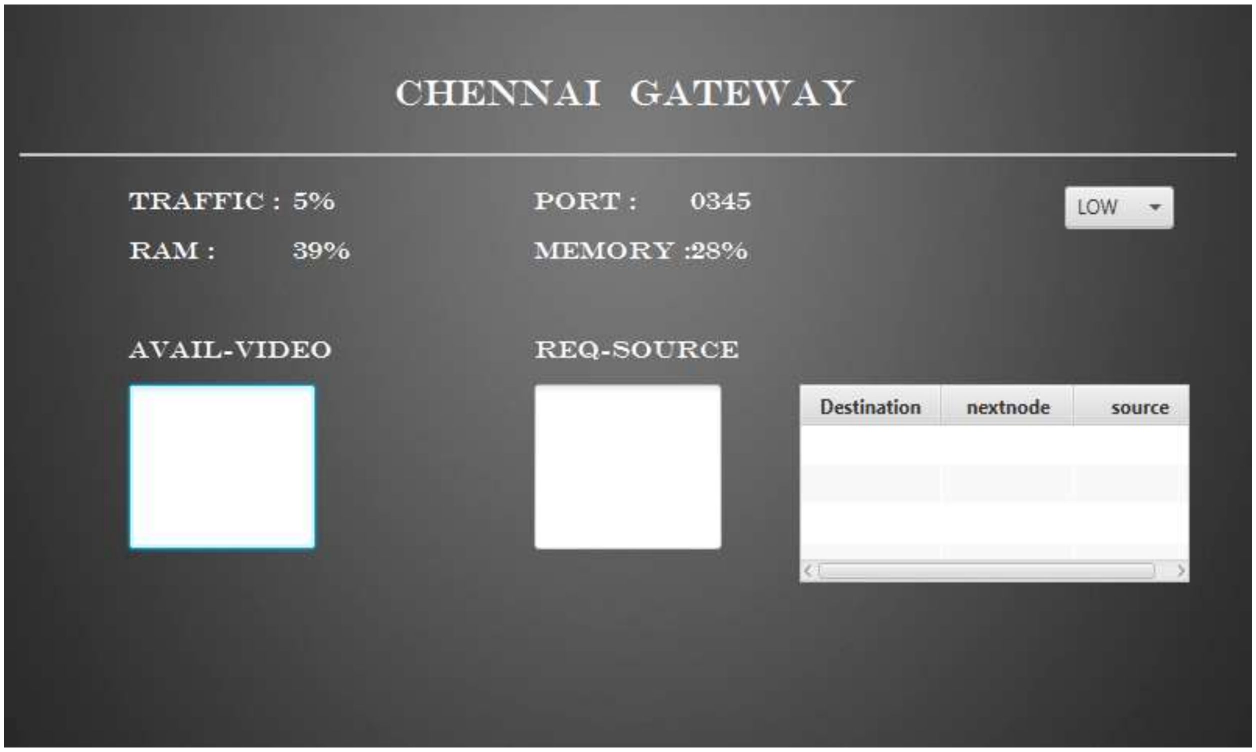
Fig 6.6 Screenshot of List of Uploaded Videos in the Server Webpage

6.3 USER PROCESS

Basically the program is executed in Netbeans IDE. The program is executed at first which invokes fxml controller file initially. The basic functionality of the fxml controller is to record user mouse patterns and executes the respective java files for creation of gateways, user console and intermediates, as shown in Fig 6.7.

Fig 6.7 Screenshot of FXML Controller Interface

The gateway option is selected for creation of gateways quadruple times based on different locations, as shown in Fig 6.8. (Say Chennai, Mumbai, Kolkata, Delhi) that has traffic recordings. The traffic level can be switched from low to high for performance study.

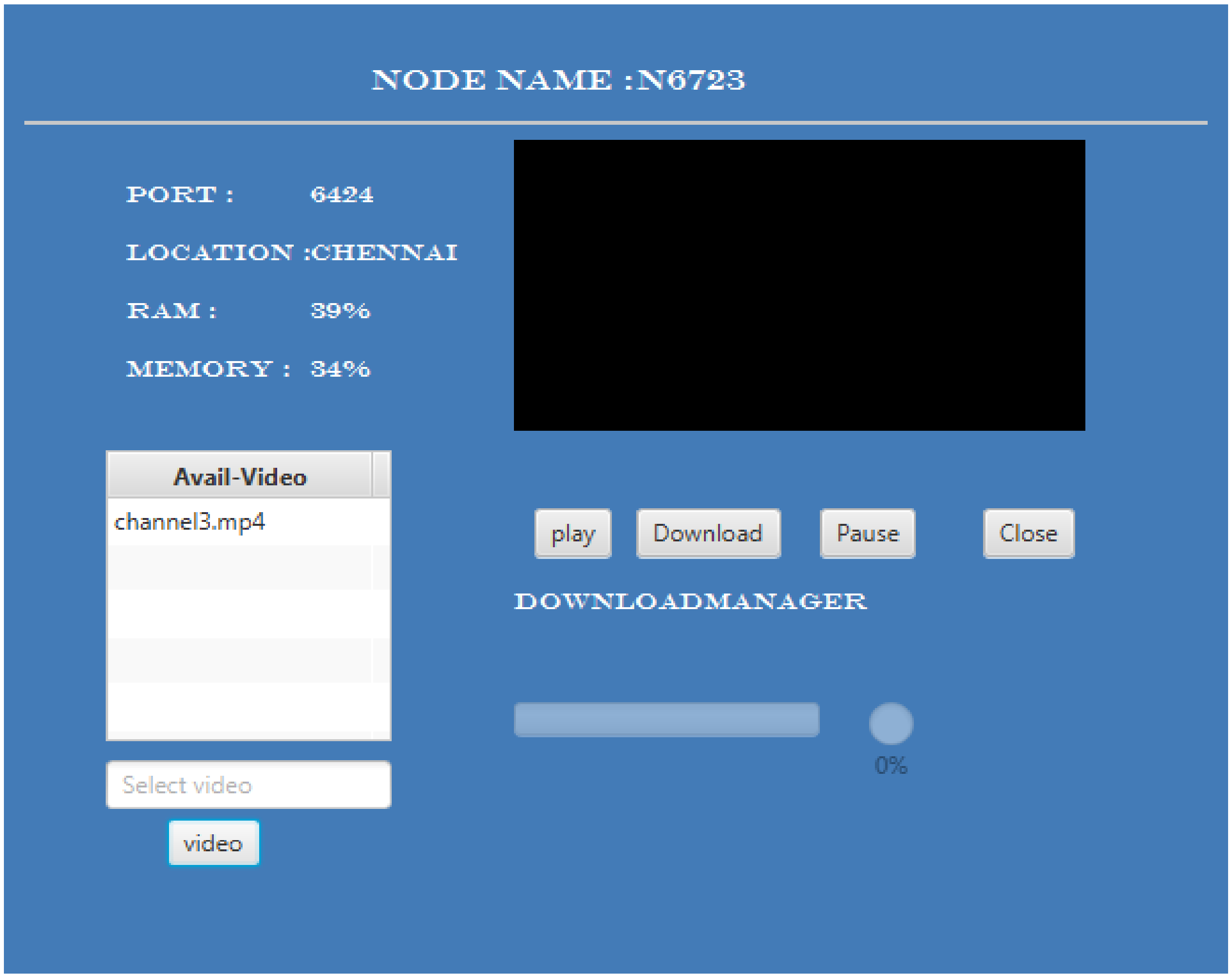

CHENNAI GATEWAY

TRAFFIC : 5%
RAM : 39%
PORT : 0345
MEMORY :28%

Destination	nextnode	source

Fig 6.8 Screenshot of Local Serving Gateway Interface

The user node is created twice for multiple user interaction experience, as shown in Fig 6.9. Both the user console displays the videos uploaded in the server. The node id and the location is generated in the user console. The video pane is present in the user console for the video to stream from the cache of the local gateways. The operations to play, pause and close the video is defined in buttons. The user needs to select the video in the available list and to play it.


NODE NAME :N6723

PORT : 6424

LOCATION :CHENNAI

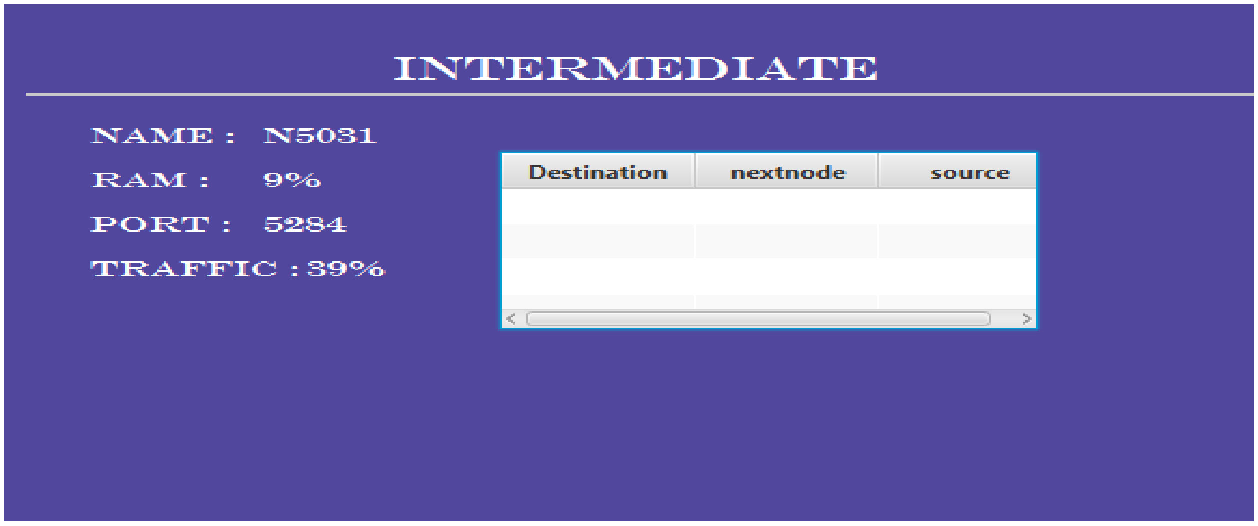
RAM : 39%

MEMORY : 34%

Avail-Video
channel3.mp4

Fig 6.9 Screenshot of User Interface to Request and Stream Video

Intermediate node is created for forwarding packets from one node to the other, as shown in Fig 6.10. This maintains the flow splitting table to route the video request or contents. The node id is displayed in the flow splitting table as source, destination and next node. This uses hop by hop forwarding mechanism to determine the adjacent or neighbor nodes. The user needs to request video that are available on the server. The user can play and pause the video from the server. The user can download the video from the server.



INTERMEDIATE

NAME : N5031

RAM : 9%

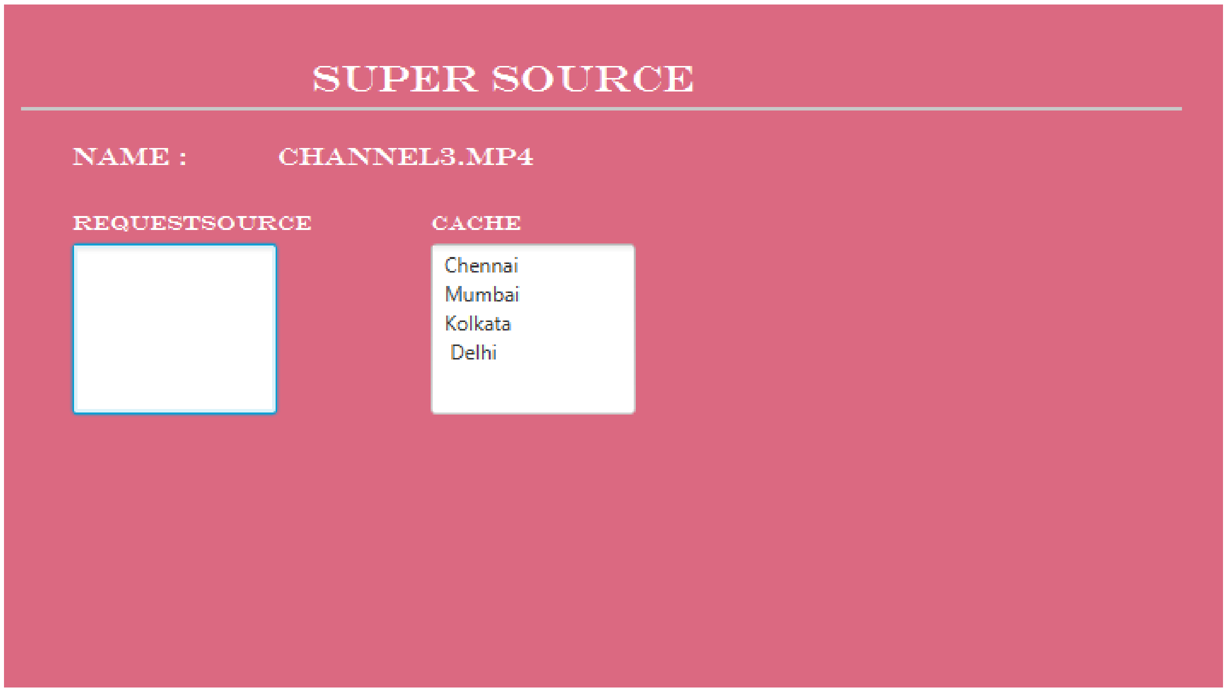
PORT : 5284

TRAFFIC : 39%

Destination	nextnode	source

Fig 6.10 Screenshot of Intermediate Node Interface for Forwarding Data

When the user requests the video from the user console the super source is created automatically for each video file, as shown in Fig 6.11. It has the information about the availability of video contents in the local gateway cache.



SUPER SOURCE

NAME : CHANNEL3.MP4

REQUESTSOURCE

CACHE

Chennai
Mumbai
Kolkata
Delhi

Fig 6.11 Screenshot of Super Source Point for Gateway Selection

6.4 EXAMINATION OF RESULTS

The user requests for the particular video to be streamed in the user console by selecting and playing the video, as shown in Fig 6.12. Depending on the traffic intensity, the request is served by the appropriate local gateway with respect to the availability of video contents in the local gateway cache. When another user requests for the same video, super source point is created for the particular video and using the reference it is redirected to the other local gateway where the video is available. This minimizes the maximum link utilization.

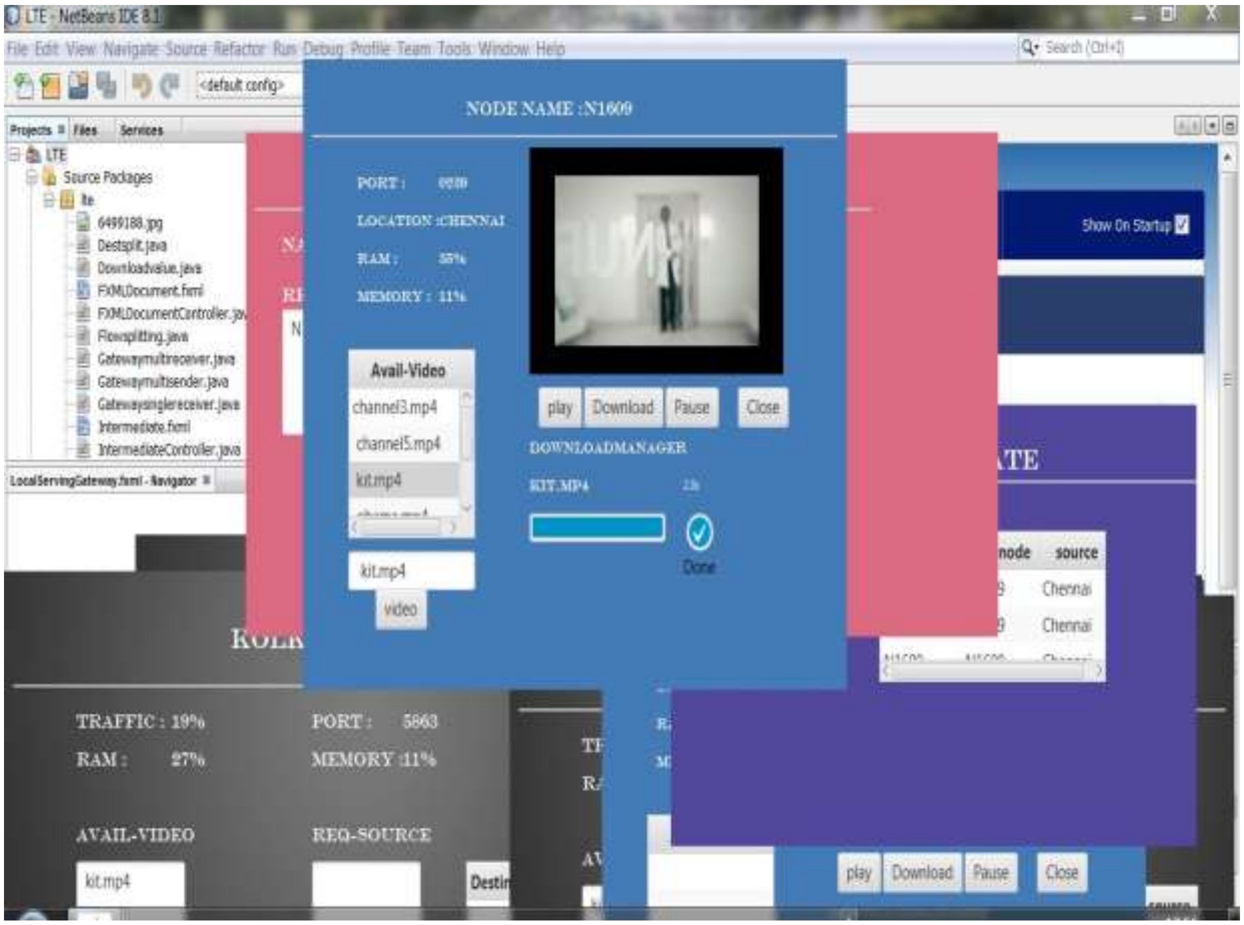


Fig 6.12 Screenshot of Video Content Streaming from Local Gateway Cache

The results are examined by switching the traffic intensity of a particular gateway for efficient routing. The user requests are redirected to the other local gateway with the help of super source point to minimize the cost path. The results are studied with the help of alert messages

CHAPTER 7

CONCLUSION

The main objective to efficiently route the video request by considering traffic and link utilization is implemented successfully with the help of Super source point. Fast algorithms are implemented and the gateway selection and traffic engineering problems are solved. The super source point helps in selecting the gateway and the requests are routed efficiently. Furthermore, hop by hop routing protocol is implemented and the network flow is improved and it has negligible overhead.

7.1 PERFORMANCE STUDY

The performance of the system has been studied by various techniques. This expresses the efficiency and reliability of the system that is designed. The performance of the system in any environment is effective and liable.

The performance is evaluated by switching the traffic intensities of all the local gateways as high. Then the routing of request is studied for limiting maximum link utilization with the alert messages, flow splitting table in the Intermediate node and the information in Super source point.

The performance is also studied by creating three or more user nodes and requesting for the same video. The routing based on the minimization of cost path is noted. Streaming multiple videos from different users are tested and studied with maximum load.

7.2 FUTURE WORK

The existing system can be further enhanced by making the super source point centralized. It would act as a single reference point which would contain the information about the location of all the videos.

Fibre optics may be used for instant transmission, which reduces the time to route a video request to a large extent. Faster optimization algorithms which reduce up to 80% of maximum link utilization and more than 60% of the link cost can be implemented.

REFERENCES

1. Adhikari, V.K., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M.and Zhang, Z. (2012) "Unreeling NetFlix: Understanding and improving multi-CDN movie delivery," in Proceedings of IEEE INFOCOM, pp. 1620–1628.

2. He, J., Zhang, H., Zhao, B. and Rangarajan, S.(2013) "A collaborative framework for in-network video caching in mobile networks," in Proceedings of IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON).

3. He, J., Zhao, X. and Zhao, B. (2012) "A fast, simple and near-optimal content placement scheme for a large-scale VoD system," in Proceedings of IEEE International Conference on Communication Systems (ICCS) pp. 378–382.

4. He, J. and Zhao, B. (2013) "A hybrid video caching framework in mobile core networks with CDN source suppliers," Univ. of Sci. and Tech. of China.

5. Jin, X., Li, L.E., Vanbever, L. and Rexford, J. (2013) "SoftCell: scalable and flexible cellular core network architecture," in Proceedings of the ninth ACM conference on Emerging networking experiments and technologies. ACM,pp. 163–174.

6. Lee, K., Zhang, H., Shao, Z., Chen, M., Parekh, A. and Ramchandran, K. (2012) “An optimized distributed video-on demand streaming system: Theory and design,” in Proceedings of IEEE Annual Allerton Conference on Communication, Control, and Computing. pp. 1347–1354.

7. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J. (2008) “OpenFlow: Enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74.

8. Roess, R.P., Prassas, E.S. and McShane, W.R.(2004) “Traffic engineering” Prentice Hall.

9. Xu, Y., Elayoubi, S.E., Altman, E. and El-Azouzi, R. (2013) “Impact of flow-level dynamics on QoE of video streaming in wireless networks,” in Proceedings of IEEE INFOCOM.

10. Zhang, Z., Zhang, M., Greenberg, A.G., Hu,Y.C., Mahajan, R. and Christian, B.(2010) “Optimizing cost and performance in online service provider networks.” in Proceedings of USENIX NSDI,pp 33–48.