An Integrated Quality-of-Service Model for Video-on-Demand Application

*D. N. Sujatha [1], K. Girish [2], K. R Venugopal [1], L. M. Patnaik [3]
[1] University Visvesvaraya College of Engineering, Bangalore University, Bangalore-560001, India.
[2] B. M. S College of Engineering, Basavanagudi, Bangalore-560019, India.
[3] Microprocessor Applications Laboratory, Indian Institute of Science, Bangalore-560012, India.
*

Abstract— **The tremendous growth of the Internet paradigm has given rise to Quality of Service (QoS) problems in heterogeneous, ubiquitous, distributed real time applications such as video-on-Demand (VoD). The challenging task in VoD applications is to satisfy diverse client requests for discrete videos with restrained resources by invoking versatile QoS schemes. In this paper, a hybrid QoS strategy, which is a combination of batching and recursive patching is implemented in the local server to ensure starvation-free resource management thereby enhancing the throughput. Batching shares network resources efficiently whereas recursive patching is adopted to reduce the time difference between the requests. The suggested algorithm delivers the complete video to the users based on one of the three communication channels: broadcast, multicast and unicast depending on whether the video is very popular, average popular and least popular respectively. The experimental results show that our strategy accomplishes 35% - 40% reduction in terms of blocking ratio and throughput is 10% - 15% higher than the Poon's strategy, which guarantees that not only the resources are efficiently utilized but also a suitable Quality of Service is provided to each user.**

Keywords:Quality of Service, Video-on-Demand, Blocking ratio, Start-up delay, Jitter.

1 Introduction

Distributed computing is a decentralized paradigm with wide geographic dispersion of resources. The main goal of distributed computing system is to connect users and resources in a transparent, open and scalable way enabling more fault tolerant than stand alone system thereby ensuring that each subsystem is continually open to interaction with other systems. Hence distributed system can be altered to accommodate changes in a number of user, resources and computing information.

Present Video-on-Demand system are designed using client-server architecture. In this architecture client will send a request to a video server requesting for a specific video and the server transmits it to the client for playback. As the number of severs increase the server will reach its limit on the capacity. In this case further increase in load enable introduction of a new server. The cost on the video server is tremendously increasing and the communication network also need to be upgraded with additional bandwidth. This problem was alleviated with the birth of the Distributed system. The distributed system reduces the cost overhead as it reduces the the need for dedicated server. This architecture is inherently scalable as addition of new user to the system adds both streaming load and streaming capacity to the system. In this architecture it is necessary to decide the placement policy for videos, an algorithm to schedule and transmit the video. As the videos are distributed in the entire system an index table is required for retrieval of the videos. As there is no centralized control each machine should manage heterogeneity, addition and deletion of nodes.

QoS has emerged as an important research area in ubiquitous, distributed multimedia applications like Internet shopping, games, news-on-demand, tele-teaching and Video-on-Demand which involve many geographically distributed clients. Distributed multimedia applications rank as one of the highest consumers of resources: CPU, bandwidth and buffer as they need resource management to ensure end-to-end QoS and to regulate resource contention for fair sharing of resources. In order to provide continuous media service such as Video-on-Demand, a suitable QoS requested by the user (client) has to be guaranteed by taking into account, the characteristics offered by the media data, the processing capabilities of both the client and the video server.

Quality-of-Service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system like Video-on-Demand that are necessary to achieve the required functionality of an application. Quality-of-Service becomes an important

*Tel.: +91-080-2662 2130-35; fax: +91-080-2661 4357.
E-Mail Address : suj_sat@yahoo.com

issue as systems are more open, therefore less predictable. In such a context, expected Quality-of-Service is difficult to achieve with static approaches. In dynamic approaches, services are adapted to provide the best Quality-of-Service according to the execution context. In recent years there has been a flurry of research activity devoted to the development of new architectures and service models to accommodate next generation applications demanding Quality-of-Service.
In spite of the diversity, dynamic and heterogeneous nature of the client, the design of these applications requires careful consideration of QoS for streaming videos. The diffusion of mobile telecommunication and mobile access to the Web further widens the heterogeneity of Internet client devices. Users tend to require differentiation and tailoring of QoS, based on personal preferences and classes of usage such as business/economics/free of charge by considering accounting aspects. The key issue in any video service is to provide an acceptable QoS to the end user. The video service applications that are currently being deployed have some significant problems, user dissatisfaction due to poor QoS and low cost performance ratio due to inefficient management of system resources especially when guaranteed service is desired. In order to resolve these issues, it is necessary to identify the bottlenecks in the system that are responsible for poor response time.
The design, implementation and deployment of data sharing techniques can significantly enhance the functionality of Video-on-Demand system. Batching, adaptive piggybacking, patching and bridging are the variegated data sharing techniques. Batching cummulates the requests for the like video, hence batching alleviates the load on the server. Among many different batching techniques, batching by timeout and batching by size [1] are the two commonly used schemes. Inspite of simplicity both the schemes introduce initial latency, hence adaptive piggybacking [2] and patching [3] are proposed. In adaptive piggybacking, different requests are merged into a single stream by varying the play-out rates. Patching expands the multicast tree [4] dynamically to include new requests, so it reduces the request waiting time but requires additional bandwidth and buffer space at the client. Bridging uses buffer space to retain certain portions of the video as they play for a particular viewer, then a viewer who is trailing can be served from this buffer. To manage resource contention for unbiased data sharing we should identify the loop holes in the system those are responsible for poor response time. The challenging issue for wide deployment of Video-on-Demand is to provide QoS with value added services.

Motivation: The fundamental parameters in achieving QoS for various applications are reliability, delay, jitter and bandwidth. Reliability and delay need not be stringent for a VoD system as it can tolerate a small amount of error and if all the video segments are delayed uniformly by a few seconds, there is no loss. Jitter and bandwidth are the two stringent QoS parameters considered for analysis. User satisfaction and resource management are critical in order to provide Quality of Service in a VoD system. User satisfaction is abstract and is subject to the visual perception of the individual. Although resources like bandwidth and buffer are available in abundance, it is to be efficiently managed. The proposed Hybrid Quality of Service strategy - HQoS addresses the following issues: to optimize resource utilization by using data sharing schemes, to reduce network bandwidth requirements by using appropriate data transmission techniques and to curtail start-up latency and reneging probability thereby providing jitter-free video to the users.

Contributions: The proposed Data Sharing Strategy (DSS) for guaranteeing Quality-of-Service in Video-on-Demand application addresses the following issues: to optimize resource utilization by using data sharing schemes, to reduce network bandwidth requirements by using appropriate data transmission techniques and to curtail initial latency and reneging probability thereby providing jitter-free video to the clients.

Outline of the Paper: The paper is organized into various sections as follows: Related works in the area of Quality of Service in Video-on-Demand system are presented in Section 2. Section 3 discusses the system architecture, the functional models and formulates the problem for providing QoS. Section 4 proposes Hybrid QoS strategy to solve the problem. Section 5 evaluates the Hybrid QoS strategy in VoD system through extensive analysis and simulation. Section 6 presents conclusions.

2 Related Works

This section presents a brief description of the existing research works in the area of Quality of Service in Video-on-Demand systems. The fundamental problem in large-scale networks is to satisfy QoS to the end user and achieve economic viability. The problem of resource allocation is discussed in [4] by partitioning the requests based on divide and conquer scheme and precomputation techniques. Divide and conquer scheme reduces the computational complexity independent of the network size. The overall network performance is maximized by adapting to a suitable apportioning scheme, thereby satisfying the end-to-end QoS requirements. In precomputation techniques, certain computations are performed in advance during the period when the resource is not in use. This reduces the time required to handle the request. The precomputation technique is carried out based on estimation which is based on heuristics leading to inconsistency. In [5], admission control and resource reservation schemes

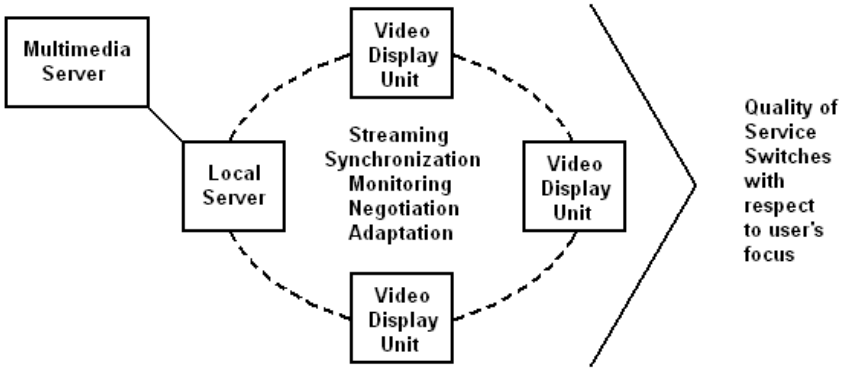

Figure 1: Distributed Architecture for VoD System.

are addressed. The admission control scheme divides the problem into smaller sub-problems: division of end-to-end QoS into the local QoS requirement, mapping of the local QoS requirement into the resource requirement, and reclaiming of the resource allocated in excess. The resource allocation problem is addressed by establishing a multicast tree and then reserve the necessary resources reducing the call rejection probability. The algorithm is developed for static groups where all the receivers are known before the session starts.

Distributed multimedia applications require dynamic QoS [6] for continuous streaming of multimedia. Load balancing can be achieved by minimizing load on heavily-loaded machines i.e., minimizing Bottleneck Resource Utilization (BRU). A client server architecture; Adaptive Distributed Multimedia System (ADMS) along with reduced BRU, is proposed. In ADMS, the user has to state the QoS requirements. The system has the flexibility to accept or reject the request based on the availability of the resources. It presents a unified feasible way to solve admission policy but fairness is not guaranteed.

In order to deliver good QoS, the VoD service should be nearly immediate and continuous [7], [8]. A set of work load models are developed to identify the limitations of greedy allocation algorithm: (i) they do not minimize customer's waiting time. (ii) they perform poorly when load conditions vary. To solve these limitations, a set of rate-based policies are proposed, which work by ensuring that the channels are available for allocation on consistent basis. In this method, it is difficult to determine the number of channels needed to provide the desired QoS.

In [9], [10], [11], [12], [13], various issues to manage group dynamics, resource reservation, allocation of resources and admission control in multicast applications are examined. Dynamic group management is critical as members can join and leave the group at any instant of time. Resources must be reserved appropriately as excess reservation leads to the wastage of the resources. Poon et al. [14] proposed an algorithm to dynamically find the batching time by newly updated arrival rate so as to minimize the bandwidth which is a complex procedure and time consuming. The Patching scheme proposed in [15], [16], [17] extends the capability of the standard multicast to support true Video-on-Demand. All the above techniques will perform efficiently only if enough resources are available. Ramesh et al. [18] proposed a hybrid strategy which is a combination of two popular approaches: pyramid broadcasting and recursive patching. Fragmentation of video into segments enables VCR functions like pause and resume, rewind and forward to be handled easily.

3 System Architecture

3.1 Definitions

- *QoS* is the collective effect of service performances which determine the degree of satisfaction of the client.
- *Initial latency* is defined as the time gap between the instance when the request is submitted to the system and the beginning of service.
- *Scheduling* is the process of assigning a video request to a set of resources.
- *Jitter* is the maximum delay between two consecutive frames. It increases under chaotic load patterns. If jitter is high, play-out process will pause, annoying the client.
- *Throughput* is defined as the ratio of the number of video requests served to the total number of video requests.

The distributed architecture for providing Quality of Service in Video-on-Demand system is shown in Fig. 1. Many servers geographically separated from each other

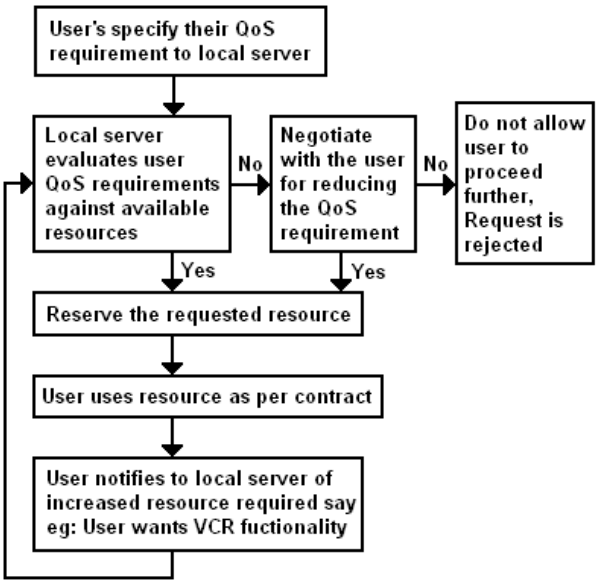

Figure 2: QoS Mapping from Local Server to Client.

are interconnected in hierarchy. A typical Video-on-Demand service allows the remote users to play any video from a large collection of videos stored on one or more servers. In response to a request, the server delivers the videos to the clients. The number of users supported by a server depends on the available bandwidth. Streaming of multimedia data, synchronization of audio and video data, monitoring of network resources, negotiation and adaptation of QoS parameters based on the available resources are the tasks of the local server. Hence QoS provided by the system varies with respect to the users focus.

The QoS mapping from local server to client is depicted in Fig.2, it includes the following process:

Client QoS Specification: The QoS specification for a Video-on-Demand system is a sequence of acceptable QoS requirements, where the client specifies his requisites in terms of a client ID, requested video ID, VCR function to the local server, in order to obtain the desired QoS.

Mapping client QoS Specification to the Available Resources: The local server compares the resources required for a specific video requested by the client with the available resources. If necessary resources are available, they are reserved and the client can start watching the video as per negotiation; else the client reduces the QoS requirements and once again submits the new requirements to the local server. This process repeats until the client specifications are matched with the available resources.

QoS Resource Bottleneck: Although the level of service can be negotiated, problems can occur during communication. A part of the route might become unavailable i.e., resource can drop due to network failure / congestion or some hosts may use more resource than available and hence these situations deteriorate the performance of the system. In order to avoid this situation various congestion control techniques are adopted.

3.2 Problem Definition

We model the network as an undirected graph M(L,U). The multimedia server M consists of a finite set of local servers $L = l_1, l_2, \ldots, l_n$. Each local server l_i is connected to a set of clients $U = u_1, u_2, \ldots, u_m$, with $u_j \neq u_k$ where $j, k \in 1, 2, 3, \ldots, m$. We associate e with each link, represented by $P(e)$ which denotes the available resources e.g., bandwidth and buffer. For each client u requesting for a video v, $P(v)$ describes the quantity of resources required by the video. In this model, a tree T is rooted at the multimedia server M spanning all the local servers L and all the clients U satisfying the condition $P(v) \leq P(e)$.

The objective of this paper is to

- reduce the blocking ratio, thereby increase the throughput.
- share resources efficiently using data sharing techniques like batching and recursive patching.
- minimize the initial latency to the client.
- provide VCR functions On-Demand.

A combination of recursive patching with / without batching is adopted in this model. Broadcasting along with recursive patching and batching is used for the popular videos. Multicasting along with recursive patching without batching is used for the average popular videos and the least popular videos are unicast. Initially, the requested videos that need to be broadcast, multicast and unicast are assigned one channel each and the remaining channels are used for recursive patching. The channels are allocated based on ECPV [14]. The popular videos are patched using multicast technique and average popular videos by unicast. Table 1 indicates the list of variables used in this paper.

Table 1: List of Variables.

Notation	Description
M	The number of videos stored in the server.
UB	The bandwidth for unicast channel.
MB	The bandwidth for multicast channel.
BB	The bandwidth for broadcast channel.
λ_j	The arrival rate for video j with Poissons arrival rate.
λ_j^{max}	The upper bound on the average popular requests.
λ_j^{min}	The lower bound on the average popular requests.
Len_j	The length of video request j.
W	The patching window for most popular video.
Y	The batching time for most popular video.
RID	The ID for regular channel.
PID	The ID for patching channel.
$skew$	The difference in the length of the video between time of arrival of new request and the start of patching window.
$workload$	The length of the video that is to be transmitted by a patching request.

3.3 Analytical Model

A combination of recursive patching with / without batching is adopted in this model. Broadcasting along with recursive patching and batching is used for the popular videos. Multicasting along with recursive patching without batching is used for the average popular videos and the least popular videos are unicast. Initially, the requested videos that need to be broadcast, multicast and unicast are assigned one channel each and the remaining channels are used for recursive patching. The channels are allocated based on ECPV [14]. The popular videos are patched using multicast technique and average popular videos by unicast. Table 1 indicates the list of variables used in this paper.

The requests are batched based on the criteria; max(δ, Y) where δ is the number of requests arriving within a specified time. It can be represented mathematically as follows:

$$\text{Requests for video j} = \begin{cases} \delta & n(R_j) = \delta \\ n(R_j) & t_j = nY. \end{cases}$$

where n is the number of requests arriving at time t_j. Mean number of customers arriving within the batching time Y is given by $\max((\lambda * Y), 1)$. A new transmission stream is initiated at the start of the patching window W. Patching is carried out only if the length of the video streamed is less than W.

Case (a):
The request R_n, where $n = 1, 2, 3, \ldots, m$ requesting for a video v_i where $i = 1, 2, 3, \ldots, y$ arriving at the beginning of a patching window p is represented by,

$$R_{n+1} = \sum_{i=1}^{n} \mathrm{v}_{i+1} \; {}^*R_i + (v_1{}^*\mathrm{p})$$

Case (b):
As shown in Fig. 3, if a new request arrives before 60% of the video is transmitted, then it can be patched as follows; for example, if the new request R_2 arrives at time t_{11} and the end of the video segment is t_2 then patching can be done according to,

$$R_2 = ((\ t_2 - t_{11})^*\ R_1 + (\ t_{11} - t_1)^*\mathrm{p})$$

The number of channels are fixed, hence in the worst case a request which needs patching might need to wait until the free channels are available. The requests arrive according to Poisson process denoted by $((\lambda^x)^* (e^{\lambda}))/\ x!$ represented as p, where λ is a constant and x is a random variable. The service time of the patching requests is uniformly distributed between 0 and W, denoted by u. The maximum patching size is $2W$ and the minimum is zero. Hence average service time is $2W / 2^k$ where k = 0, 1, 2, 3,. . . ,n. All the requests are handled using single server. The patchi1ng service is modeled as p / u / 1 queuing model. If v is the video that is to be transmitted to the nodes $(n_1, n_2, \ldots, n_x)$, $v * \sum_{i=1}^{x} n_i$ represents the the video transmitted at once to all the nodes in the network. Let g_i denote the group which consists of nodes $(n_1, n_2, \ldots, n_x)$ belonging to the multicast group. The videos delivered to a group g_i is denoted by $v * \sum_{i=1}^{x} g_i(n_i)$. The video delivered to a node in the network is represented as $(v{*}n_i)\ \forall i = 1, 2, 3, \ldots, n$.

Reneging occurs when a customer decides to leave

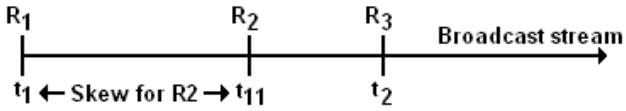

Figure 3: Request Arrival.

the system without being serviced after waiting for some time. This is expressed in terms of variable γ, such that the customer for a video will not wait for more than γ units of time. The function for reneging or blocking ratio is denoted by $Renege(x)$, which is the fraction of the rejected customers who leave after waiting for x units of time. Reducing blocking ratio improves throughput. The blocking ratio of the customer for a video in an interval t is given by,

$P(t) = \frac{1}{t} \int_0^t Renege(x)dx$ at any instant of time.

We expect the reneging probability to be small since all the customers do not leave the system. Initial latency is defined as the time difference between the request submission and the request service. The average initial latency is defined as $\sum(r_{ser} - r_{sub})$ / N, where r_{ser} is the time required to service a request, r_{sub} is the time at which a request is submitted to the system and N is the total number of requests.

4 Algorithm

A user submits the request using the token (CID, VID, VCR) to the server, where CID denotes the client ID, VID the ID of the requested video and VCR represents that the request is for VCR function. Users with the above token form a queue and wait for service. The requests are fetched from the queue using FCFS queue discipline. Users requesting for videos are classified into *very popular, average popular*, and *least popular*, based on the popularity of the requested video which follows Zipf distribution.

If a user requests for the VCR function, if enough resources are available, then the VCR function is provided to the user using a dedicated channel. The video requested by the user is divided into the fixed number of segments and the dedicated channel is divided into the same fixed number of channels. Each segment is continuously broadcast on each of the specific channels, a user can use VCR function by tuning to a particular channel. However, if sufficient resource is not available, then the user is requested to wait for some time until the resources becomes available or reneging time is reached. Table II shows the algorithm for Hybrid Quality of Service strategy - HQoS. The functions for very popular videos, average and least popular videos, Stream Routine(), FreeUnicastChannnel() are depicted in table III, IV, V and VI respectively.

Table 2: Hybrid Quality of Service Strategy - HQoS

{Input: Request token(CID,VID,VCR)}
{Output: Allocation of different types of channels based on popularity of the video.}

1. Classify the requests into very popular videos, average popular videos and least popular videos.
2. If the request is for very popular videos, then call the function for very popular videos- Table III
   else call the function for average and least popular videos- Table IV.

Table 3: Function for Very Popular Videos

```
Fetch a request from the FCFS Queue
if λ_j > λ_j^max then
  if ( none of the existing regular broadcast currently
  serving video j ) then PID=Null
      RID=FreeBroadcastChannel
   else if ( skew_j > W_j ) then PID=Null
         RID=FreeBroadcastChannel
       else RID=LatestRegularBroadcast
         Workload = Skew_j
         n=number of pid's serving video j
          for (i = 0; i < n; i++)
              PID[i]=LastPatchingChannel[i]
              Workload=Workload-PID[i].Workload
           end for
          PID[i+1]=FreeUnicastChannel(j)
    end if
  update LatestRegularBroadcast=PID
  Start Streaming using Stream Routine (j)
  end if
end if
```

Very Popular Videos: If the arrival rate for a video j with Poisson distribution denoted by λ_j is greater than the upper bound on the average popular videos denoted by λ_j^{max}, then the request is for very popular videos. If the skew for the video j is greater than the patching window, then *FreeBroadcastChannel* is initiated; else the request is patched to the *LatestRegularBroadcast*. The number of channels that are to be used for patching is calculated by *FreeUnicastChannel()* and the requested video starts streaming to the user by using *Stream Routine()*.

Average and Least Popular Videos: If the arrival rate for a video j with Poisson distribution denoted by λ_j is

Table 4: Function for Average popular and Least Popular Videos

```
if λ_j > λ_j^min then
    if ( none of the existing regular multicast currently
    serving video j ) then PID=Null
       RID=FreeMulticastChannel
    else if skew_j > W then
          PID=Null
          RID=FreeMulticastChannel
          else RID=LatestRegularMulticast
            n=number of pid's serving video j
            for ( i = 0; i < n; i++ )
              PID[i]=LastPatchingChannel[i]
              Workload=Workload-PID[i].Workload
            end for
           PID[i+1]=FreeUnicastChannel(j)
           end if
           update LatestRegularMulticast=PID
           Start Streaming using Stream Routine (j)
else
           PID=Null
           RID=FreeUnicastChannel(j)
           Start Streaming using Stream Routine (j)
    end if
end if
```

Table 5: Function for Streaming the Video - Stream Routine (j)

```
RID.transmitted = RID.Transmitted+RID.Bandwidth
Trans_j = Trans_j + RID.transmitted
for (i = 0; i < n; i++)
if (PID[i].transmitted < PID[i].workload) then
PID[i].transmitted=PID[i].transmitted+PID[i].Bandwidth
Trans_j = Trans_j + PID[i].transmitted
else PID[i] = Null
end if
end for
if Trans_j = Len_j then RID=Null
end if
```

Table 6: Function to calculate channels-FreeUnicastChannel(j)

```
if λ_j > λ_j^max then T= Skew Time
    X= (BB/ UB)* T/(2W-T)
else if λ_j > λ_j^min then T=Skew Time
    X=(MB/UB) * T/(2Y-T)
      else X= UB
     end if
end if
return X
```

greater than the lower bound on the average popular videos denoted by λ_j^{min}, then the request is for the average popular videos; else it is for least popular videos. If the skew for the video j is greater than the patching window, then *FreeMulticastChannel* is initiated; else the request is patched to the *LatestRegularMulticast*. The number of channels that are to be used for patching is calculated by *FreeUnicastChannel()* and the requested video starts streaming to the user by using *Stream Routine()*. The least popular video is served using a dedicated channel.

FreeUnicastChannel(): This routine is used to keep track of the bandwidth available. When a very popular or average popular video requires patching, it is necessary to compute the available channels. *FreeUnicastChannel()* is used to find out the number of channels that can be assigned to a patching request.

Stream Routine(): This routine updates the length of video *Transmitted* by the patching and the regular channel. It also modifies the length of video that is to be transmitted (*Trans*) to the user. The process of updating is carried out periodically according to the available bandwidth and the play-back rate of the client.

5 Performance Evaluation

In this section, we evaluate the performance of the Data Sharing strategy (DSS) and compare it with the PQoS. We evaluate the initial latency, blocking ratio, bandwidth utilization and throughput by analysis and simulation. Simulation is carried out in order to evaluate the effectiveness of our algorithm. The video requests follow Poisson process and the popularity of the videos follows Zipf distribution with the parameter of 0.271 [16]. Simulation is carried out for 600 minutes with 100 videos, each video is of 120 minutes duration. The maximum time a client waits for service is 3 minutes. The capacity of each channel is assumed to be 1 Mbps. Maximum number of channels that are available in order to service the request is assumed to be 20. The buffer is managed effectively in the local server. Simulation experiments are conducted on an NT workstation using Visual Basic 6.0 as front end and Oracle 9i as back-end. Visual Basic 6.0 is chosen, as it is an event driven programming language the coded lines of the program are not written and executed in a sequential logic because the client action of clicking a key or check box or button, triggers an event. Since, the event like selection of the category of a video does not occur in a sequential order, it is preferable to use an event-driven programming language.

Fig. 4 shows the number of requests serviced in both the schemes. In earlier stages (till 60 seconds), both the techniques serve the requests and none of them has completed the service. Later, the number of requests served grows linearly in both the schemes. Since recursive patching is

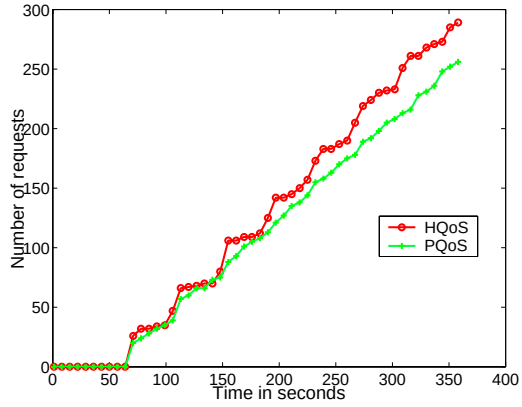

Figure 4: Number of Requests Serviced.

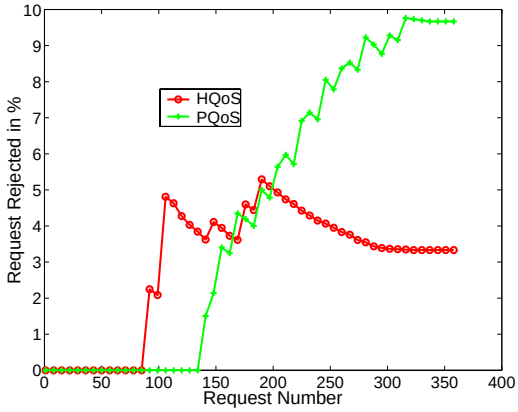

Figure 6: Blocking Ratio.

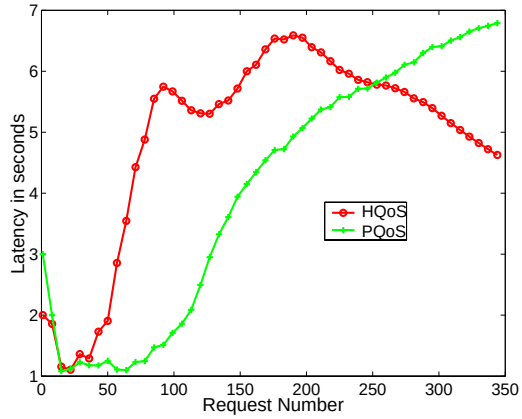

Figure 5: Average initial latency in Secs.

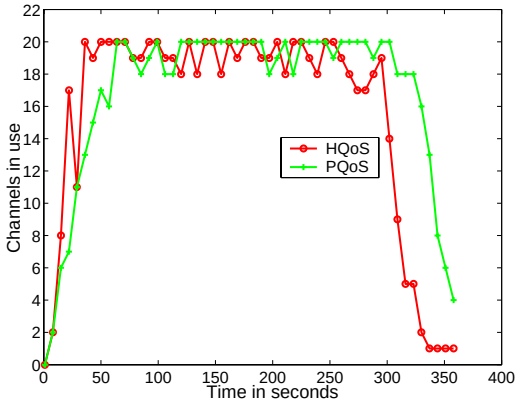

Figure 7: Resource Utilization.

adopted, more number of requests are processed in HQoS strategy and hence achieve significantly greater throughput than that of PQoS. Requests are served faster using the available resources and the resources which enable other requests to be served are released. For example, at time 250, in our scheme almost 250 requests are served when compared to the other scheme where only 200 requests are served which clearly indicates that in our approach more than 20% of the requests are served when compared to PQoS.

Initial latency is a significant parameter from clients perspective which determines the QoS in a VoD system. The performance in terms of initial latency in both the strategies is shown in Fig. 5. Very popular videos are batched and then only serviced, hence incurring initial latency. Majority of requests are for very popular videos, hence initial latency is observed in the graph. It is evident from the graph that in PQoS strategy initial latency always increases. After 250 requests, our scheme shows a significant decrease in initial latency as efficient buffer management technique is implemented in the local server.

Fig. 6 shows the blocking ratio which is the ratio of the number of requests rejected to the total number of requests. This is an important parameter to determine the QoS in a VoD system. This ratio needs to be less in order to provide a suitable QoS. Our scheme has blocking ratio of around 4%, whereas the PQoS strategy has a blocking ratio of 10%.

Fig. 7 shows the utilization of bandwidth in both the schemes. Initially, in both the schemes the resource utilization is not maximum. After 30 requests, all 20 channels are utilized and after 300 secs new requests do not arrive; the existing requests finish the service with pre-allocated channels. Hence, all the channels are not utilized. On an average, in our scheme 90% of the channels are utilized whereas 77% of the channels are utilized in PQoS strategy which demonstrates the superiority of our approach.

6 Conclusions

To maximize the performance of VoD system and to avail the benefits of data sharing techniques, it is crucial to ascertain the amount of resources to be awarded to a particular client. In this paper, we have proposed a plausible data sharing strategy which promises a desirable QoS. The limitations of other designs are (i) they suffer computational overhead for average popular videos as slotted rate is adopted for broadcasting the video, (ii) the resource utilization is not optimum as the patched demands are always served with double rate duration, even if more resources are available. To sweep over these limitations, we have projected a data sharing strategy which is combination of batching and recursive patching with following advantages: (i) the technique of slotted rate is not adopted thus computational overhead is minified, (ii) the resources are efficiently handled by using patching window which ensures that the patched requests are served using available bandwidth, (iii) recursive patching is used for a brief period to broadcast first few minutes of video instead of broadcasting the entire video, hence more batches can be serviced per unit time. The classification model used in data sharing strategy is useful to service providers to fix different pricing schemes for various categories of videos.

References

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

[2] L. Golubchik, C. S. Lui, R. R. Muntz, *"Adaptive piggy backing : a novel technique for data sharing in Video-on-Demand Servers"*, Multimedia Systems, vol. 4, pp. 140-155, 1996.

[3] Kien A Hua, Ying Cai Simon Sheu, *"Patching: A Multicast Technique for True Video-on-Demand Services"*, ACM Multimedia 1998, pp. 191-200.

[4] Murali Kodialam, Steven H. Low, *"Resource Allocation in a Multicast Tree"*, INFOCOM 99, pp. 262-266.

[5] Ariel Orda, Alexander Sprintson,*"A Scalable Approach to the Partition of QoS Requirements in Unicast and Multicast"*, IEEE INFOCOM 2002, pp. 685-694.

[6] Victor Firoiu, Dow Towsley, *"Call Admission and Resource Reservation for Multicast Sessions"*, IEEE INFOCOM '96, pp. 94-101.

[7] Mohammad Riaz Moghal, Mohammad Saleem Mian, *"QoS−Aware Adaptive Resource Management in Distributed Multimedia System Using Server Clusters"*, IEEE Intl. Conference on Cluster Computing, 2003, pp. 238-242.

[8] 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 System Employing Data Sharing Techniques"*, IEEE Trans. on Knowledge and Data Engineering, vol. 14, no. 3, May/June 2002, pp. 508-511.

[9] Kevin C. Almeroth, Asit Dan, Dinkar Sitaram, William H. Tetzlaff, *"Long Term Resource Allocation in Video Delivery Systems"*, IBM Research Report: RC 20249.

[10] Nalini Venkatasubramanian, Klara Nahrstedt, *"An Integrated Metric for Video QoS"*, ACM Multimedia Conference, Nov. 1997, pp. 371-380.

[11] A. Striegel, G Manimaran, *"Managing Group Dynamics and Failures in QoS Multicasting"*, IEEE Communications, June 2002, pp. 249-257.

[12] De-Nian Yang, Wanjiun Liao, Yen-Ting Lin,*"MQ: An Integrated Mechanism for Multimedia Multicasting"*, IEEE Trans. on Multimedia, vol. 3, no.1, March 2001, pp. 82-97.

[13] Murali Kodialam, Steven H. Low, *"Resource Allocation in a Multicast Tree"*, INFOCOM 99, pp. 262-266.

[14] A. Dan, D. Sitaram, P. Shahabuddin,*"Scheduling Policies for an On-Demand Video Server with Batching"*, Proc. ACM Multimedia 1994, pp. 391-398.

[15] W. F. Poon, K. T. Lo, J. Feng, *"Adaptive Batching Scheme for Multicast Video-on-Demand Systems"*, IEEE Trans. on Broadcasting, vol. 47, no. 1, March 2001, pp. 66-70.

[16] Kien A Hua, Ying Cai Simon Sheu, *"Patching: A Multicast Technique for True Video-on-Demand Services"*, ACM Multimedia 1998, pp. 191-200.

[17] Ying Cai, Kein A. Hua, *"Sharing Multicast Videos Using Patching Streams"*, Multimedia Tools Applications, 2003, pp. 125-146.

[18] S. Sen, L Gao, D. Towsley, *"Optimal Patching Schemes for Efficient Multimedia Streaming"*, Proc. IEEE NOSSDAV, June 1999, pp. 255-263.

[19] Ramesh Yerraballi, Xiaoru Zhao, Jasmin Kanabar, *"A New Asynchronous Hybrid Mechanism for Video-on-Demand"*, IEEE Conference on New Waves in System Architecture, 2003, pp. 230-238.

D. N. Sujatha is a Ph.D. student in Computer Science at Bangalore University. Currently she is working as Assistant Professor and Head, Department of M. C. A, B. M. S College of Engineering, Bangalore. She received Bachelors degree in Science and Masters degree in Computer Applications from the University of Mysore in the year 1988 and 1991 respectively. She is a member of ACM. Her research interests are multimedia applications like Video-on-Demand systems, distributed systems and mobile networks.

Girish K received his B.Sc. degree in Electronics from the University of Mysore in 2001, Masters degree in Computer Applications from the Visvesvaraya Technological University in 2004. He is currently working as Lecturer, Department of M.C.A, B.M.S College of Engineering, Bangalore. He is a member of IEEE. His research interest includes computer networks, multimedia applications and sensor networks.

K. R. Venugopal obtained his Bachelor of Technology from University Visvesvaraya College of Engineering in 1979. He received his Masters degree in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph.D. in Economics from Bangalore University and Ph.D. in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored several books on Computer Science and Economics, which include Petrodollar and the World Economy, Programming with Pascal, Programming with FORTRAN, Programming with C, Microprocessor Programming, Mastering C++ etc. He has been serving as the Professor and Chairman, Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He has over 110 research papers to his credit. His research interests include computer networks, parallel and distributed systems and database systems.

L M Patnaik is a Professor since 1986 with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. During the past 35 years of his service at the Institute. He has over 400 research publications in refereed International Journals and Conference Proceedings. He is a Fellow of all the four leading Science and Engineering Academies in India; Fellow of the IEEE and the Academy of Science for the Developing World. He has received twenty national and international awards; notable among them is the IEEE Technical Achievement Award for his significant contributions to high performance computing and soft computing. His areas of research interest have been parallel and distributed computing, mobile computing, CAD for VLSI circuits, soft computing, and computational neuroscience.