Prefix based Chaining Scheme for Streaming
Popular Videos using Proxy servers in VoD

M Dakshayini,
Research Scholar, Dr. MGR University. Working with
Dept. of ISE, BMSCE,
Member, Multimedia Research Group, Research Centre,
DSI,
Bangalore, India.

shantha_dakshu@yahoo.co.in
Dr T R GopalaKrishnan Nair
Director, Research and Industry Incubation Centre, DSI,
Bangalore, India
trgnair@yahoo.com

Abstract—Streaming high quality videos consumes significantly
large amount of network resources. In this context request-to-
service delay, network traffic, congestion and server
overloading are the main parameters to be considered in video
streaming over the internet that affect the quality of service
(QoS). In this paper, we propose an efficient architecture as a
cluster of proxy servers and clients that uses a peer-to-peer
(P2P) approach to cooperatively stream the video using
chaining technique. We consider the following two key issues
in the proposed architecture (1) Prefix caching technique to
accommodate more number of videos close to client (2)
Cooperative client and proxy chaining to achieve the network
efficiency. Our simulation results shows that the proposed
approach yields a prefix caching close to the optimal solution
minimizing WAN bandwidth usage on server-proxy path by
utilizing the proxy-client and client-client path bandwidth,
which is much cheaper than the expensive server–proxy path
bandwidth, server load, and client rejection ratio significantly
using chaining.

Keywords-component: prefix caching, cooperative clients, video
streaming, bandwidth usage, and Chaining.

I. INTRODUCTION

Generally streaming any multimedia object like high quality
video consumes a significantly large amount of network
resources. So request-to-service delay, network traffic,
congestion and server overloading are the main parameters to
be considered in video streaming over the internet that affect
the quality of service (QoS). So providing video-on-demand
(VoD) service over the internet in a scalable way is a
challenging problem. The difficulty is twofold first; it is not
a trivial task to stream video on an end-to-end basis because
of a video’s high bandwidth requirement and long duration.
Second, scalability issues arise when attempting to service a
large number of clients. In particular, a popular video
generally attract a large number of users that issues requests
asynchronously [2]. There are many VoD schemes proposed
to address this problem: batching, patching, periodical
broadcasting and prefix caching and chaining.

In the batching scheme [5,8 &10], the server batches the
requests for the same video together if their arrival times are
close, and serve them by one multicast channel. In the
batching scheme [2], the server sends the entire video clip to
the first client. Later clients can join the existing multicast
channel, and at the same time each of them requires a unicast
channel to deliver the missing part of the video.

Periodical broadcasting [12] is another innovative technique.
In this approach, popular videos are partitioned into a series
of segments and these segments are continually broadcasted
on several dedicated channels. Before clients start playing
videos, they usually have to wait for a time length equivalent
to the first segment. Therefore, only near VoD service is
provided.

Proxy caching [1, 4 & 9] is also a promising scheme to
alleviate the bandwidth consumption issue. In this approach,
there exists a proxy between a central server and client
clouds. Partial video (or entire video) files are stored in
proxies and the rest are stored in the central server. Proxies
send cached videos to clients, and request the remaining
from servers on behalf of clients. Recent works investigate
the advantages of connected proxy servers within the same
intranet [3, 4 and 8].

II. RELATED WORK

In this section we briefly discuss the previous work. Tay and
pang has proposed an algorithm in [7] called GWQ (Global
waiting queue) to reduce the initial startup delay by sharing
the videos in a distributed loosely coupled VoD system.
They have replicated the videos evenly in all the servers,
for which the storage capacity of individual proxy server
should be very large to store all the videos. In [11] Sonia
Gonzalez, Navarro, Zapata proposed an algorithm to
maintain a small initial start up delay using less storage
capacity servers by allowing partial replication of the videos.
They store the locally requested videos in each server. We
differ by caching the partial prefix-I at proxy and prefix-II at
tracker in proportion to popularity there by utilizing the
proxy server and tracker storage space more efficiently. In
[3] authors have proposed an hybrid algorithm for chaining, but they do not discuss about the scenario of client failure. LEMP in [6] also have proposed a client chaining mechanism and a solution for handling client failure situation involving too many messages which increases the waiting time for playback start to \( t_w \). Another approach to reduce the aggregated transmission cost has been discussed in [12] by caching the prefix and prefix of suffix at proxy and client respectively. But they have not considered chaining. [5 and 8] proposes a batching technique, which increases the client initial waiting time. Edward Mingjun Yan and Tiko kameda in [10] proposes a broadcasting technique which requires huge amount of storage capacity and sufficient bandwidth. Yang Guo in [2] has suggested an architecture to stream the video using patching technique. Hyunjoo and Heon in [13] have proposed another chaining scheme with VCR operations. But they do stream the video data from main server and considered a constant threshold value, due to which more number of clients may not be able to share the same chain. And WAN bandwidth usage on server-proxy path may be comparatively high.

In this paper, we propose an efficient load sharing algorithm and a new VoD architecture for distributed VoD system. This architecture consists of a centralized Main multimedia server [MMS] which is connected to a set of trackers [TR]. Each tracker is in turn connected to a group of proxy servers [PS] and these proxy servers are assumed to be interconnected using ring pattern. To each of this PS a set of clients are connected. And all these clients are connected among themselves. This arrangement is called as Local Proxy Servers Group [LPSG(Lo)]. Each of such LPSG, which is connected to MMS, is in turn connected to its left and right neighboring LPSG in ring fashion through its tracker. And an efficient prefix caching based chaining (PC+Chaining) scheme using proxy servers to achieve the higher network efficiency.

The organization of rest of the paper is as follows: Section 3 analyzes various parameters used in the problem. In section 4 we present a proposed approach and algorithm in detail, Section 5 describes Simulation model and section 6 describes the PC+Chaining scheme in detail, Section 7 presents the performance evaluation. Finally, in section 8, we conclude the paper and refer to further work.

### III. Model of the Problem

Let \( N \) be a stochastic variable representing the group of videos and it may take the different values (videos) for \( V_i \) \( (i = 1, 2, \ldots, N) \). And the probability of asking for the video \( V_i \) is \( p(V_i) \), let the set of values \( p(V) \) be the probability mass function. Since the variable must take one of the values, it follows that \( \sum_{i=1}^{N} p(V_i) = 1 \).

So the estimation of the probability of requesting \( V_i \) video, is 
\[
p(V_i) = \frac{n_i}{I}.
\]

Where \( I \) is the total number of observations and \( n_i \) is the number of requests for video \( V_i \). We assume that client’s requests arrive according to Poisson process with the arrival rate \( \lambda \). Let \( S_i \) be the size (duration in minutes) of \( i^{th} \) video \( i = 1, 2, \ldots, N \) with mean arrival rates \( \lambda_1, \ldots, \lambda_N \) respectively that are being streamed to the users using \( M \) proxy servers (PSs) of \( J \) LPSGs \( L_p = 1, 2, \ldots, J \). Each TR and PSq \( (q=1, M) \), has a caching buffer large enough to cache total \( P \) and \( B \) minutes of \( H \) and \( K \) number of video prefixes respectively. The complete video is divided into three parts, first \( W_i \) minutes of each video \( V_i \) is referred to as prefix-1 \( (\text{pref-1}) \), of \( V_i \), and is cached in any one of the proxy servers of the group only once. And next \( W_i \) minutes of video \( V_i \) is referred to as prefix-2 \( (\text{pref-2}) \) of \( V_i \), is cached at TR of \( L_p \).

\[
p = \sum_{i=1}^{N} (\text{pref-2})_i \quad \text{and} \quad B = \sum_{i=1}^{N} (\text{pref-1})_i.
\]

![Fig.1 Simulation Model](image)

Based on the frequency of user requests to any video, the popularity of the videos and size of \( (\text{pref-1}) \) and \( (\text{pref-2}) \) to be cached at PS and TR respectively is determined. And the size \( (W) \) of \( (\text{pref-1}) \) and \( (\text{pref-2}) \) for \( i^{th} \) video is determined as.

\[
W(\text{pref-1})_i = x_i \times S_i \quad \text{where} \quad 0 < x_i < 1
\]

\[
W(\text{pref-2})_i = x_i \times (S_i - (\text{pref-1})_i) \quad \text{where} \quad 0 < x_i < 1
\]

Where \( x_i \) is the probability of occurrence of user requests with frequency for video \( i \) from last \( t \) minutes. This arrangement caches maximum portion of most frequently requesting videos at \( L_p \). So most of the requests are served immediately from \( L_p \) itself, which reduces the network usage on server-proxy path significantly and makes the length of the queue \( Q_i \) almost negligible. Let \( b_i \) be the available bandwidth for \( V_i \) between the proxy and main server. After requesting for a video \( V_i \) at \( PS_a \), the WAN bandwidth required on server-proxy path for the video \( V_i \) may be 
\[
bw^{(\text{pref-1})}_i = bw(S_i - (\text{pref-1})_i - (\text{pref-2})_i)
\]

where \( i = 1, \ldots, N \), and \( bw^{(\text{pref-2})}_i \) is the WAN bandwidth usage required for \( i^{th} \) video on server to proxy path. This
depends on the amount of $v_i (\text{pref-1}) - (\text{pref-2})$ to be streamed from main server to the proxy. So the aggregate server-proxy bandwidth usage would be
\[
WAN_{bw}^{S-P} = \sum_{i=1}^{Q} bw(S - (\text{pref} - 1)(\text{pref} - 2))^i_{S-P}
\]

$bw()$ is a non linear function of $S(\text{pref-1} & \text{pref-2})$. And another output stochastic variable $R_{rej}$ is the request rejection ratio, which is the ratio of the number of requests rejected ($N_{rej}$) to total number of requests arrived at the system ($R$), which is inversely proportional to system efficiency $S_{eff}$.

That is $S_{eff} = \frac{1}{R_{rej}}$, where $R_{rej} = \frac{N_{rej}}{R}$ and $S_{eff} = \frac{Q}{R}$ is the ratio of number of requests served ($Q$) to total number of requests arrived ($R$) at the system. The optimization problem to maximize $S_{eff}$, thereby minimizing the client rejection ratio $R_{rej}$ at the PS, and average WAN bandwidth usage $WAN_{bw}^{S-P}$ is.

Maximize System Efficiency $S_{eff} = \frac{Q}{R}$

Minimize Avg WAN Bandwidth usage
\[
WAN_{bw}^{S-P} = \sum_{i=1}^{Q} bw(S - (\text{pref} - 1)(\text{pref} - 2))^i_{S-P}
\]

\[
Y = \frac{1}{Q} \sum_{i=1}^{Q} (Wt_i)
\]

Subject to
\[
B = \sum_{i=1}^{K} (\text{pref} - 1)i, \quad P = \sum_{i=1}^{H} (\text{pref} - 2)i
\]

$W(\text{pref} - 1) & W(\text{pref} - 2) > 0$

IV. PROPOSED ARCHITECTURE AND ALGORITHM

A. Proposed Architecture

The proposed VoD architecture is as shown in Fig.2. This architecture consists of a MMS, which is connected to a group of trackers (TRs). As shown in Fig.3, each TR has various modules like Interaction Module (IM$_{TR}^{}$) – Interacts with the PS and MMS. Service Manager (SM$_{PS}^{}$) – Handles the requests from the PS. Database – Stores the complete details of presence and size of (pref-1) of videos at all the PSs. Video distributing Manager (VDM) – Responsible for deciding the videos, and sizes of (pref-1), (pref-2) of videos to be cached. Also handles the distribution and management of these videos to group of PSs, based on video’s popularity.

And each TR is in turn connected to a set of PSs. These PSs are connected among themselves in a ring fashion. Each PS has various modules like Interaction Module of PS (IM$_{PS}^{}$) – Interacts with the client and TR. Service Manager of PS (SM$_{PS}^{}$)– Handles the requests from the user, Client Manager (CM) – Observes and updates the popularity of videos at PS as well as at TR as shown in Fig.3. And also to each of these PSs a large number of users are connected [LPSG]. Each proxy is called a parent proxy to its clients. All these LPSGs are interconnected through their TR in a ring pattern. The PS caches the (pref-1) of videos distributed by VDM, and streams this cached portion of video to the client upon the request through LAN using its less expensive bandwidth. We assume that,

1. The TR is also a PS with high computational power and large storage compared to other proxy servers, to which clients are connected. It has various modules, using which it coordinates and maintains a database that contains the information of the presence of videos, and also size of (pref-1) and (pref-2) of video in each PS and TR respectively.

2. Proxies and their clients are closely located with relatively low communication cost[1]. The Main server in which all the videos completely stored is placed far away from LPSG, which involves high cost remote communication.
threshold time $t$ by the clients. Here, we assume that the frequency of requests to a video follows Zipf law of distribution.

B. Proposed Algorithm

1. When a request from $C_i$ for video $V_i$ arrives at time $t$ to a particular proxy $PS$, 
2. Do the following
3. If $(V_i$ is present at $PS$) 
4. Check the $SCL$
5. If $IS=STREAMING$ is TRUE & $((t-m) \leq (pref_i), s)$ where $j=1, d$
6. req-hdr of $PS$ sends msg with $LAC$ to $C_i$ and updates $SCL$
7. else serv-mgr starts the new stream to $C_i$ and adds $C_i$ to $SCL$
8. else pass the req to $TR$, $TR$ checks in $L_p$
9. If $(V_i$ is present at some $PS$ of $L_p$) 
10. $TR$ initiates the new stream to $C_i$ thru $PS$ & serv-mgr updates the $SCL$
11. else $TR$ checks at $NBR[L_p]$
12. If $(V_i$ is present at some $PS$ of $NBR[L_p]$)
13. $TR$ initiates the new stream to $C_i$ thru $PS$
14. else $TR$ decides to download the $V_i$ from $MMS$
15. Start new stream to $C_i$, compute $(pref_1), (pref_2), (pref_3)$
16. $W((pref_1), (pref_2), (pref_3))$ can be cached
17. Cache $(pref_1), (pref_2), (pref_3)$ at $PS$ & $TR$ respectively and update $SCL$
18. else try to make room for $(pref_1), (pref_2)$ at $L_p$ using some buffer allocation algorithm

Whenever a client $C_i$ requests for the video $V_i$ at $PS$, request-handler (req-handler) checks whether $IS=STREAMING$ flag of that video is true and the arrival time difference of the new client and the most recent client of the existing chain of $V_i$ is below the threshold $W(pref-1)$ then the service-manager (serv-mgr) adds $C_i$ to the existing client chain of $V_i$ and instructs it to get streamed from any one of the last $d$ clients of the chain. Then by adding $C_i$ to the client list of $V_i$, service-mgr updates the Streaming Clients List ($SCL$), which is the list with complete details of list of videos being streamed from that PS and the corresponding chain of clients of that video [line 3-7 of proposed algorithm]. Otherwise $PS$ starts the new stream to $C_i$ and new chain to $V_i$ and $SCL$ is updated by creating a new entry for $V_i$ [line 8]. If it is not present in its cache, the $IM_{req}$ forwards the request to its parent $TR$, VDM at $TR$ searches its database using perfect hashing to see whether it is present in any of the $PS$s in that $L_p$. If it finds starts streaming from that $PS$ to $C_i$ and updates the $SCL$ accordingly [line 9,10 & 11]. Otherwise request is passed to $TR$ of $[NBR[L_p]]$. If $V_i$ is found there it will be streamed from that $PS$ to $C_i$ and $SCL$ is updated accordingly by service-manager [line 12,13 & 14]. If $V_i$ is not found at $NBR[L_p]$ also then the $V_i$ is downloaded from the MMS and is streamed to $C_k$. While streaming the $W(pref-1)$ and $W(pref-2)$ of $V_i$ is calculated as

$$W(pref-1) = \chi_i \times S_i$$

where $0<\chi_i<1$ and

$$W(pref-2) = \chi_i \times (S_i-(pref-1))$$

and cached at $PS$ and $TR$ respectively, if sufficient space is available. Otherwise an appropriate buffer allocation algorithm is executed to make space to cache these prefixes of $V_i$ according to popularity [line 15,16 & 17 & 18]. So most of the requests are served immediately from the client, who is already being served or from $L_p$ by sharing the videos present among the $PS$s, which reduces the client rejection-request ratio $R_{req}$, load at the server $S_i$, and increases the video hit ratio $VHR$. 

$$W(pref-2) = \chi_i \times (S_i-(pref-1))$$

where $0<\chi_i<1$ and
V. SIMULATION MODEL

In our simulation model we have a single MMS and a group of 6 TRs. All these TRs are interconnected among themselves in a ring fashion. Each of these TR is in turn connected to a set of 6 PSs. These PSs are again interconnected among themselves in a ring fashion. To each of these PS a set of 25 clients are connected and all these clients are interconnected among themselves. We use the video hit ratio (VHR), the average client waiting time $Y$ to measure the performance of our proposed approach more correctly. In addition we also use the WAN bandwidth usage on MMS-PS path and probability of accessing the main server as the performance metrics.

| Notation | System Parameters | Default Values |
|----------|-------------------|----------------|
| $S$      | Video Size        | 2-3hrs, 200units/hr |
| $C_{MMS}$ | Cache Size(MMS)   | 1000 |
| $C_{TR}$ | Cache Size(TR)    | 400(40%) |
| $C_{PS}$ | Cache Size(PS)    | 200(20%) |
| $\lambda$ | Mean request arrival rate | 44 req/hr |

We assume that the request distribution of the videos follows a zipf-like distribution. The user request rate at each PS is 35-50 requests per minutes. The ratio of cache sizes at different elements like MMS, TR and PS is set to $C_{MMS} : C_{TR} : C_{PS} = 10:4:2$ and transmission delay between the proxy and the client, proxy to proxy and TR to PS as 100ms, transmission delay between the main server and the proxy as 1200ms, transmission delay between tracker to tracker 300ms, the size of the cached $[(pref-1)+(pref-2)]$ video as 280MB to 1120MB(25min – 1hr) in proportion to its popularity.

VI. PC+CHAINING

Our proposed scheme PC+Chaining is an efficient streaming technique that combines the advantages of prefix caching and both client-server and peer-to-peer approaches to cooperatively stream the video using chaining. The main goal of PC+Chaining is to make each client act as a server while it receives the video, so that the available memory and bandwidth of the clients can be utilized more efficiently. The un-scalability of traditional client-server unicast VoD service lies in the fact that the server is the only contributor and can thus become flooded by a large number of clients submissively requesting the service. In the client-server service model, a client sets up a direct connection with the server to receive the video. In this case an amount of WAN bandwidth requirement on server-proxy path equal to the playback rate is consumed along the route. As the number of requests increases, the bandwidth at the server and in the network is consumed so network becomes congested and the incoming requests must eventually be rejected [2]. In contrast, we propose two schemes to address these issues 1) Local group of interconnected proxies’ and clients architecture with prefix caching technique and load sharing among the proxies of the group reduces the frequent access to MMS which in turn reduces the amount of bandwidth consumption between client and main server. 2) PC+Chaining, where the clients not only receive the requested stream, but also contribute to the overall VoD service by forwarding the stream to other clients, whose request arrives within the threshold time of $W(pref-1)$. In PC+Chaining all the clients are treated as potential server points. When there is a request for a video $V_i$ from the client $C_k$ at particular proxy $PS_q$, if the requested video $V_i$ is present at $PS_q$, then the service will be provided to $C_k$ in the following stages.

C. Client admission phase

When the request arrives at $PS_q$, the request-handler (req-handler) of that proxy checks for the presence of the video at $PS_q$. If it is present then it checks the flag IS-STREAMING of the video $V_i$. If it is not true indicates that, there are no clients existing having streamed with the same video object $V_i$. Then the req-handler informs the service-manager (service-mgr) to provide the streaming of $V_i$ to $C_k$. So the service-mgr starts new stream and updates the streaming clients list (SCL) by adding a new entry for the video $V_i$ along with its (pref-1) size. The format of each entry of SCL is $<video id - sz(pref-1)- list of clients being streamed with V_i>$. If the flag IS-STREAMING is true, indicating that there is already a chain of clients being streamed with the same video. Then the req-handler looks into the corresponding entry for $V_i$ in SCL to check whether the arrival time difference of the new client and the most recent client of the existing chain of $V_i$ is below the threshold $W(pref-1)$, if so $C_k$ is added to the existing chain of $V_i$ and sends the sub list of d applicant clients (LAC) to $C_k$, along with the msg to get streamed from the last client of the list LAC. Also sends the msg to all d clients of LAC, so that the client agent of these clients sends ready signal to $C_k$. Then $C_k$ sends start signal generally to the most recent client being streamed with $V_i$ and starts getting the stream. This client becomes the parent client(PC) of $C_k$. In case any problem with this client, $C_k$ can request the other clients.

At the end of this client admission phase, SCL will have the complete details of list of videos being streamed and the corresponding chain of clients of that video along with the prefix size. Also each client knows its applicant ancestors to change its parent client (PC) in case of PC failure.

D. Streaming phase

We assume that the client has sufficient buffer space and network bandwidth to accommodate the (pref-1) of $V_i$ when the client $C_k$ requested for $V_i$ starts getting the stream from the last client of LAC, the buffer manager (buf-mgr)
starts buffering the video clips received from its parent client. And media-player starts playing it from the buffer.

E. Closing Phase

Any client \(C_i\) in the chain may wants to close the connection in the following two cases

case1: Once the streaming of the complete video is finished. The client agent closes the connection with its parent client (PC) by sending close signal to it, but checks whether this client is PC for any other clients in the chain before sending close signal to its parent proxy server (PPS). If so, it waits until the streaming of the complete video to its child client (CC) finishes and then sends the close signal to its PPS and then closes the connection. The service-mgr updates the stream details of its PPS, and the same is intimated to the requested \(PS_i\) and hence request-service delay is very small.

If \(V_i\) is not present in any of the PSs in that \(L_p\), then the IMTR passes the request to the tracker of NBR(\(L_p\)). Then the VDM(NBR(\(L_p\))) checks its database using perfect hashing, to see whether the \(V_i\) is present in any of the PSs of its \(L_p\). If it is present, then the SM(\(L_p\)) in turn initiates the streaming of \(V_i\) to the requested \(PS_i\) through the optimal path, and the same is intimated to the requested \(PS_i\) and hence request-service delay is very small.

If \(V_i\) is not present in any of the PSs in that \(L_p\), then the IMTR passes the request to the tracker of NBR(\(L_p\)). Then the VDM(NBR(\(L_p\))) checks its database using perfect hashing, to see whether the \(V_i\) is present in any of the PSs of its \(L_p\). If it is present, then the SM(\(L_p\)) in turn initiates the streaming of \(V_i\) to the requested \(PS_i\) through the optimal path, and the same is intimated to the requested \(PS_i\) and hence request-service delay is very small.
service delay is comparatively high but acceptable because it bypasses the downloading of the complete video from MMS using MMS-PS WAN bandwidth.

If the \( V_i \) is not present in any of the PSs of its NBR(Lp) also, then the TR(Lp) modules decide to download the \( V_i \) from MMS to PSq. So the IM(Lp) coordinates with MMS to download the \( V_i \), and hence the request-service delay is very high, but the probability of downloading the complete video from MMS is very less as shown by our simulation results.

Whenever the sufficient buffer and bandwidth is not available in the above operation the user request is rejected.

VII. PERFORMANCE EVALUATION

In comparison with the prefix caching without chaining (PC-Chaining), we use \( T=\text{sz}(\text{pref-1}) \) as the threshold value and following performance metrics in our evaluation:

i) Client rejection ratio \( R_{rej} = \frac{N_{rej}}{R} \) where \( N_{rej} \) is the number of requests rejected, \( R \) is the total number of requests arrived.

ii) Average network bandwidth usage on MS-PS path \( BW = \sum_i b_{wi} \)

iii) Main Server load \( S \)

iv) Number of requests served from \( L_p \).

**Client rejection ratio & Video hit ratio:** Fig. 8 shows the client rejection ratio of our proposed system, which is very less. We can observe that PC+Chaining allows more clients to join the chain, because as the request arrival rate increases \( T(W(\text{pref-1})) \) is also increases. This keeps IS-STREAMING flag of some client in the chain true. So the request can be served immediately from the PS by adding it to the existing chain, which increases the video hit ratio as shown in fig. 7 and significantly reduces the client waiting time as shown in the fig. 11.

**Average server-proxy bandwidth usage:** The main server may reject many requests due to its bandwidth constraints, when the clients directly request the main server. Our proposed \( L_p \) architecture and a scheme of caching maximum portion of (Pref-1) and (Pref-2) of most frequently asked videos at PS and at TR respectively, and load sharing algorithm with PC+Chaining increases the system efficiency and reduces the average MMS-PS network bandwidth usage as shown in fig. 6. and fig. 9.

![Fig. 6 No. of Requests served vs Prefix Size of popular videos](http://sites.google.com/site/ijcsis/)

![Fig. 7 Video Hit Ratio at LPSG, PC with chaining vs without chaining](http://sites.google.com/site/ijcsis/)

### Table 2: Simulation results

| Notation | System Parameters | Results |
|----------|-------------------|---------|
| \( Q(\text{Total}) \) | Total No. of requests served at \( L_p \) | 320 |
| \( Q(\text{PS}) \) | No. of requests served with PC+Chaining from Lp | 296(93%) |
| \( B_W^{\text{P}} \) | Mean bandwidth usage from server to Proxy | 21% |
| VHR | Average Video Hit ratio | 92% |
| \( Y \) | Client waiting time | 2sec - 3Sec |
| \( S_L \) | MMS Load reduction | 70% |

Main Server load: As popularity based (Pref-1) and (Pref-2) of most of the videos are cached and streamed from \( L_p \) itself, main server is contacted for least amount of video data (S-(Pref-1)-(Pref-2)), and for the complete video very rarely, which has reduced the server load significantly as shown in fig. 10. This in turn reduces the WAN bandwidth usage from MMS to PS as shown in the Fig. 9.

**Impact of PC+Chaining on Client waiting Time:** Our proposed approach increases the aggregate storage space of \( L_p \) by distributing the large number of videos across the PSs and \( TR \) of \( L_p \), and hence achieves the high cache hit rate there by reducing the client waiting time significantly. For example, if 10 PSs within a LPSG managed 500 Mbytes each, total space available is 5 GB. 200 proxies of LPSG could store about 100 GB of movies. Generally the scalability of PC-Chaining improves because of the popularity based presence of the video at any one of the PS in \( L_p \). As the size of the (Pref-1) increases, more and more number of clients can be joined to the existing chain of the video and they can be served immediately from the \( L_p \) itself. This reduces the client waiting time significantly as shown in the fig. 11, which also reduces the WAN bandwidth usage along server-proxy path as depicted in the fig. 9.
In this paper we have proposed an efficient video sharing mechanism with chaining and an architecture where proxies and clients cooperate with each other to achieve reduced network bandwidth usage on the server-proxy path and client rejection ratio, by caching and streaming maximum portion of the most frequently requested videos from $L_p$, and by sharing the video data of the currently played video object with other existing clients. And the popularity based $W(\text{pref-1})$ allows more number of clients to get benefit from the existing chain. Our simulation results demonstrated that our proposed approach has reduced the average bandwidth usage on server-proxy path, and also the load of main server, client rejection ratio and achieves the client failure recovery successfully. The future work is being carried out to improve the performance using dynamic buffer management at PS.

REFERENCES

[1] LiZhu, Gang Cheng, Nirwan Ansari, Zafer Sahinoglu “Proxy Caching for Video on Demand Systems in Multicasting” CISS, The Johns Hopkins University, March 12-14, 2003.

[2] Yang Guo, Kyungwon Suh, Jim Kurose “P2Cast: Peer-to-Peer patching scheme for VoD service” www2003, May 20-24, 2003, Budapest, Hungary ACM 1-58113-680-3/03/0005

[3] Fouliras P, Xanthos S, Tsantalis N, Manitisaris A (2004) LEMP: lightweight efficient multicast protocol for video on demand. ACM Symposium on Applied Computing, pp 14–17, March

[4] Lian Shen, Wei Tu, and Eckehard Steinbach “A Flexible Starting point based Partial Caching Algorithm for Video On Demand” 1-4244-1017/07 @2007 IEEE.

[5] Wallace K. S. Tang, Eric W M. Wong “Optimal Video Placement Scheme for Batching VOD services” IEEE TRANSACTIONS ON BROADCASTING, Vol. 50 No.1 MARCH 2004.

[6] Roopalakshmi.R. and Ashok Kumar “Hybrid Chaining Scheme for Video-on-Demand Systems”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, Vol.12, No.3, May/June 2000.

[7] Simon Sheu, Kien A. Hua “Chaining: A Generalized Batching Technique for Video-On-Demand”, 0-8186-7819-4/97 ©1997 IEEE

[8] Edward Mingjun Yan and Tiko Kameda “An Efficient VOD Broadcasting Scheme with User Bandwidth Limit”, SPIE/ACM MM CN 2003, Santa Clara, CA, Jan. 2003.

[9] S. González, A. Navarro, J. López and E.L. Zapata, “Load Sharing in Distributed VoD (Video on Demand) Systems”. (SGRR 2002w), L’Aquila, Italy, January 21-27, 2002.
[12] “COPACC: An Architecture of cooperative proxy-client caching System for On-Demand Media Streaming. A report Alan T.S Ip, Jiangchuan Liu, John C.S.Lui.

[13] Hyunjoo Kim, Heon Y. Yeom “P-chaining: a Practical VoD service scheme automatically handling interactive operations” @springer Science+Business Media, 11 July 2007.

AUTHORS PROFILE

M Dakshayini holds M.E degree in Computer science. She has one and half decades of teaching experience. She has published many papers. She is currently working as teaching faculty at Information and science and engineering department, BMS College of engineering, Bangalore, India.

T.R. Gopalakrishnan Nair holds M.Tech. (IISc, Bangalore) and Ph.D. degree in Computer Science. He has 3 decades experience in Computer Science and Engineering through research, industry and education. He has published several papers and holds patents in multi domains. He won the PARAM Award for technology innovation. Currently he is the Director of Research and Industry in Dayananda Sagar Institutions, Bangalore, India.