Hypercube Architecture for Resource Management in a Video-on-Demand System

D. N. Sujatha*, Girish K*, Rajesh Srivastava*, Venugopal K. R*, L. M. Patnaik**
*Department of Computer Science and Engineering
University Visvesvaraya College of Engineering, Bangalore University, Bangalore-560001, India.
**Microprocessor Applications Laboratory, Indian Institute of Science, Bangalore-560012, India.
suj_sat@yahoo.com

***Abstract*— A challenging problem faced by researchers of distributed real-time systems like Video-on-Demand (VoD), is devising and implementing adaptive resource management strategies. The traditional resource management model such as hierarchical model does not address the ever-growing resource requirement of a VoD system. This paper examines the hypercube approach and designs a scalable architecture which addresses the heterogeneous requirements of the clients. This work also attempts to reduce the search space as each node in the hypercube has only $\lceil \log N \rceil$ neighbors where N is the total number of nodes. However, for simulation studies we have considered hypercube with *N=8*. The analytical and simulation results confirm that HARM : Hypercube Architecture for Resource Management is scalable as storage and delivery capacity grows with the number of clients in the system and is cost effective as the videos are not replicated. HARM increases system performance with less rejection ratio and the bandwidth utilization is 85 % compared to hierarchical scheme. The result indicate that the resources are appropriately allocated within the network.**

Index Terms: Bandwidth, Cluster, Hypercube, Resource Management, Video-on-Demand.

I. INTRODUCTION

In this era of digital information the emergence of the Internet as a pervasive communication media has enabled the convergence of voice, video and data. This convergence has enabled a wide spectrum of multimedia applications including distance learning, entertainment and information services to be incessantly used on the Internet. These multimedia applications require a sizable quantity of multimedia data to be moved to and fro. Multimedia data contains an enormous amount of embedded information. They have the potential to convey an infinite amount of derived or associated information.

Compared to traditional data multimedia data incorporates two distinct characteristics: *delay sensitivity* and *network bandwidth*. Delay Sensitivity is the timing constraint associated with multimedia data. Timing in delivering multimedia data is of the essence. The delayed arrival of the data may be of little or no use. Delivering multimedia data to users consumes large amount of *network bandwidth* as multimedia systems need to transfer enormous amount of data within a short span of time thus, requiring a large quantity of bandwidth from the underlying network.

The Video-on-Demand (VoD) system takes the basic television concept of providing instant entertainment to a level that can cater to the individual selection of the viewer. The VoD system either streams the content, enabling viewing while the video is being downloaded, or download the program entirely to a set-top box before the viewing starts. VoD systems operating in cable networks will see significant change in usage patterns and demand over the next five years. The peak usage of VoD system is likely to increase from the current 5% to approximately 30% as a result of larger deployments and the introduction of new and innovative applications [1]. Such a system maintains large quantity of resources needed to be harnessed to develop a reliable architecture.

A resource can be defined as an available supply that can be drawn from when it is needed. The parameters to be addressed in resource management in a VoD system are content management, delay, reliability, operating costs, bandwidth, buffer and storage capacity. Resource management include efficient and effective managing, handling, supervising and controlling of these resources. Content management deals with managing the content (video) that is streamed to the client from the video server. The content which is streamed should be delivered with the highest quality that can be achieved. The content should be reliable in the sense that the quality of the video should not vary while streaming. To achieve this, a sufficient bandwidth should be allotted by the video server to each client or for each request made. The bandwidth for a video server is fixed and therefore an efficient mechanism is made to utilize the entire bandwidth effectively.

To handle the incoming stream the client should have a sufficient buffer to store the streaming content as they are being processed. The buffer should be utilized in a manner that would maintain a balance that would not effect the speed of the video server and the processing ability of the client. The streaming cannot be faster than the processing ability of the client as number of frames would be lost. The speed should not be too slow as this would create a jitter that would affect the client viewing the video. Therefore the buffer should maintain this balance efficiently.

Resource management is a process of guaranteeing the availability of resources for the admitted requests in the VoD system. Distributed continuous media application are expected to provide service to a large number of clients often geographically dispersed. The challenging task of these type of application is how efficiently resource allocation are

implemented in a cost effective manner. The bandwidth is arguably the most important resource in a video server and therefore the most expensive resource. The bandwidth is inexplicably linked to all other resources and therefore should be handled delicately. To address this problem researchers have focused on various techniques to reduce the overall bandwidth requirements.

A. Our Contribution

The enlarging market in networking and multimedia technology is focused on the development of distributed multimedia applications with suitable resource management schemes. In this paper, we propose a hypercube architecture in order to optimize resources requirements and to efficiently allocate the resources. The hypercube architecture enhances scalability and robustness of the VoD system. The advantage of the proposed scheme is that the clients experience less delay when compared to hierarchical scheme which attracts more number of clients thereby increasing the profit of the VoD system. The algorithm ensures maximum system throughput with 85% bandwidth utilization.

B. Organization

The reminder of this paper is organized as follows. Section II presents literature survey. The model for providing efficient resource management is discussed in Section III. The hypercube architecture for resource management is explained in Section IV. The analytical approach along with cost calculation for hypercube is discussed in section Section V. The algorithm along with the illustrated example is discussed in Section VI. We elucidate the usefulness of our algorithm through simulation in Section VII. Finally, we provide concluding remarks in Section VIII.

II. LITERATURE SURVEY

In [2] a new architecture for server-side communication which supports the concept of Quality-of-Service (QoS) contracts is suggested. A QoS contract which specifies an acceptable QoS levels along with their utility function was proposed and the results were compared with simple reservation-based solutions. In [3] a overview of the recent developments in the area of QoS support for multimedia applications is explored. Time-dependent multimedia data types are saved with various formats and the support of operating system in system resource management, scheduling and adaptation as per real time requirements is studied. This paper only concentrates on OS issues overview but does not do a detailed survey.

In [4] the benefits of chaining is discussed and an optimal chaining scheme that utilizes not only the backward and forward buffers, but also all the available client buffer in a collaborative network is analyzed. This method reduces the server load and avoids the network bottleneck at the server. However, this method does not discuss the QoS issue while video streaming. Determining the proper amount of resources to be allocated is crucial for optimizing the performance of VoD systems, so as to maintain the benefits of data sharing technique.

In [5] segmentation and multicast techniques for VoD system is evaluated and their impact on performance of the system is studied. In this method multicast technique is adapted assuming all the videos have equal popularity which is not possible in real time and and does take unpopular video into account. Orallo et al. [6] deals with transmission of VoD service over the Internet. The scheme introduce a fast method to optimize network resources which is based on generating envelope points from empirical envelope. This process is done off-line using the stored video. This method is not checked for larger real-time application. Shahabi et al. [7] deals with distributed resource management for decentralized media server and minimizes the communication storage cost. To achieve this objective, Redundant Hierarchy (RedHi) topology was proposed with the following goals: (i) to reduce start-up latency and reduce resource management overhead.

Yu et al. [8] addresses the resource management problem in VoD system. In order to manage resources efficiently allocation, video server selection, replication, cache management are considered by making use of video title characteristics and to efficiently utilize system resource and minimize waiting time. However, this scheme does not address admission control. Raman et al. [9] proposed a scheme developed and implemented the classified (classads) matchmaking framework for resource management in a distributed environment with decentralized ownership of resources. The framework include a semi-structured data model that combines schema, data and query in a simple but specification language and a separation of matching and claiming phases of resource allocation. The technique of aggregating classadds so that matches can be performed in groups is not considered.

Cahill et al. [10] examines issues of resource management and content placement within a Video Content Distribution Network. A placement heuristic is proposed which reduces the search space. The placement algorithm uses cost model to determine best proxy for each cluster and the proxy is the potential location for replica requested by the cluster. Golubchik et al. [11] have considered as K-server threshold-based queuing system with hysteresis and is governed by forward and reverse threshold vector. The use of hysteresis was to control the cost during workload fluctuation. The time required to add server is negligible which needs to be calculated accurately for many real time applications.

Leung et al. [12] propose a combination of data sharing techniques (batching with static partitioning) which result in efficient resource management and system sizing for a large scale VoD system. The problem of achieving fairness in the allocation of the bandwidth when a link is shared by multiple flows of traffic has been extensively study in [13] and arrive at a solution defining fairness. This method assumes that each flow has a unique weight which determines its relative rightful share of each of the resources.

III. MODEL

A. BACKGROUND

n-cube	Graph	Names	Vertices 0-Faces	Edges 1-Faces	Faces 2-Faces	Cells 3-Faces
0-cube		Point	1			
1-cube		Digon {} OR{2}	2	1		
2-cube		SQUARE Tetragon {4}	4	4	1	
3-cube		CUBE Hexahedron {4.3}	8	12	6	1

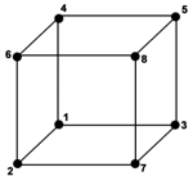
Fig. 1. Elements of Hypercube

A hypercube of dimension *n* has *2n* sides. The number of vertices (points) of a hypercube is 2^n (a cube has 2^3 vertices, for instance). The number of m-dimensional hypercubes on the boundary of an n-cube is $2^{n-m}\binom{n}{m}$. The advantages of hypercube is that each node has only $\lceil \log N \rceil$ neighbors and also the longest route in the hypercube is $\lceil \log N \rceil$. However Hypercube construction must be done sequentially i.e. one node at a time and also physical network structure is completely ignored.

DEFINITION : The logical Hypercube overlay network topology organizes the applications with a logical n-dimensional Hypercube. Each node is identified by the label (e.g. **010**) which indicates the position of the node in a logical Hypercube.

Theorem 1. *Each and every node in a Hypercube has only* $\lceil \log N \rceil$ *neighbors where N is the total number of nodes.*

Proof.



Let us consider a Hypercube with 8 nodes (i.e. Hexahedron *N=8*) and each node is numbered from node 1 to node 8. Now, let us take node 1 its adjacent nodes are node 2, node 3 and node 4 respectively i.e. 3 nodes. Similarly consider node 8, its adjacent nodes are node 5, node 6 and node 7 respectively i.e. 3 nodes. In the same manner if we consider node *N*, then its adjacent nodes are *N-1*, *N-2* and *N-3* respectively i.e. 3 nodes.

= 3

= 3 ∗ 1

$= 3 \log_2 2 \qquad [\log_2 2 = 1]$

$=\log_2 2^3$

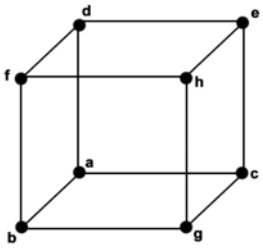
$= \log_2 8$

$= \log_2 N$ [From assumption N=8].

This proves that each and every node in a Hypercube has only $\lceil \log N \rceil$ neighbors.

Theorem 2. *The longest route in the Hypercube is* $\lceil \log N \rceil$ where N is the total number of nodes.

Proof.



Let us consider a Hypercube with 8 nodes (i.e. Hexahedron *N=8*) and each node is numbered from node *a* to node *h*. Consider all the paths from node a to node h of all possible length.
Path of length 1 are : $a \longrightarrow b, a \longrightarrow c, a \longrightarrow d$.

Path of length 2 are :

$a \longrightarrow c \longrightarrow e, \quad a \longrightarrow d \longrightarrow e, \quad a \longrightarrow e \longrightarrow g,$
$a \longrightarrow b \longrightarrow g, \quad a \longrightarrow d \longrightarrow f, \quad a \longrightarrow b \longrightarrow f.$

Path of length 3 are :

$a \longrightarrow c \longrightarrow g \longrightarrow h, \quad a \longrightarrow c \longrightarrow e \longrightarrow h,$
$a \longrightarrow d \longrightarrow e \longrightarrow h, \quad a \longrightarrow b \longrightarrow f \longrightarrow h,$
$a \longrightarrow b \longrightarrow g \longrightarrow h.$

This shows that maximum path length is 3.

= 3

= 3 ∗ 1

$= 3 \log_2 2 \qquad [\log_2 2 = 1]$

$= \log_2 2^3$

$= \log_2 8$

$= \log_2 N$ [From assumption N=8].

This proves that longest route is $\log N$.

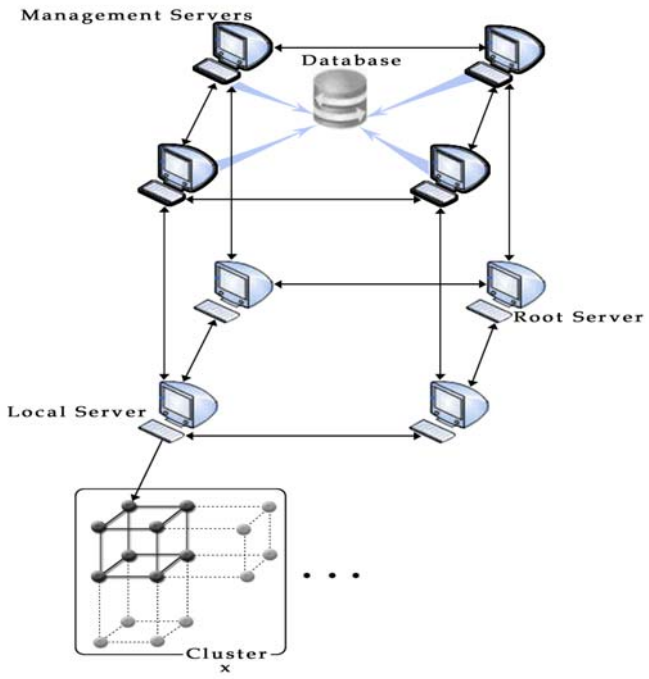


Fig. 2. Overview of Hypercube Architecture.

IV. ARCHITECTURE

A. PROBLEM FORMULATION

Given a physical network HC = (V, L) consisting of a finite set of nodes $V = \{v_0, v_1, v_2, \ldots, v_{N-1}\}$ and a finite set of links $L \subseteq \{(n_i, n_j)/n_i, n_j \in n \wedge n_i \neq n_j\}$. The objectives of resource management is to guarantee steady flow of continuous media data for each session, minimize the system overhead and maximize the total number of concurrent streams. It also eliminates client side storage, efficiently manages system resources such as storage and bandwidth.
Fig. 2, depict Hypercube Architecture for resource management in VoD system. It consists of a *database, management server, root server* and *cluster*.

Database is an information repository that maintains state of the network such as the list of management servers, root server and clients and the location of video in the architecture and all the meta data that is associated with the video. The meta data will be used to facilitate client searching and browsing of archived videos. It also stores all the complete videos which need to be streamed to clients on request.

Management Server is used to monitor the entire network. The functions of management server include collecting statistical data to check video popularity, accounting details help in billing the videos watched. It also helps in performing content control. In the proposed hypercube architecture the management server is used to grant or revoke access to content.

Root Server is used to provide the requested videos to the client which are in the Hypercube cluster. The root server which is directly attached with the cluster is known as local server. Initially, the requested video is searched in the local server if it is found it is streamed to the client else if it is not found it will send request to neighboring root server and continues till the requested video is found.

Cluster is a group of clients who are situated in close proximity within a specific network. The usage of clusters decreases content placement decisions. For an average sized cluster optimal placement of videos for the cluster is optimal for all the clients within the cluster. Here, we are arranging all the clients in the form of Hypercube so that, latter the Hypercube can grow dynamically which enhance the scalability of the architecture.

V. ANALYTICAL APPROACH

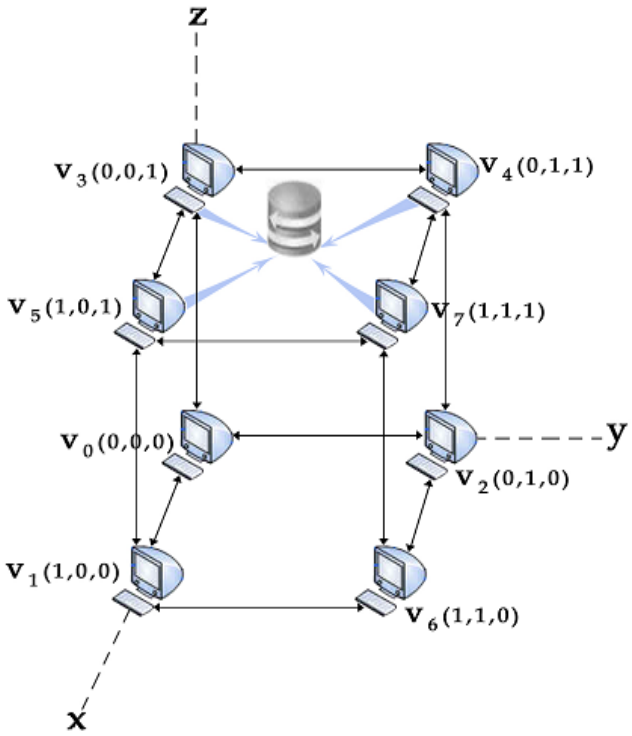


Fig. 3. Coordinate Representation of Hypercube Architecture.

Fig. 3 depicts the coordinate representation of Hypercube architecture. Let us consider two position vectors $\vec{V_s}$: ($\widehat{i}_s$, $\widehat{j}_s$, $\widehat{k}_s$), $\vec{V_d}$: ($\widehat{i}_d$, $\widehat{j}_d$, $\widehat{k}_d$) and the magnitude or resultant of the length of the path between the two position vectors $\vec{V_s}$ and $\vec{V_d}$ can be calculated as:

$$\left|\vec{V_s}\vec{V_d}\right| = (\widehat{i}_s + \widehat{i}_d, \quad \widehat{j}_s + \widehat{j}_d, \quad \widehat{k}_s + \widehat{k}_d) \tag{1}$$

TABLE I
MODULUS TABLE $(+_2)$.

$+_2$	0	1
0	0	1
1	1	0

(1) is obtained by using $(Z_2, +_2)$ operation (Refer Table 1.)

A. State Transition Diagram For Hypercube Architecture

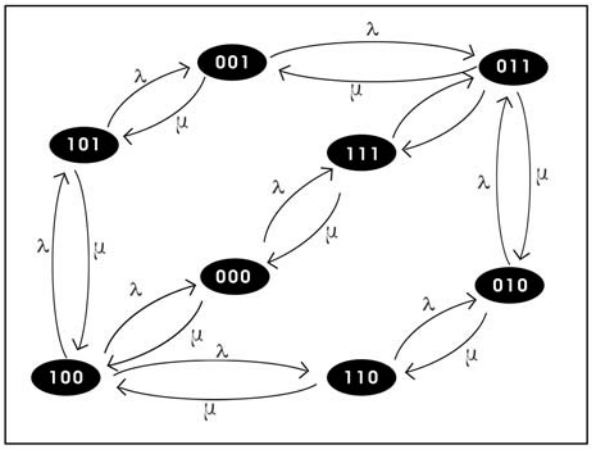


Fig. 4. State Transition Diagram for N=8.

In this subsection, we describe our model which is illustrated in Fig. 2. There are *N* servers with *N/2 management servers* and *N/2 root servers* in the system, where *N* is unrestricted, each with service rate μ and the client request for video process is with rate λ. Formally the transition structure of *N* homogeneous servers, where *N* is unrestricted is specified as follows:

(0,0,0) → (1,1,1)

$$\lambda$$

$(\widehat{i}_s, \widehat{j}_s, \widehat{k}_s) \rightarrow (\widehat{i}_d, \widehat{j}_d, \widehat{k}_d)$

$$\lambda f\left\{\widehat{i}_s + \widehat{i}_d, \widehat{j}_s + \widehat{j}_d, \widehat{k}_s + \widehat{k}_d\right\} \text{ [See Table 1]}$$

$(\widehat{i}_d, \widehat{j}_d, \widehat{k}_d) \rightarrow (\widehat{i}_d, \widehat{j}_d, \widehat{k}_d)$

$$\lambda f\left\{\widehat{i}_d + \widehat{i}_d, \widehat{j}_d + \widehat{j}_s, \widehat{k}_s, +\widehat{k}_s\right\} \text{ [See Table 1]}$$

(1,1,1) → (0,0,0)

$$\mu$$

where $f\{x\}$ which is a function of x which calculates the length of path from source node V_s to destination node V_d. The state transition diagram for N=8 is shown in Fig. 4.

B. Path Matrix Geometric Approach

The path matrix P represents all the values which represents path from one node to another node. The element of path matrix $P[i,j]$ can be expressed mathematically as

$$P[i,j] = \begin{cases} 0 & \text{when i = j} \\ \mathrm{l}_{i,j} & \text{when i} \neq j. \end{cases}$$

where $0 \leq i \leq N-1$ and $0 \leq j \leq N-1$ and $l_{i,j}$ value can be calculated by using (1). The length of path from

$$P = \begin{bmatrix} 0 & l_{0,1} & l_{0,2} & l_{0,3} & l_{0,4} & l_{0,5} & - & - \\ l_{1,0} & 0 & l_{1,2} & l_{1,3} & l_{1,4} & l_{1,5} & - & - \\ l_{2,0} & l_{2,1} & 0 & l_{2,3} & l_{2,4} & l_{2,5} & - & - \\ l_{3,0} & l_{3,1} & l_{3,2} & 0 & l_{3,4} & l_{3,5} & - & - \\ l_{4,0} & l_{4,1} & l_{4,2} & l_{4,3} & 0 & l_{4,5} & - & - \\ l_{5,0} & l_{5,1} & l_{5,2} & l_{5,3} & l_{5,4} & 0 & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \end{bmatrix}$$

Fig. 5. Path Matrix

node $\vec{V_i}$ to $\vec{V_j}$ is given in Fig. Path matrix.
$l_{0,1}$ represents transition rates from l_0 to states l_1. After calculating all the values of $l_{i,j}$ and matrix representation of Hypercube is shown in Fig.6.

$$P = \begin{array}{c} \\ V_0 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \end{array} \begin{array}{c} \begin{array}{cccccccc} V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 \end{array} \\ \begin{bmatrix} 0 & a & a & a & 2a & 2a & 2a & 3a \\ a & 0 & 2a & 2a & 3a & a & a & 2a \\ a & 2a & 0 & 2a & a & 3a & a & 2a \\ a & 2a & 2a & 0 & a & a & 3a & 2a \\ 2a & 3a & a & a & 0 & 2a & 2a & a \\ 2a & a & 3a & a & 2a & 0 & 2a & a \\ 2a & a & a & 3a & 2a & 2a & 0 & a \\ 3a & 2a & 2a & 2a & a & a & a & 0 \end{bmatrix} \end{array}$$

Fig. 6. Matrix Representation of Hypercube

C. Cost Function for Hypercube

The storage space and link efficiency are the two important resources which influences high-quality video over shared

TABLE II

THE PARAMETERS USED IN COST FUNCTION CALCULATION.

Sl. No.	Symbol	Definition
1.	L_s	Local server.
2.	S(m)	Video title of size m bytes.
3.	B_c	Remaining part of video (in bytes) to be delivered to cluster *c*.
4.	$\alpha_{L_s,c}$	Bulk delivery cost is the cost associated with bulk delivery of *i* bytes between the local server L_s and the cluster *c*.
5.	$\beta_{L_s,c}$	Stream delivery cost is the cost associated with the streaming *i* bytes on the link between local server L_s and a cluster *c*.
6.	ξ_{L_s}	Storage cost is the cost for storing *i* bytes between the local server L_s and the cluster *c*.
7.	T_c	The time when the last client within the cluster will reach the end of video stream.

infrastructure. These days the clients request for high quality video with optimal cost. Hence video placements in the network has an immense impact on the required resources to deliver the objects with optimum cost. A cost function is proposed which calculates the resource requirements for delivering content from any server to the cluster. The Parameters used in cost function calculation is shown in Table II.

The cost of delivering an object from any server to cluster is comprised of following costs:
(i) $Storage \quad cost - SC_{L_s,m}$ is the cost of storing the video *m* on local server L_s and is a function of time. It is represented by (2).
(ii) $Delivery \quad cost - DC_{L_s,c}$ is the cost of streaming contents from L_s to the clusters and is shown in (3).
The total cost of serving all the clients in the cluster is given by (4).

The cost of using local server L_s to *N* clients in a cluster, all viewing the video *m* is given by the following equations.

$$SC_{L_s,m,c} = \xi_{L_s} * \mathrm{S(m)} * T_c \tag{2}$$

$$DC_{L_s,c} = \sum_{c=1}^{N} (\alpha_{L_s,c} * \beta_{L_s,c} * B_c) \tag{3}$$

$$Cost_{L_s,m,c} = SC_{L_s,m,c} + DC_{L_s,c} \tag{4}$$

VI. ALGORITHM.

The HARM algorithm (Refer Table III) is called when the client requests for a video. The request for the video is routed to the root server N. The root server would initiate a search for that particular video in its information repository. If the search is successful, the root server would then check its buffer for sufficient space. If the buffer has sufficient space, then move the video block by block to the buffer and then stream it to the client. If sufficient space in the buffer is not available then the Video Replacement algorithm (Refer Table V) is called. If the search is not successful then the Generate Request algorithm (Refer Table IV) is called.

TABLE III

HARM MAIN ROUTINE

```
Found = false
Request(Video,Node)
begin
found=search(VideoId, Node)
if(found)
   if(buffer not full)
      move the video block by block to buffer and stream the
      video block to client
   else
   VideoPlacement(VideoId, Node)
else
   GenerateRequest(VideoId)
end
```

The Generate Request algorithm is called when the video i.e. requested by the client is not found in the root server. The algorithm would initiate a search for the video in the rest of the video servers. The search for the video starts from the *log N* neighbors. If the requested video is not found, then it will increment the distance by 1 and initiate a new search for the video. When the requested video is found in the root server it will be moved to the buffer of the requested node and then streamed to client. If video is not found amongst the peers, then the request is forwarded to the root node of the cluster.

Video Placement algorithm examines the storage space available to place the video. If storage space is available, it stores the video at the respective node otherwise it will search

TABLE IV

GENERATE REQUEST ROUTINE:

```
GenerateRequest(VideoId)
begin
for( distance 0 to n*a)
begin
for(log N neighbors)
begin
   search video in node N
   if video is found
     found = true
end
   if video found
   Move the video blocks to requesting video server
   Else
   Increment distance
end
   if video found
   stream video to client
   else
   GenerateRequest(RootNode)
end
```

TABLE V

VIDEO PLACEMENT ROUTINE:

```
VideoPlacement(VideoId,Node)
begin
   if(videoSize < storage space)
     store video with VideoId at Node
   else
   begin
     search least popular video
     Replace least popular video with VideoId at Node
   end
end
```

TABLE VI

INSERT ROUTINE:

```
insertNode(N)
begin
   coord = v(000)

   while(Node < N)
   increment coord
   assign coord to node N
   for(i=0 to Node N)
   begin
     for(j=0 to Node N)
       begin
         sum = 0;
         for(each vertex k in coord)
       begin
         sumK = (coord of Node_i+coord of Node_j) % 2
         sum = sum + sumk
       end
         Node.distance[i][j] = sum;
       end
   end
end
```

the least popular video in the video array and then replaces that least popular video with the new video.

The search in our architecture requires to be done in constant time i.e. O(1). Ideally it may not be possible to achieve this but we can achieve a performance very close to it as we use a data structure called *Hash Table*. Searching using open hashing techniques requires less amount of time. An item can be searched using the hash address found by the hash function h as given by:

$$h(k) = k \bmod m$$

The key k is to be mapped into one of the m slots in the hash table. It is divided by m and the remainder is taken as index for the hash table. Since it requires only one operation hashing is fast.

The algorithm isertNode (Refer Table VI) inserts a node into the cluster and assigns coordinates to that node which is joined to the hypercube cluster. It also stores the distance between the new node and all the other nodes. The distance is found by calculating the path length from the new node to all the other nodes in the cluster to enable efficient search for

TABLE VII
TABLE INDICATING NUMBER OF VIDEOS AT EACH NODE.

Node-id	Number of Videos
0	500
1	800
2	600
3	900
4	1200
5	500
6	700
7	1500

video, by traversing a minimum distance.

A. Illustrated Example

Consider a cluster with 8 nodes indicating number of videos at each node is as given in Table VII. At any instant of time requests arrive at nodes or video server randomly. Considering arrival of requests $R = \{R_1, R_2, R_3, R_4, R_5\}$ at node-1. The requested video might be present in the same node or in other nodes in the cluster. Table VIII illustrates the video requested and the characteristics of the video and the node in which the corresponding video is located.

If the requested video is present in node-1, request is served immediately and the video frames are streamed to the client. If the video is not found in node-1 video search packets are forwarded to servers at distance-1 that are nodes 0, 5 and 6. If the video is found the video is streamed from the node in which the video was found to node-1 and then streamed to the client requesting the video. If the video is not found at distance-1 the search process is continued with distance-2 to nodes 2, 3, 7 and then at distance-3 to node 4. The reneging time is considered to be 8 ms. Request is rejected if it is not serviced within this time. Table IX illustrates the search process, bandwidth consumption and delay incurred for requests.

VII. SIMULATION AND PERFORMANCE EVALUATION

A number of experiments were carried out using OMNeT++ in order to determine the performance of the proposed algorithm. Experiments were conducted using various sized Internet topologies. These topologies were generated using the BRITE topology generator. The topologies comprised of 20 to 100 servers (management/root). A small set videos of video length ranging from 30 minutes to 150 minutes was used. The requests for the particular video were distributed following the Zipf distribution with the parameter θ = 0.271.

Fig. 7, plots a graph of the requests serviced and the number

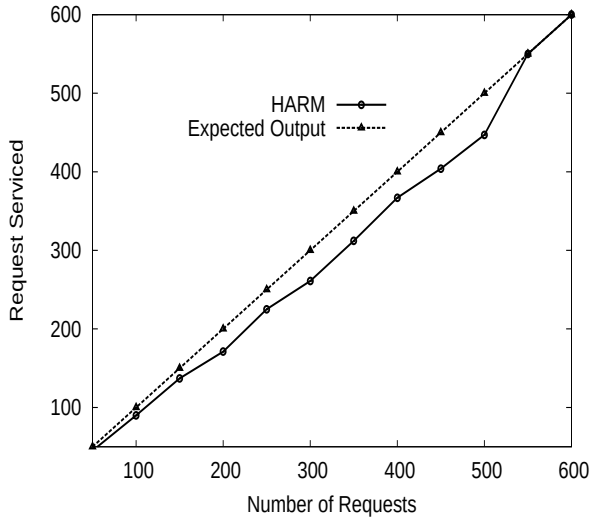


Fig. 7. Throughput

of requests. As depicted in the graph, throughput which is the number of requests serviced to the total number of requests increases with the increase in the number of requests. The graph depicting throughput of a realistic system should be a straight line i.e.*y = x* with gradient of 1 indicated as *Expected Output* in the Fig. 7. As seen in the graph, the difference between simulated value of throughput for hypercube architecture and expected value for throughput is almost the same.

The bandwidth utilization for HARM is indicated in Fig.

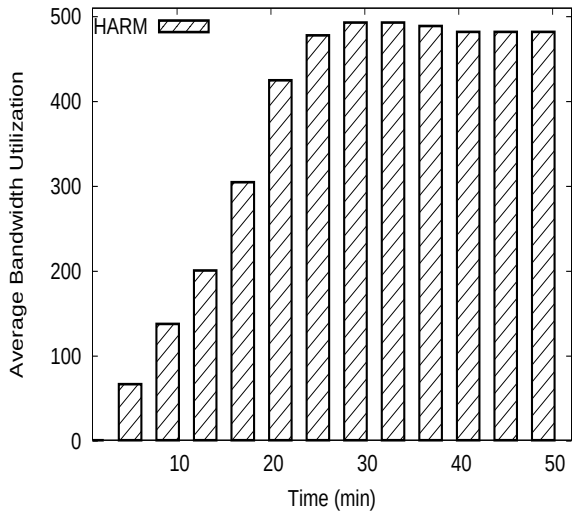


Fig. 8. Bandwidth Utilization

8. Up to 25 requests, bandwidth utilization keeps increasing and later it gets saturated. As our interest is in the area of high load HARM utilizes 95% of bandwidth under high load as shown in the graph.

The number of video requests/sec vs average delay (sec) is

TABLE VIII

SAMPLE VIDEOS FOR SIMULATION EXPERIMENT.

Req.id.	Node where Video is Present	Requested Video Title	Length(min)	Frame Rate/ms	BW(Mbps)	Video Size(MB)
R_1	1	Benhur	120	30	20	800
R_2	6	Demo	50	30	9	300
R_3	7	Blade2	100	30	17	650
R_4	1	Jurassic Park	150	30	24	900
R_5	2	Rambo	90	30	16	600

TABLE IX

TABLE DEPICTING ALLOTMENT OF BANDWIDTH AND CALCULATION OF DELAY.

Request-id	Video Search Process	Total Bandwidth Utilized (Mbps)	Delay (ms)
R_1	Video found at node-1	20 (node-1)	0
R_2	Video not found at node-1 Search packet to nodes 0,5,6 Video found at node-6	9+9=18 (node1+node6)	1
R_3	Video not found at node-1 Search packet to nodes 0,5,6 - Not found Search packets to nodes 2,3,7 Video found at node-7	17+17+17=51 (node1+node6+node7)	3
R_4	Video found at node-1	24 (node-1)	0
R_5	Video not found in node-1 Search packet to nodes 0,5,6 - Not found Search packets to nodes 2,3,7 Video found at node-2	16+16+16=48 (node1+node0+node2)	3

shown if Fig 9. It is clearly seen from the graph that HARM shows average delay of 5 seconds compared to the hierarchical model which has a average delay of 10 seconds. The average delay is reasonably low because HARM adopts the hypercube architecture which has $\lceil \log_N \rceil$ neighbors where *N* is the total number of nodes compared to hierarchical model which has $N-1$ neighbors. The search space is also reduced because the open hashing technique is adopted.

Fig. 10, depicts the rejection ratio for both HARM and the hierarchical scheme. As seen in the graph there is a wide gap in terms of rejection ratio in both the schemes. The rejection ratio is 2% which is negligible. In HARM most of the requests are serviced as in each of the server when the buffer is full, video replacement algorithm is used in order to replace least popular video with the requested video.

VIII. CONCLUDING REMARKS

The proposed Hypercube architecture has been simulated and tested with analytical methods. The results of the simulation indicate that a service provider using the hypercube architecture will be able to utilize the bandwidth up to 95% under heavy load. The characteristics of the scheme are as follows:

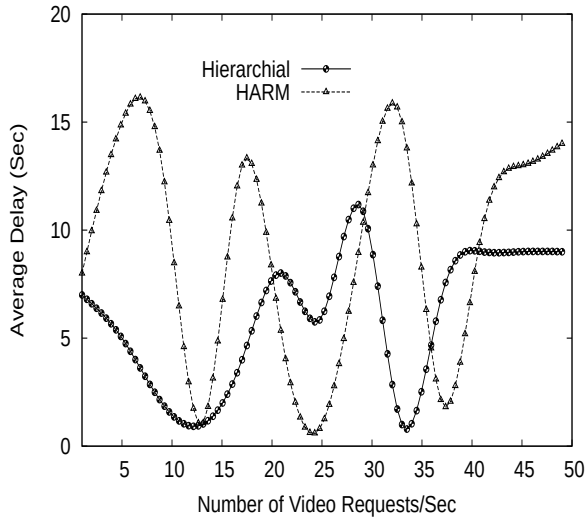


Fig. 9. Delay

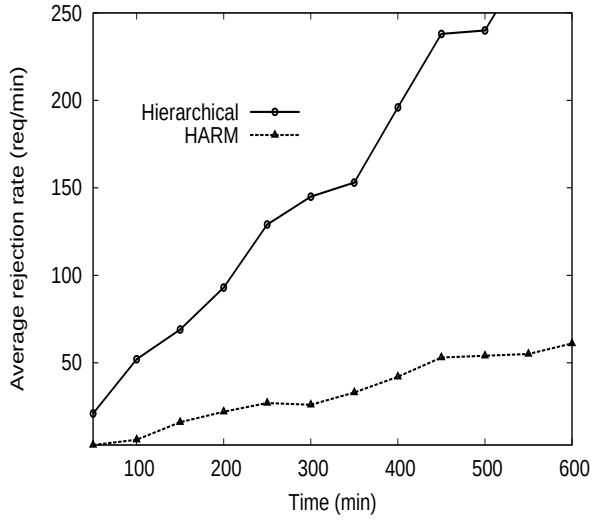


Fig. 10. Rejection Ratio

- to design a fully decentralized resource management system that guarantees scalability, robustness as well negligible delay of 5 seconds to the clients.
- maximize the total number of concurrent streams with high throughput.
- to provide improved system resource utilization - storage and bandwidth.
- eliminate the necessity of client side storage.

Our future work is to evaluate how the proposed architecture can be used in real time applications considering heterogeneous nature of the network.

REFERENCES

[1] A. Dan, D. Sitaram, P. Shahabuddin, Dynamic Batching Policies for an On-demand Video Server, ACM Multimedia Systems, vol 4, pp. 112-121, 1996.

[2] Tarek F Abdelzaher, Kang G Shin, Nina Bhatti, *"User-Level QoS-Adaptive Resource Management in Server End-System"*, IEEE Transaction on Computers, 2003, pp. 678-685.

[3] Thomas Plagemann, Vera Goebel, Pal Halvorsen,*"Operating System Support for Multimedia Systems"*, Computer Communications Journal-IDMS, 1998, pp. 1-26.

[4] Te Chou Su, Shih Yu Huang, Cheg Lung Chan, Jia Shung Wang ,*"Optimal Chaining Scheme for Video-on-Demand Applications on Collaborative Networks"*, IEEE Transaction on Multimedia, vol. 7, no. 5, 2005, pp. 972-980.

[5] Hari Kalva, Borko Furht, *"Techniques for Improving the Capacity of Video-on-Demand Systems"*, Intl. Conference on System Sciences, 1996, pp. 308-314.

[6] Enrique Hernandez Orallo, Joan Vila Carbo, *"A Fast Method to Optimize Network resources for Video-on-Demand Transmission"*, Euromicro, 2000, pp. 123-131.

[7] Cyrus Shahabi, Farnoush Banaei Kashani, *"Decentralized Resource Management for a Distributed Continuous Media Server"*, IEEE Transaction on Parallel and Distributed Systems, vol. 13, no. 6, 2002, pp. 1-18.

[8] Hongtao Yu, Chor Pinf Low, Yacine Atif,*"Design Issues on Video-on-Demand Resource Management"*, In the Proc. fo the 8th IEEE International conference on Networks, 2000, pp. 199-203.

[9] Rajesh Raman, Miron Livny, Marvin Solomon, *"Matchmaking: Distributed Resource Management for High Throughput Computing"*, Proc. of the Seventh IEEE International. Symposium on High Performance Distributed Computing, 1998, Chicago, pp 140-146.

[10] Adrain J Cahill, Cormac J Sreenam, *"An Efficient Resource Management System for a Streaming Media Distribution Network"*, Interactive Technology and Smart Education, 2006, pp. 31-44.

[11] Leena Golubchik, John C S Lui, *"Bounding of Performance Measures for Threshold-Based Queuing Systems : Theory and Application to Dynamic Resource Management in Video-on-Demand Servers"*, IEEE Transaction on Computers, vol. 51, no. 4, 2002, pp. 353-372.

[12] Mary Y. Y Leung, John C. S Liu, Leana Golubchik,*"Use of Analytical Performance Models for System Sizing and Resource Allocation in Interactive Video-on-Demand Systems Employing Data Sharing Techniques"*, IEEE Transaction on Knowledge and Data Engineering, vol. 14, no. 3, 2002, pp. 615-635.

[13] Yunkai Zhou, Harish Sethu,*"On Achieving Fairness in the Joint Allocation of Processing and Bandwidth Resources: Principles and Algorithms"*, IEEE/ACM Transaction on Networking, vol. 13, no. 5, 2005, pp. 1054-1067.