M.Sc. Engg. Thesis

Multicast Video-on-Demand Service in Enterprise Networks with Client Assisted Patching

by

S. M. Farhad

Submitted to

Department of Computer Science and Engineering
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

September 2007

The thesis titled "**Multicast Video-on-Demand Service in Enterprise Networks with Client Assisted Patching,**" submitted by S. M. Farhad, Roll No. 040405056P, Session April 2004, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on September 16, 2007.

Board of Examiners

1. [illegible] Dr. Md. Mostofa Akbar Associate Professor Department of CSE BUET, Dhaka 1000	Chairman (Supervisor)
2. M Masroor Ali. Dr. Muhammad Masroor Ali Professor & Head Department of CSE BUET, Dhaka 1000	Member (Ex-officio)
3. [illegible] Dr. M. Kaykobad Professor Department of CSE BUET, Dhaka 1000	Member
4. [illegible] Dr. Md. Humayun Kabir Assistant Professor Department of CSE BUET, Dhaka 1000	Member
5. [illegible] Dr. Mohammad Zahidur Rahman Associate Professor Department of CSE Jahangir Nagar University, Savar, Dhaka	Member (External)

Candidate's Declaration

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

[illegible]

S. M. Farhad
Candidate

Contents

List of Figures

List of Tables

Acknowledgments

All praises due to Allah, the most benevolent and the most merciful.

I express my heart-felt gratitude to my supervisor, Dr. Md. Mostofa Akbar for his constant supervision of this work. He helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. His patient hearing of my ideas, critical analysis of my observations and detecting flaws (and amending thereby) in my thinking and writing have made this thesis a success.

I would also want to thank the members of my thesis committee for their valuable suggestions. I thank Professor Dr. Muhammad Masroor Ali, Dr. Md. Humayun Kabir, Dr. M. Kaykobad and specially the external member Dr. Mohammad Zahidur Rahman.

In this regard, I remain ever grateful to my parents who always inspire for every steps of my life. I also remember my two brothers for their consistent support in my everyday living.

Abstract

Multicast Video-on-Demand (VoD) systems are scalable and cheap-to-operate. In such systems, a single stream is shared by a batch of common user requests. In this research, we adapt multicast communication technique in an Enterprise Network where multimedia data are stored in distributed servers. We consider Batching with Patching. Batching incurs service latency but increases the possibility of larger group formation. Patching eliminates this service latency and further improves the performance of multicast communication in VoD systems. However, in conventional patching, each shortly requested client requires a separate server stream to patch the missing portion. We consider a novel patching scheme called *Client-Assisted Patching* which outperforms conventional patching. Clients' buffer of a multicast group can be used to patch the missing portion of the clients who will request the same movie shortly. The newly admitted client in turn can also provide patching stream to the later client. This scheme significantly reduces the server load without requiring larger client cache space than conventional patching scheme. The conserved server bandwidth can be used to serve more client requests. Simulation experiments show that our scheme is more scalable and cheap-to-operate than conventional patching.

Chapter 1

Introduction

A typical Video-on-Demand service allows a customer to select and enjoy any one movie from a large collection of movies at any time. Typically, these video files are stored in a set of central or distributed video servers, and transmitted through high speed communication networks to geographically-dispersed clients. Upon receiving a client's service request, a server delivers the video to the client as an isochronous video stream. Each video stream can be serviced at the cost of resource reservation both from a particular server and from the particular network path. Thus, sufficient storage-I/O bandwidth and sufficient communication bandwidth on the path of the network from the server to the client must be available for continuous transfer of data before accepting a client's request.

VoD service has a wide variety of applications, such as home entertainment, digital video library, movie-on-demand, distance learning, tele-shopping, news-on-demand, and medical information service.

1.1 Characteristics of VoD Service

The characteristics of the VoD service are described in [24]. They are like the followings:

- The VoD system should support long-lived sessions. A typical movie-on-demand service usually lasts 90-120 minutes.

- High bandwidth requirements; for example server storage I/O and network bandwidth requirements are 1.5 Mbps (3-10 Mbps) for MPEG-1 (MPEG-2) stream.
- The VoD should have a support of the VCR like functionality. The client is able to pause, rewind and fast-forward.
- The quality of service (QoS) guarantees that VoD consumers and service providers should care includes service latency, jitter, defection rate, interactivity, playback effects of videos, etc.

1.2 Unicast and Multicast VoD Techniques

Conventional true VoD (TVoD) systems use one dedicated channel for each service request, offering that client the best TVoD service (unicast service). However, such a system incurs very high costs, especially in terms of storage-I/O and network bandwidth. Moreover, such a VoD service is not scalable and cost effective. Such a dedicated approach quickly exhausts the network bandwidth and the server I/O bandwidth. In order to support a large population of clients, we need new solutions that efficiently utilize the server and network resources.

The popularity of videos plays an important role in determining the effectiveness of a video delivery technique. This is because different videos are requested at different rates and at different times. Videos are usually divided into hot (popular) and cold (less popular) [24], and requests for the top and recently released movies are known to be higher than the less popular movies.

Thus, requests by multiple clients for the same video arriving within a short time interval can be batched together and serviced using a single stream. This is referred to as batching. The multicast facility of modern communication networks offers an efficient means of one-to-many[1] data transmission. The key idea is to avoid transmitting the same packet more than once on each link of the network by having branch routers duplicate and then send the packet over multiple downstream branches. The VoD service in multicast communication is called near VoD service (NVoD). The reasons that multicast can significantly improve the VoD performance are as follows:

- Alleviates the workload of the VoD server and improves the system throughput by batching requests.
- Reduces the required network bandwidth significantly, thereby decreasing the overall network load.
- Offers high scalability which, in turn, increases the system capacity to house large number of clients.
- Provides considerable cost/performance benefits.

1.3 Problems with Multicast VoD

In spite of several advantages multicast VoD service has some requirements. They are described below:

Efficiency: The system should use minimum resources to satisfy a given number of users. The system should impose a minimal additional burden on the server and the network, and should sufficiently utilize critical resources on the server and the network.

Scalability: The system should scale well with the increasing number of customer requests.

Real-Time: The system should respond to the user request and transmit the requested video in real time.

[1]1Multicast also covers multipoint-to-multipoint communication, but for the purpose of this work, it is sufficient to consider only one-to-many communication.

Interactivity: The system should support the VCR like interactivity. The user should be given full control of interactivity.

Reliability: The system should be high fault tolerant to failures both in the server and in the network, and also easy to recover from failures. The transmission of messages and video streams should also be reliable.

Security: The system should provide efficient support for copyright protection in transmitting video streams to multiple clients.

Ability to deal with heterogeneity: The system should deal with heterogeneous networks and CPEs.

Heterogeneous service: The system should have heterogeneous service support to the consumers.

Fairness: The system should provide fair scheduling policy so that a customer is not starved again and again.

In order to satisfy the above requirements we must solve the key problems. The first problem is how to deal with the coupling between system throughput and batching interval. Increasing batching interval can save server and network bandwidths significantly at the expense of increasing the chance customers' reneging. Consumers are likely to renege if they are forced to wait too long, whereas shortening their waiting time will diminish the benefits of multicast session to serve as many consumers as possible.

The second problem is how to support scalability and interactivity. Support for full interactivity requires an individualized service for each customer by dedicating an interaction channel per customer, which limits the scalability of multicast VoD. Again, the heterogeneous service also demand distinguished bandwidth requirement for each QoS service. Hence, the possibility of forming group of same requests also diminishes.

The third problem is how to guarantee customers' QoS with limited bandwidths. In multicast VoD, customers' QoS can be expressed in terms of the waiting time before receiving service, the customers' defection rate due to long waits, and the VCR action blocking probability and playback effect. However, since the system resources are limited, we must strive to maximize their utilization.

Moreover, the multicast VoD service generally prefers popular videos, but how to serve the requests for unpopular videos in a multicast VoD framework is also important to the fairness of service.

1.4 Problem Definition and Previous Work

The major challenge in providing a VoD service is handling enormous demand for high quality video in a real-time network with real time interaction. In existing architectures of VoD systems, customers are served individually by allocating and dedicating a transmission channel and a set of video server resources. The problem with this approach is that it leads to an expensive-to-operate system and moreover such system cannot scale well as the number of clients of the system grows.

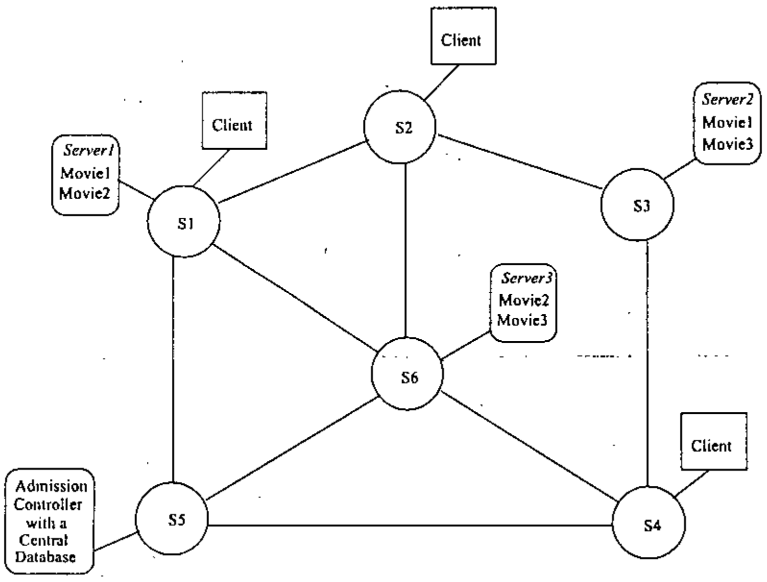


Figure 1.1: The architecture of a distributed system

Video-on-Demand service in an Enterprise Network using unicast communication technique has been discussed in [19]. The architecture of this VoD system is presented in Figure 1.1. In this architecture, there are several storage servers and a powerful *Admis-*

sion Controller with a central database. The clients put their requests of a specific movie to the Admission Controller in terms of *Service Level Agreement* (SLA). The Admission Controller works as a central resource allocator that allocates and dedicates the network bandwidth and the sever I/O bandwidth to the client when his/her multimedia session starts. It also works as a selector of optimal number of users along with their delivery routes and media source. Each prospective client provides a revenue offer for each level of QoS for the multimedia service. Admission control will be done based on the offered revenue, available and required resources. Revenue maximizing problem considering these resource constraints is mapped to *Multiple Choice Multiple Dimension Knapsack* (MMKP) problem. A heuristic solution to this problem is also proposed in [19]. The system offers TVoD service nevertheless it is non-scalable and expensive-to-operate.

One of the best ways to deliver information to more than one customer is through the use of multicast communication. Multicasting VoD (MVoD) systems supporting different forms of VCR actions: *continuous* and *discontinuous* VCR actions are examined in [4, 5, 12, 24, 28]. Most of the works found in the literature considered the *star-bus* topology as the network architecture even though the architecture of typical Enterprise Network is often the *mesh* topology. Star-bus topology is shown in Figure 1.2. In multicast communication technique, a single server stream is required to deliver media data to a group of customers. So, the gain of multicast technique mainly depends on the larger multicast group formation.

Customer requests arriving within a short time can be batched together and serviced by a single stream is called *Batching* [12]. Batching increases the system throughput by increasing the possibility of larger multicast group formation. This is because when the batch duration is introduced it is more likely that more similar requests will be accumulated this short interval of time. However, it increases initial *service latency* that may cause some impatient customers to renege.

Patching eliminates the service latency imposed by the Batching scheme [17]. The objective of Patching is to substantially improve the number of requests each channel can

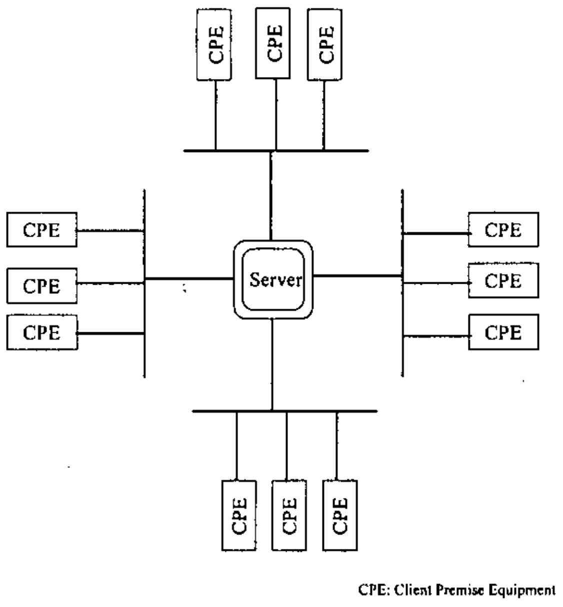


Figure 1.2: The architecture of a typical VoD system.

serve per time unit, thereby sufficiently reducing the per-customer system cost. In Patching scheme channels are often used to patch the missing portion of a service or deliver a patching stream, rather than multicasting the video in its entirety. In Patching, a client might have to download data from both regular multicast and patching channels simultaneously. However, Patching temporarily puts a heavy load on the servers as patching streams are dedicated to the patched clients. We propose a new patching technique which eliminates the heavy server load temporarily incurred by conventional Patching technique.

Cooperative Patching approach is recently proposed approach that relies on the cooperation of the video clients in forming an overlay network over which the video is propagated [14, 15, 20]. In this approach, a client currently in the overlay network forwards the content it is receiving and serves other client's request as a server. Thus the forwarding capability of the overlay network will grow incrementally. However, this scheme puts larger responsibility on clients making the system more prone to service interruption especially in a highly interactive environment.

A client in the system should have a significant amount of buffer space to forward the entire video to other clients. Moreover, it is a *periodic multicast* technique which is only suitable for popular movies. The client buffer can be used to forward the initial portion of a movie to patch a client into an ongoing session. This obviously ease the burden of forwarding the entire movie to other clients and also decrease service interruptions in the system. Thus patching streams will be provided by the clients rather than the server itself. The new scheme is proposed in our research and it is defined as *Client-Assisted Patching.* Incorporating multicast communication technique in Enterprise Network is very much necessary. Such incorporation will make the system highly scalable and increase cost-performance benefits. However, such a system will provide NVoD. Some of the features of the unicast service like high support of interactivity are sacrificed in order to achieve scalability and cost-effectiveness.

1.5 Scope and Focus of the Thesis

In multicast communication service, a single stream may be used to serve a group of clients. Allocation of resources such as, link bandwidth and server I/O bandwidth is essential to ensure Quality of Service (QoS) of multimedia services delivered over the network. The aim of this research is to address this problem in an Enterprise Network with a set of media servers.

This thesis represents a solution of VoD service aided with multicast communication technique in an Enterprise Network. There will be a set of media servers in the Enterprise Network. We analyze the methodologies to handle the interactive requests in the proposed system and also suggest a new technique in admitting VCR requests.

We combine both *Batching* scheme and a new *Patching* scheme in the proposed system. The new patching technique is proposed in this thesis called Client-Assisted Patching. In this patching technique each client is assumed to be cooperative and can serve other clients by dedicating a patching stream who request the same movie shortly after the session begins.

This research also represents an *Admission Controller* for an Enterprise Network. An Admission Controller is out of scope of this research. Clients requests are made to the Admission Controller and bandwidth requirement for these requests is ignored compared to the bandwidth requirement for actual data transmission. It is also assumed that a central database will keep all network and server resource information but the actual multimedia data stream will be kept only on the servers with possible replications of only popular movies. The media servers and the underlying network are assumed to be able to reserve resources for admitted clients' sessions. We also adopt a profit maximization and fair policy.

A simulation based study is required to measure the performance of the system and also compare with other conventional systems. However, it will not design to demonstrate all the communication protocols required for the proposed system and development of the system is out of the scope of this research.

1.6 Outline of the Thesis

This thesis describes the detail design and implementation issues of a scalable video-on-demand service in an Enterprise Network based on multicast communication technique which outperforms similar approaches. The focus of the contribution is on designing a cooperative client based efficient patching scheme. This introductory chapter summarizes the contribution by depicting focus and related works.

The details of video-on-demand service has been illustrated in Chapter 2. A literary review of Cooperative client approach and conventional Patching approach has been presented. This chapter also includes preliminary description of some terminologies and concepts of video-on-demand services. Different languages and tools supporting discrete event simulation are described in this chapter as well.

Chapter 3, the main chapter of this dissertation, illustrates our proposed patching technique and various adaptation techniques in Enterprise Network. A detail description of an Enterprise Network and its components has been given. The admission control methodologies and client buffer requirement analysis are also presented in this chapter. The detail description of interactivity support mechanism of the proposed system has also been illustrated in this chapter.

Chapter 4 consists of the simulation results and comparative study against conventional Patching scheme. This chapter also includes the detail description of simulation settings and different comparative parameters that are used in the simulation process.

Chapter 5 concludes this thesis by summarizing the key contributions and presenting directions towards future research in this field.

Chapter 2

Preliminaries

Video-on-Demand allows clients to connect to a VoD server using a television set-top box (STB) and use the STB to make a selection from the server's video library, and begin watching the selected video after a short interval of time. A workstation can also be used instead of STB. Interactive VoD allows customers to interact with a movie being shown with VCR like functions like pause, fast-forward, rewind and so on. The major challenge of VoD service is to handle the enormous demand of such service in a real time network with real time interaction. In typical VoD system architectures, the approach is to service customers individually by allocating and dedicating a transmission channel and a set of server resources to each customer. The problem with this approach is that it leads to an expensive-to-operate and non-scalable system.

2.1 Video-on-Demand Services

One of the best ways to improve system performance is to satisfy customer requests using delayed server response and multicast communication. Advantages are gained in both the server and the network when several similar requests occur at nearly the same time. A single server I/O and single movie stream can be used to satisfy all the requests accumulated for the same movie within that short time. Customer can share single movie stream resulting in reduced system cost per customer and improved system scalability. This system of sharing resources contradicts, to a certain extent, the need for the individualized service in an interactive VoD system. Providing interactive functions in a multicast VoD system requires either added complexity in the client premise equipment and redefining interactive functions from traditional VCR-style. It may also be necessary to sacrifice some of the on-demand nature of true VoD system. The systems in which interactivity some of the on-demand nature are sacrificed in order to achieve cost-effectiveness or other purposes are sometimes referred to as *Near* VoD systems [22]. The main classification of VoD services are listed and compared in Table 2.1.

Table 2.1: Classification of VoD systems

VoD service	**Comparison**
No-VoD	same as broadcast TV, in which the user is a passive participant and has no control over the session.
NVoD	service latency may occur. Functions like forward, pause and rewind are simulated by transitions in discrete time intervals.
TVoD	the user has complete control over the session presentation. The user has full function of VCR capabilities including change of motion of pictures.

Video data are generally stored in a centralized server or distributed servers. Systems in a distributed environment are more challenging than systems in centralized environment. Most of the research goes for centralized systems. However, some of the works have been done for Enterprise Networks but all most all of them proposed unicast communication technique rather than modern multicast communication technique. In this research we propose multicast VoD system in a distributed environment.

2.2 Scheduling Strategies

Existing schemes for allocating server channels can be broadly classified into *user-centered* and *data-centered* approaches [16, 22, 31]. A user-centered algorithm dedicates channels on behalf of individual users (see [2, 19] and references therein) whereas a data centered algorithm will dedicate channels to video objects rather than to users. This latter approach allows users to share a video stream using the multicast facility of modern communication networks [4, 5] and has the potential of dramatically reducing network and server bandwidth. Thus, the data-centered approach is potentially more scalable than the user-centered approach.

2.2.1 User-Centered Scheduling Strategies

A conventional VoD system is implemented the user-centered scheduling scheme [2, 19] in which a user eventually acquires some dedicated bandwidth. When a client makes a request to the server, the server sends the requested object to the client via a dedicated unicast channel. This system incurs high system costs, especially in terms of server storage I/O and network bandwidths. These techniques are said to be "user-centered", because channels are allocated to users, not data or objects. These simplify the implementation, but dedicating a stream to each viewer will quickly exhaust the resources of the system.

In [2, 19], unicast VoD service has been proposed in an Enterprise Network with multiple servers. We have already discussed it in Chapter 1. Different *Service Level Agreements* (SLA) with batching scheme are considered. There are many definitions of SLA found in [3]. A heuristic solution has been developed to the problem of profit maximization in selecting the optimized client requests over the accumulated requests during a batch interval.

2.2.2 Data-Centered Scheduling Strategies

The solution of the user-centered scheduling policies is data-centered scheduling strategies which dedicates channels to video objects, instead of users. It allows users to share a server stream by batching their requests. Requests by multiple clients for the same movie arriving within a short time interval can be batched together and served by using a single stream [12].

The data-centered strategy significantly reduces the demand for both network and server bandwidths. The data-centered strategies can be either *server-initiated* or *client-initiated* [13]. In the client-initiated service, channels are allocated among the users and the service is initiated by clients, so it is also known as *scheduled* or *client-pull* service. In the server-initiated service, the server channels are dedicated to individual video objects, so it is also called *periodic broadcast* or *server-push* service. Popular videos are broadcast periodically in this scheme, and a new request dynamically joins, with a small delay, the stream that is being broadcast. In practice, it is efficient to use hybrid batching that combines the two schemes mentioned here.

2.3 Server-Initiated Multicast Schemes

In server-initiated scheme, the bandwidth is dedicated to video objects rather than to users. Videos are decomposed into segments which are then broadcast periodically via dedicated channels. Although the worst-case service latency experienced by any subscriber is guaranteed to be less than the interval of broadcasting the leading segment and is independent of the current number of pending requests, this strategy is more efficient for popular videos than for unpopular ones due to the fixed cost of channels.

One of the earlier periodic broadcast schemes was the *Equally-spaced interval Broadcasting* [12]. Since it broadcasts a given video at equally-spaced intervals, the service latency can only be improved linearly with the increase of the server bandwidth. To significantly reduce the service latency, *Pyramid Broadcasting* (PB) was introduced in [1]. In PB, each video file is partitioned into the segments of geometrically-increasing sizes, and the server capacity is evenly divided into K logical channels. The i-th channel is used to broadcast the i-th segments of all videos sequentially. Since the first segments are very small, they can be broadcast more frequently through the first channel. This ensures a smaller waiting time for every video. Some other works [16, 21, 31] are also discussed in the literature to address different issues of periodic multicast VoD services. *Cooperative Client* approach has been discussed in [14, 15]. We want to discuss this approach in the following section as it is related to our research.

2.3.1 Cooperative Client Approach

Cooperative client approach is recently proposed approach that relies on the cooperation of the video clients in forming an overlay network over which the video is propagated [14, 15]. In this approach, a client currently in the overlay network forwards the content it is receiving, and serves other client's request as a server. Thus the forwarding capability of the overlay network will grow incrementally. Each newly admitted client will bring an extra bandwidth capacity to the system.

A video server S periodically broadcasts video program in C channels after a certain time interval d. Each channel can be used to transmit one video stream. There is an application layer multicast tree associated with each channel. The server serves the clients with the original stream, and m time-shifted streams. The streams are labeled as $s_0, s_1, \cdots, s_m$. Stream s_0 is the original stream, while s_i starts after a $i \times d$ delay. Video server is the single source of the video content, and is the root of the application layer multicast tree. It processes client requests to join, leave, and rejoin the multicast group, and is responsible for maintaining the topological structure and resource availability of the multicast tree.

When a client first joins the multicast group, it always joins a multicast tree of the original stream. If the server has free video channel available, the client connects to the server directly. Otherwise, the client joins the tree by connecting to a client already in the tree who has enough available bandwidth resources, while at the same time, has the shortest overlay path to the video server. The cooperating client is called *patching parent* of the other client. A client in the multicast tree suffers service disconnection in two cases: 1) upstream link congestion, or 2) an ancestor node's failure. Patching parent selection algorithms are discussed in [15].

Thus, the streaming problem from a single server to a large number of clients is solved in this work. But the system relies extensively on client cooperation which might create significant service interruption in the system.

2.4 Client-Initiated Multicast Schemes

In this thesis, we mainly concentrate on client initiated approach. Using a client-initiated multicast, when a server channel becomes available, the server selects a batch to multicast according to some scheduling policies discussed in the next section. Requests for a movie arriving within short time can be batched together and serviced using a single stream is called *Batching* [12]. Customer reneging behavior has been discussed in [12]. To eliminate the service latency, several dynamic multicast techniques have been proposed in [10, 17, 18, 25, 29]. In subsequent section we will describe some of the works related with this research.

2.4.1 Batching Policies

The free channels of the server are made available to the customers according to a policy called scheduling policy on which the system performance is related to. Some of the important scheduling policies are discussed in Table 2.2. The scheduling policies shown in Table 2.2 and some of their variants are discussed in [12, 24]. As MFQLF policy

Table 2.2: Classification of batching polices

Batching Policies	**Working Principle**	**Objective**
Maximum Queue Length First (MQLF)	requests for the video with the largest nnmber of pending requests to serve first.	maximizing the server throughput but unfairness to unpopular videos.
First-Come-First-Served (FCFS)	the request with the longest waiting time to serve next.	Fairness but a lower system throughput.
Maximum Factored Queue Length First (MFQLF)	the pending batch with the largest size waited by the factor, (the associated access frequency)$^{-1/2}$, to serve next.	a throughput close to that of MQLF without compromising fairness.

maximizes the system throughput without compromising fairness, we have adopted this policy in this work.

2.5 Dynamic Multicasting

Multicast session tree can be dynamically expanded after session starts. Such schemes save the resources that are otherwise required to serve the clients who request the same movie shortly afterwards. Many works on dynamic multicast have been discussed in the literature. However, we will discuss some of the related works.

2.5.1 Patching

Patching was introduced in [17] in order to eliminate the initial service latency of the clients. Patching increases the number of requests each channel can serve per time unit and decreases service cost. In Patching scheme, channels are often used to patch the missing portion of a service or deliver a patching stream, rather than multicast the video in its entirety. The Figure 2.1 illustrates the Patching scheme. At time zero a multicast session starts and at time five a new request for the same video arrives and it is patched by a dedicated separate channel from the server.

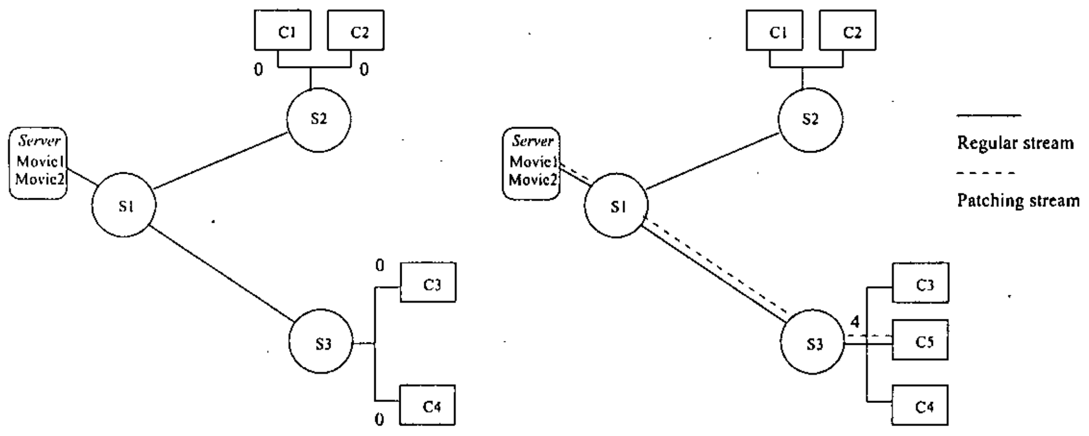


Figure 2.1: Patching: A dynamic multicasting technique.

Given that there is an existing multicast video, when to schedule another multicast for the same video is crucial. The time limit, up to when a newly arrived client will be patched after a multicast session starts, is called *patching window* [8, 17]. Some modifications of patching technique are discussed in [9, 25]. Two simple approaches of setting the patching window are discussed in [17]. The first one uses the length of the video as the patching window. That is, no multicast is initiated as long as there is an in-progress multicast session for the video. This approach is called the *Greedy Patching* because it tries to exploit an in-progress multicast as much as possible. However, over-greed can actually reduce data sharing [17]. The second approach, called the *Grace Patching*, uses a patching stream for the new client only if it has enough buffer space to absorb the skew. Hence, under Grace Patching, the patching window is determined by the client buffer size.

In conventional patching scheme there is a problem of server load mainly experienced at patching time. Server spares a patching stream to each client who will join dynamically to the on going session. This significantly increases server load. This issue has been discussed in [18]. One of the objectives of this thesis is to alleviate this server load. In this case we propose a new patching technique which greatly decreases the demand for the server load.

2.6 Range Multicast

Range Multicast is a new communication paradigm for video-on-demand applications [18]. This scheme is a shift from a conventional thinking about multicast where every receiver must obtain the same data packet at all times. In Range Multicast environment, RM-enabled nodes are placed on the Internet and forms an overlay topology shown in Figure 2.2. As video stream passes through a sequence of nodes on the delivery path each caches the video data in a fixed-sized FIFO buffer. The *root* node does not need to cache any movie because it is directly connected to the *server*. A client requests a video to its nearby RM router called *representative router.* In Figure 2.2 R_2, R_3, R_6 and R_8 are representative routers. The representative router then broadcasts a find request to the overlay nodes if it does not posses the cache of that movie. A RM router which has the cached copy of the initial part of that movie, responds to the broadcast request. The representative router then sends ACK to the earlier response. Thus the client gets the movie.

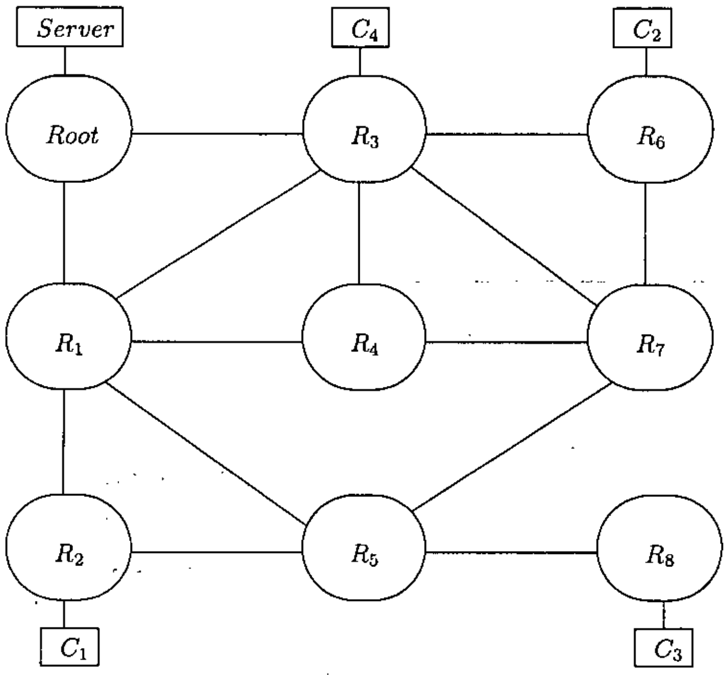


Figure 2.2: Overlay topology of range multicast enabled routers.

2.7 Interactivity in Multicast VoD System

There are two types of VCR actions one is continuous and another is discontinuous [5]. Continuous VCR actions are those in which customer can pause, rewind and fast forward for any discrete time interval. In discontinuous VCR actions customer can pause, rewind and fast forward for integer multiple of predetermined time increment, τ. Whenever a client issues a VCR request, it no longer belongs to the current multicast group. So, the client needs to assign a new multicast group. The details of interactive actions is discussed in the following section.

2.7.1 Actual and Displaced Start Times

In order to understand the discontinuous VCR actions we define the *actual* and *displaced* start times with respect to each customer viewing a movie. The actual start-time defines the time when a customer begins to enjoy a movie. The displaced start-time is defined as the current time minus the duration of the portion of the movie that has already been enjoyed. Initially the actual and displaced start times are the same. After an n time increments of pause request, the n increments are added to the displaced start-time. A rewind (or fast forward) of n time increments subtracts (or adds) n time increments from (to) the displaced start-time (see Figure 2.3). Thus Figure 2.3 represents the fact that any kind of interactive request made by the clients, is regarded as a multicast session change request to the admission controller. Thus, if sufficient resources are available to start a new session with the new start-time, the VCR request is accepted by the admission controller otherwise it is rejected. In case of rejection, the customer continues with the current multicast session and it seems that no interactive request is made. Solution to this problem has been discussed in the following section.

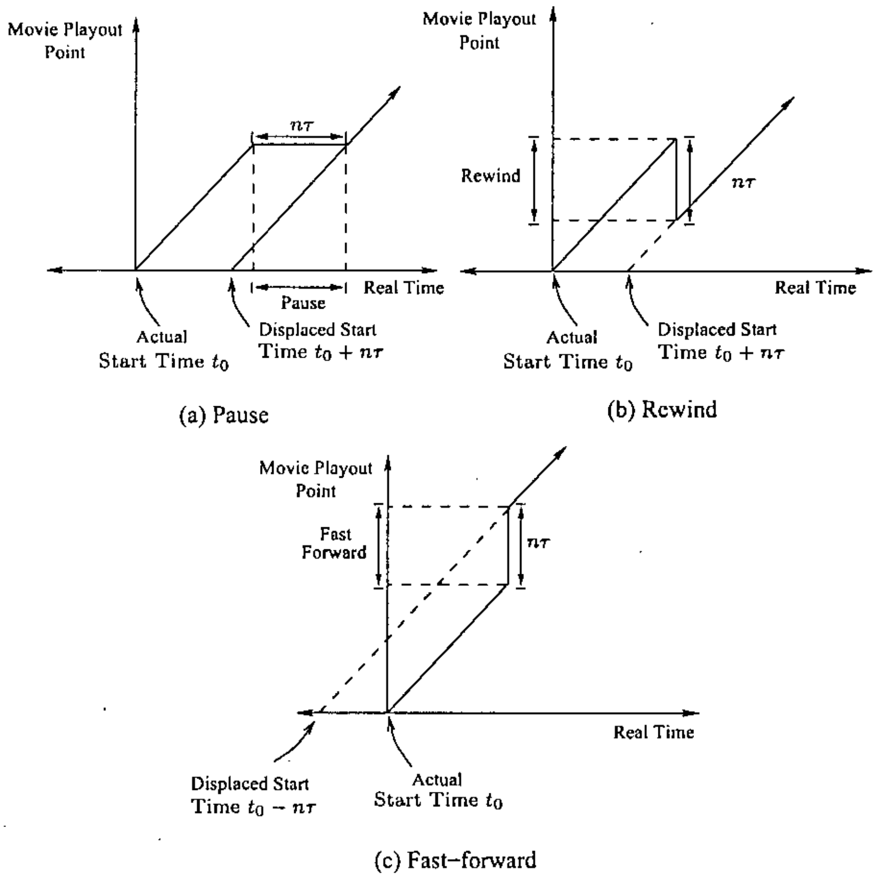


Figure 2.3: Effect of VCR action on a movie's displaced start-time.

2.7.2 Contingency Channel

There is always the possibility that the group change request for VCR action cannot be granted. This will happen when there is no resource available to satisfy the group change request. This scenario is called *VCR action blocking* [4, 5]. There must be some channels reserved to facilitate the VCR actions. These channels are called *contingency channel* or *emergency channel.* They will not be available to satisfy the normal requests. These channels will be used to lower the VCR action blocking rate. In extreme case, when there is no channel available to support the VCR action the request is blocked. There is a trade-off between the number of channels required for initial admission of the customers and the number of contingency channels to support smooth VCR action.

2.8 Network Simulator

In this section, we briefly describe some important simulators that are used in the simulations of different types of networks like wired, wireless and sensor networks.

2.8.1 Parsec

Parsec [6] (for PARallel Simulation Environment for Complex systems) is a C-based discrete-event simulation language and it is a package as well. It adopts the process interaction approach to discrete-event simulation. An object (also referred to as a physical process) or set of objects in a physical system is represented by a logical process. Interactions among physical processes (events) are modeled by timestamped message exchanges among the corresponding logical processes. One of the important distinguishing features of Parsec is its ability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures. Parsec is designed to cleanly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute it. Thus, with few modifications, a Parsec program may be executed using the traditional sequential (Global Event List) simulation protocol or one of many parallel optimistic or conservative protocols. In addition, Parsec provides powerful message receiving constructs that result in shorter and more natural simulation programs. Hence we have implemented our simulation programs with this language.

2.8.2 GloMoSim

GloMoSim [7] is a scalable simulation environment for wireless and wired networks systems developed initially at UCLA Computing Laboratory. It has been designed using the parallel discrete-event simulation capability provided by Parsec. GloMoSim currently supports protocols for purely wireless networks. It is build using a layered approach.

Standard APIs are used between the different layers of the system. This allows the rapid integration of models developed at different layers by users. To specify the network characteristics, the user has to define specific scenarios in text configuration files: app.conf and Config.in. The first file contains the description of the traffic to generate (application type, bit rate, etc.) and the second contains the description of the remainder parameters. The statistics collected can be either textual or graphical. In addition, GloMoSim provides various applications (CBR, ftp, telnet), transport protocols (TCP, UDP), routing protocols (AODV, flooding) and mobility schemes (random waypoint, random drunken).

Chapter 3

Client-Assisted Patching

In conventional Patching scheme, the patching streams are provided by the server. This incurs heavy load to the server as it spares a dedicated channel for each client who requests the same movie after a session starts. The client side cache can be used to store the initial part of the movie and client can supply the patching stream to the later clients to reduce the server load. The conserved server bandwidth can be used to satisfy more multicast sessions and increase the scalability of the proposed VoD system.

3.1 Client-Assisted Patching Technique

To understand the motivation behind Client-Assisted patching, we first discuss the weak point of conventional Patching and then describe the details of the proposed system in this section. We analyze the buffer requirement of the clients of the newly proposed system and also compare it with conventional Patching. However, in the proposed system, each client requires certain buffer space which is only required for patched client in conventional Patching scheme.

3.1.1 Weak Point of Conventional Patching

Although Patching is more efficient than multicast with batching technique, it puts heavy load on server bandwidth. In Patching scheme, if a session starts with multiple clients and after a while a client requests the same movie, then the newly arrived client is merged with the ongoing session. If Patching was not considered, another session is to be created and the movie is to be transmitted in its entirety. Thus Patching increases system throughput. It also eliminates service latency incurred by the Batching scheme. Patching contributes the efficiency gain at the cost of client side cache requirement. However, in patching scheme, server reserves a unicast channel for each client who requests the same movie shortly afterward. This temporarily puts heavy load on servers at the time of getting more and more patchable requests. The problem is illustrated in Figure 3.1.

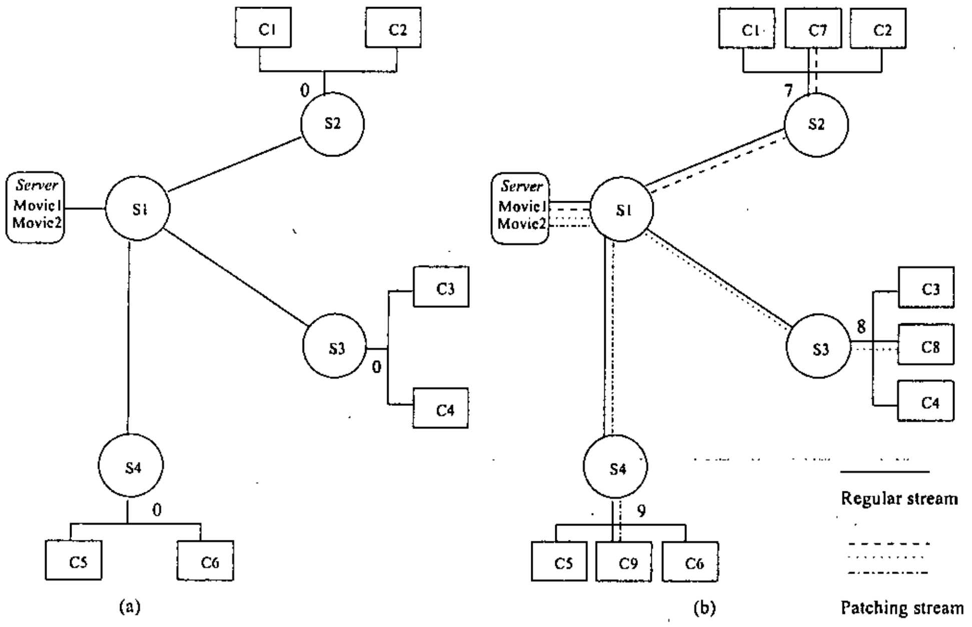


Figure 3.1: Server load in Patching.

A session is started with six clients at Time 0 as shown in Figure 3.1(a). *Patching window* is assumed to be 10 time units long. We have discussed in previous chapter that the patching window is the time limit up to when a newly arrived client will be patched after a multicast session starts. At Time 7, 8 and 9, clients $C7, C8$ and $C9$ request the same movie respectively as shown in Figure 3.1(b). As all the new requests arrive within the time interval of the patching window, patching streams are dedicated from the server for all of them. Different dotted lines in Figure 3.1(b) have been shown as the patching streams from the server. This demands heavy bandwidth requirement for the servers as shown in Figure 3.1(b). Client-Assisted Patching reduces the server load by using the client side cache. In this approach, all patching channels will be provided by the cooperative clients rather than the server itself. Thus, the proposed system alleviates server load and the conserved bandwidth can be used to satisfy more multicast groups. Thus it increases the throughput and scalability of the system.

3.1.2 Technique of the Proposed System

Basic principles of the proposed system is demonstrated in the following points:

- In client-assisted patching, a client will be patched if it requests the same movie that has an ongoing session available within the duration of the patching window. A nearby client of that session will be selected by the Admission Controller to supply the patching stream. The selected nearby client is called the *patching parent* and the requesting client is called *patched child client.*

- Each client of a session will store the initial part of the movie so that each client can supply the patching stream to its nearby client when selected as a patching parent within the time schedule of the patching window. The newly merged client in turn can also supply patching stream to other newly arrived client. So, every newly admitted client will bring some extra bandwidth to the system. However, each client in the proposed system should have at least a certain amount of buffer space to store the initial part of a movie and should have certain I/O bandwidth to forward video data to a patched client.

- In our proposed system, a client will supply patching stream to at best a single client. Thus clients have a limited responsibility in our approach unlike Cooperative Patching [14, 15] where client forwards the whole movie to multiple clients. Thus client management task will be easier and efficient in the proposed system.

- Patched clients may experience *service interruption* in the proposed system. This may happen when a client issues a *VCR request* or *session leave* request while serving as a patching parent. We also propose a solution to this problem. Service interruption and client buffer requirement will be discussed in the subsequent sections.

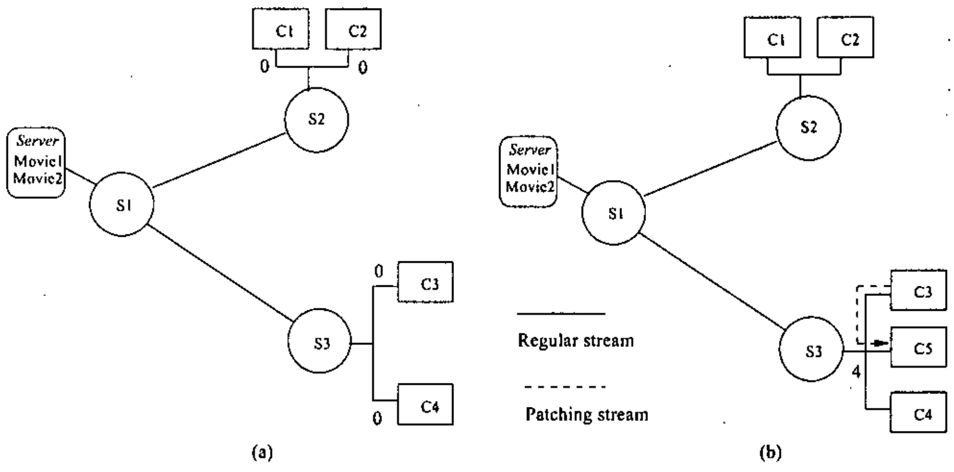


Figure 3.2: Client-Assisted Patching.

Client-assisted patching technique is illustrated in Figure 3.2. A session is assumed to be started at Time 0 with four clients as shown in Figure 3.2(a). Patching window is assumed to be 5 time units long. At Time 4 Client $C5$ requests the same movie. As the request time is within the duration of the patching window, the Admission Controller will select a nearby client to supply the patching stream. At this time, Client $C3$ is selected as shown in Figure 3.2(b) to supply the patching stream to the newly arrived Client $C5$. The patching stream is released when the missing portion is made up and the client will continue with the regular stream until the end of the session.

3.2 Architecture of the Proposed System

The architecture of the proposed video-on-demand service in an Enterprise Network is shown in Figure 3.3. The system components are described below.

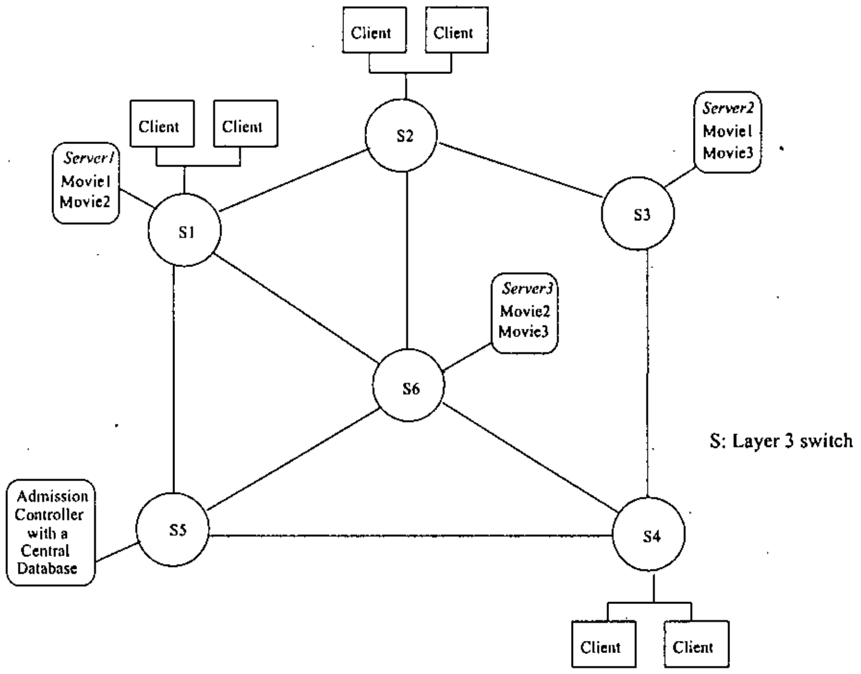


Figure 3.3: The architecture of Enterprise Network

- **Enterprise network:** We consider a privately owned small enterprise network with network switches (nodes) and links between them. In Figure 3.3, $S1$ through $S6$ represent network nodes. We use switches and network nodes interchangeably in this text to interpret the same entity. Links or connections between them are shown by lines between them.
- **Video server:** The video servers stores video streams and delivers the requested movies through multicast channels. First round movies are replicated in different servers in order to satisfy enormous demand for that kind of movies.
- **Client:** A client is connected to the enterprise network through an interface. The interface can be a workstation or a device. The device must be able to send and

receive control messages required to communicate with the Admission Controller, patching parents and servers. The device receives and processes a video stream. It also receives and processes input from the customer via a remote control. The customer can either request a movie or request VCR-style functions such as pause, rewind and fast forward. Buffering a small number of frames in the customer end also helps provide continuous playout in the event of short and unexpected delays in the video server or network.

- **Admission Controller:** The Admission Controller is in charge of accepting or rejecting the clients' requests and acts as a moderator. The Admission Controller maintains a centralized database that contains all necessary information of the system. These include available memory, CPU cycles and I/O bandwidth of different servers and bandwidth of connecting links. Multimedia data information i.e. the where abouts of the media data are also stored in the database. Note that the centralized database does not contain actual multimedia data. The server will send the media data to the clients according to the instruction of the Admission Controller. The Admission Controller also maintains the session related information.

3.2.1 Practical Description of the Architecture

The architecture and its different components have been described in the previous subsection. However, a sensible description of the architecture will help understand the real environment of the system. This will also provide information about the implementation details of the proposed system.

- **Connectivity:** In this architecture there are connections at different levels as shown in Figure 3.4. The switches or nodes form the backbone of the enterprise network. The interconnection between nodes can be established by optical fiber of Gigabits per second (Gbps) range. 1000BaseF fiber optics or 1000BaseT copper wires are typical example of 1 Gbps connectivity available in the market. If necessary the bandwidth of some links can be of 10 Gbps and high end servers with data rate 10Gbps can be attached to these ports. However, when there are few users in the system low capacity connection like Optical Carrier level 1 (OC1) can be used. The workstations are connected through an Ethernet Local Area Network (LAN) or an Asymmetric Digital Subscriber Line (ADSL).

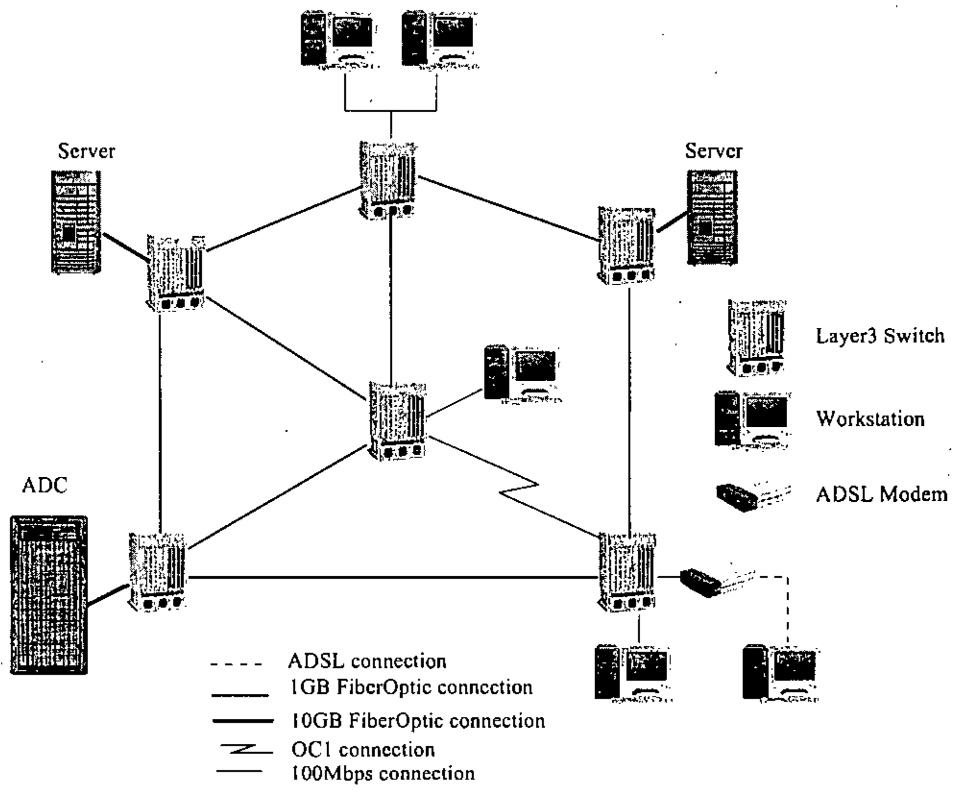


Figure 3.4: The architecture of Enterprise Network

- **Switches:** The switches in the enterprise network should be high speed switches having appropriate number of gigabit ports. The capacity of such switches could be some millions packets per second. However, switches can have some 10GB ports in case of connecting heavy duty servers.

- **Servers:** The servers are special Network Attached Storage (NAS) devices. The servers are dedicated for sending video data. They are directly connected to the switch with gigabit capacity connection.

- **Customer Interface:** The customer interface can be workstation or set-top box. Workstations are standalone personal computers connected to the network through LAN or ADSL. The set-top box is simply a channel tuner and a low cost device. Whatever be the customer interfaces, they should have sufficient resources like memory, disk space, graphics, video data decoding capacity and video software execution capacity. Our proposed VoD system requires the interface to have enough memory to buffer certain amount of video data.

- **Conncetivity and Storage of Admission Controller:** The Admission Controller is a high performance machine. It is directly connected to the high speed switch. A NAS device can be used for the centralized database attached to it. The controller is connected directly to the switch with a gigabit capacity port and gigabit connection media.

3.2.2 Protocols of the System

Although this work is not focused on developing the system for VoD using Client-Assisted Patching, we want to discuss some of the protocols that should be used in different stages of the system. The protocols are mainly related in communication among Admission Controller, clients and servers. We also discuss the protocols that should be used in cooperative clients approach of patching.

The *Real Time Streaming Protocol* (RTSP) is a protocol for use in streaming media systems which allows a client to remotely control a streaming media server, issuing VCR-like commands such as "play" and "pause", and allowing time-based access to files on a server [32]. The control data such as advertisements of different servers about their media data and their resource information, requests to the Admission Controller by the users and also the corresponding notifications, patching requests to the patching parents, all these need guaranteed data transmission. However, bandwidth requirements to transmit all these control data is negligible as compared to the media data transmission.

On the other hand, video playback demands real time data delivery. It requires ordered and timely delivery of packets with no retransmission. It is only possible when all media data are routed through the same path from the same source for any requested video. Moreover, a request has been accepted by the Admission Controller by considering a fixed routing path and a fixed server. So, *Multi Protocol Label Switching* (MPLS) may be used for this strategy in which an end to end virtual path is created in the packet switching network [27]. The Admission Controller sets the MPLS labels in the switches of the enterprise network for a particular path. For video data transmission, the *Real-time Transport Protocol* (RTP) can be used as it defines a standardized packet format for delivering audio and video over the Internet [33]. Then a multimedia session starts transmitting the multimedia stream using RTP over MPLS.

Cooperative clients need to cache the video data received from the server and can serve other clients to patch in an ongoing session. *Cache-and-Relay* protocol can be used in this purpose where clients cache data received from the server and future requests are satisfied by the client which contains the data.[20].

3.2.3 Working Principle of the Admission Controller

Communication takes place among different components of enterprise network at different stages using different protocols. The following points depict the activities of the Admission Controller:

- The servers advertise about the media data stored in their storage to the Admission Controller. They also advertise about their resources to the Admission Controller. Thus the Admission Controller updates its database of the movie stores.

- The clients put their request to the Admission Controller. The request contains the information about the movie to be enjoyed, cache size and bandwidth of the client, source of the client and so on.

- The Admission Controller maintains shortest path multicast tree of nodes of the entire network rooted at each server node. It admits a client's request and allocates resources based on a particular multicast tree. It accepts or rejects any client's request based on the availability of the resources. The detail admission policy is described in the following section. The acceptance reply includes multicast address of the session and patching parent address if required. Eventually the client will be notified about the acceptance or rejection.

- A client may be admitted at the beginning of the session or after a short while from the starting point of a session. In the later case, the client is provided with a patching parent information. The Admission Controller will select a patching parent and also keep track of parent information. The client will download data both from the regular channel and a patching channel. Server provides the regular channel and a patching parent provides the patching channel. If a patching parent leaves the session while serving other client, the Admission Controller will select another patching parent from the participants of that session. This may introduce *service interruption*. We will discuss more about service interruption later in this chapter.

- A client may request interactive VCR action to the Admission Controller at any time during a play back of a movie. We consider pause, fast-forward and rewind interactive functions. Any kind of VCR request is interpreted as a multicast session change request to the Admission Controller. We have already illustrated about multicast session change request due to interactive request in Chapter 2. The Admission Controller will try to assign another session to the client who issues a VCR action. However, interactive requests may be ignored by the Admission Controller due to resource shortages. This is known as *VCR action blocking*. We will describe more about this in the following section.

3.3 Interactivity in the Proposed System

Any kind of interactive request will be regarded as a multicast group change request. So, the customer who requests the interactive VCR action, will be assigned to a new multicast group with a new start time called *displaced* start-time [5]. We have already discussed about interactivity in a multicast VoD system in Chapter 2. We will now focus on the issues of interactive VCR actions in *Client-Assisted* patching scheme. We also discuss two alternatives of incorporating interactivity in the proposed multicast VoD systems.

3.3.1 Incorporating VCR Action by Joining Other Session

When a customer issues an interactive VCR request, a message is sent to the Admission Controller with a new start-time of the movie that is the displaced start-time. It should be noted that any interactive request is regarded as a multicast group change request. When the Admission Controller receives a multicast group change request it determines the number of customers watching the same movie which have the actual start-time that matches with the displaced start-time. This information is used by the Admission Controller to decide whether this request can be granted. The strategy of the Admission Controller is as follows.

- The Admission Controller must decide whether sufficient resources are available to satisfy the interactive action. A request can be granted if any one of the following conditions can be met:

  i. If another session of a multicast group exists playing the same movie that has the actual start-time within the interval [*displaced start-time, displaced start-time* + *threshold*] and the path from the customer to the server serving the group is already included in the multicast tree of the session then the customer can join the multicast group. This will not require any additional resources to serve this client. Here, *threshold* is an integer multiple of predetermined time units. Usually *threshold* can be an integer multiple of *batch interval.* Thus, the client may join in a multicast group which has an actual start-time that does not exactly match the new displaced start-time of the requested customer. This is very much necessary because the possibility of finding an existing session that has an actual start-time within a range is higher than the possibility of finding an existing session that has a particular actual start-time.

ii. If another session of a multicast group exists showing the same movie that has the actual start-time within the interval [*displaced start-time, displaced start-time* + *threshold*] and resources are available for the path from the customer to the server serving that multicast group if the path is not already included with the multicast tree of the session.

iii. If no such group exists, but resources are available to create a new multicast group.

- If the interactive VCR request can be satisfied, the customer who performed the interactive VCR action is switched to the new multicast group. Depending on the number of customers left in the previous multicast group the followings three things may happen:

  i. If the customer who switched to the new multicast group was the only one watching the old stream, the stream is no longer needed and can be deallocated. These resources can be used to satisfy other requests.

  ii. If some other customers exist in the multicast group and there is no other customer along the path from the server to the client of the multicast tree then the resources along the path can be released.

  iii. If some other customers exist in the multicast group and also on the path from the server to the client of the multicast tree then the resources along the path cannot be released.

There is always the possibility that the group change request for interactive VCR action cannot be granted as we have already discussed in Chapter 2. This will happen when there is no resource available to satisfy the group change request. So, in this scheme it is possible to reserve some channels to use in this emergency situation. However, as these channels are not used for admitting initial requests, more initial movie requests are likely to be rejected by the Admission Controller.

3.3.2 Using Client-Assisted Patching in VCR Actions

In this section, we discuss the provision of using the Client-Assisted Patching in the same way as we have discussed for admitting initial client requests. That is, patching the displaced clients (for issuing interactive VCR action) with ongoing sessions by the cooperation of other clients. *Service interruption* may occur in Client-Assisted Patching scheme when the patching parent itself issues an interactive VCR request or terminate the session. We have covered the details of service interruption later in this chapter.

The service interruption problem may also occur at the time of patching initial portion of a movie to the clients. Generally interactive VCR request is issued after the initial part of a movie is played out. Thus, such requests at the beginning of a movie will not create considerable service interruption in the system. However, it might introduce service interruption to a significant number of clients in the system. This due to the fact that the probability of issuing an interactive action by a customer is higher at the later part of a movie. On the other hand, the users issuing VCR action are generally habituated with waiting few seconds (usually a *threshold* amount). For this reason, we propose to use Client-Assisted Patching in a low interactive environment but not in a high interactive environment. We also propose to use conventional Patching to patch the displaced clients to ongoing sessions.

3.4 Admission Control Algorithm

The Admission Controller plays a very important role in the proposed VoD system. In this section we will describe the admission control procedure in detail. The complexity of the proposed procedure is discussed in the following section. The flow chart of the admission control procedure is illustrated in Figure 3.5.

Before describing the admission control algorithm, we first present the requests and replies to and from the different components of enterprise network and some data structures that are necessary to get an overall idea of the algorithm.

3.4.1 Requests and Replies

Each switch, server, session, client, movie are distinguished to the Admission Controller by a unique identifier. Client requests and responses of the Admission Controller that are used in this algorithm are as follows:

- **MOVIE-REQUEST:** Each client will place request for a movie to the Admission Controller through this message. The message contains the requested node id and movie id.
- **VCR-REQUEST:** Clients will put their interactive VCR requests to the Admission Controller through this message. The message contains the source node id, session id and the movie id of the client. The message also specify the type of interactive VCR action like pause, fast-forward, rewind and also the new start-time of the movie due to the VCR action.
- **SESSION-END-REQUEST:** Each client will leave a session by sending this message to the Admission Controller. The message contains the information about the source node id and session id. The message is particularly important as the revenue is calculated based on movie start-time and leave-time.

- **SELECT-PATCH-PARENT:** The child client, facing patch stream loss, uses this message to request the Admission Controller to select an alternate patching parent. The Admission Controller will find out an alternative patching parent and send the parent address to the respective patched child client.
- **MOVIE-ACCEPTED:** This is a response message from the Admission Controller of MOVIE-REQUEST of a client. It contains the multicast address of the accepted session. It also contains the patching parent address if patching is necessary.
- **MOVIE-REJECTED:** Due to the lack of necessary resources the client's movie request will be rejected by the Admission Controller. The message is used to reject any client's MOVIE-REQUEST.
- **VCR-ACCEPTED:** This is also a reply message of the Admission Controller to the VCR-REQUEST of a client. It contains the multicast address of a new session with different start-time due to the shift from the original start-time for such requests.
- **VCR-REJECTED:** When there is no resource available to satisfy the VCR-REQUEST, the Admission Controller will send the message to the requested client.
- **PATCH-PARENT:** The Admission Controller will find out an alternate patching parent in case of current patching parent leaving the session and send the parent address to the corresponding child client through this message.

Necessary data structures and procedures required for admission control algorithm are as follows:

3.4.2 Data Structure

network : This data structure represents the enterprise network for the multicast VoD system. It is a graph with multiple nodes and connecting links. It must contain information about the servers and clients connected with the network.

multicastTrees[ ] : This is an array of shortest path trees rooted at the servers. There is one shortest path tree for each server.

serverSource[ ] : This is an array of nodes to which servers are connected. No more than one server is connected to a node.

batchList[*movie-id*] This is an array of the waiting queues of movies. The client requests that are not satisfied readily by Client-Assisted Patching scheme are queued to corresponding movie queue. The Admission Controller will handle these requests after the batch interval.

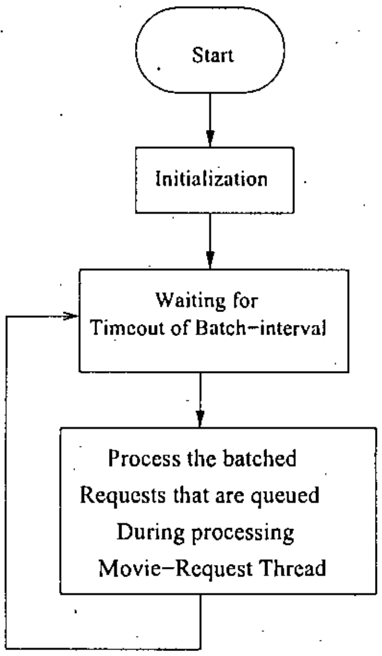
3.4.3 Procedures

The admission control procedure consists of two threads, batched requests processing thread and online requests processing thread as shown in Figure 3.5(a) and Figure 3.5(b) respectively. The details of the procedure and its sub-procedures are described below:

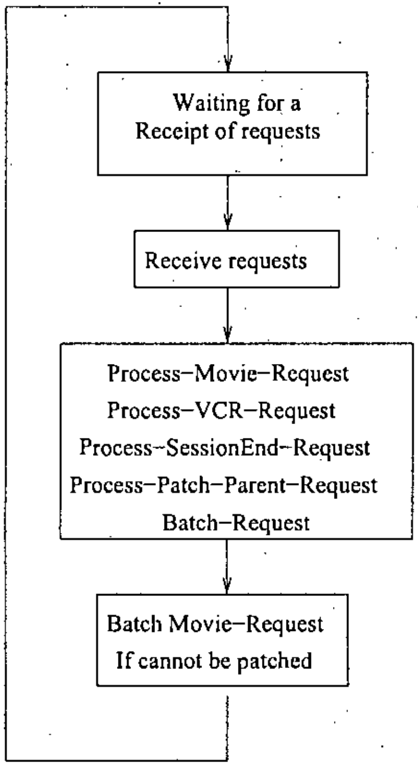
multicastTrees[ ] *dijkstra*(*network*, *serverSource*[ ]) : The procedure returns a list of shortest path trees of the *network* assuming roots are at *serverSource*[ ]. Dijkstra algorithm [11] is used to construct these shortest path trees.

session selectPatchSession(*client*, *movie-id*) : The procedure will find if the client request can be patched to an ongoing session of *movie-id*. If there are more than one patchable sessions then the procedure will return the session which requires least resources for the client to join.

patchingParent selectPatchingParent(*session*, *client*) : The procedure will find out a client from the remaining clients of *session* as a patching parent of *client*. This procedure will be called after procedure *selectPatchSession* is called. If a patchable session and a patching parent are found, the request is said to be patched otherwise the request is queued to *batchList*[*movie-id*].


Start
Initialization
Waiting for Timeout of Batch–interval
Process the batched Requests that are queued During processing Movie–Request Thread

(a) Thread for batch processing of the queued requests


Waiting for a Receipt of requests
Receive requests
Process–Movie–Request
Process–VCR–Request
Process–SessionEnd–Request
Process–Patch–Parent–Request
Batch–Request
Batch Movie–Request If cannot be patched

(b) Online request receiving and processing thread

Figure 3.5: Flow chart of admission control

add(batchList[movie-id], client) : If the movie request of *client* cannot be patched then it will be queued to the array denoted by *batchList[movie-id]* by this function. The decision about the acceptance of the requests of the batches will be made after the batch interval.

session admitClient(time, client, movie-id) : This procedure will admit a client in a multicast session to be started for the current batch of requests. It will create a new session of *movie-id* if there is none and set the actual start-time of the session as *time*. It returns *session* as a session identifier of the multicast session. If there is no resource available in admitting a request, it returns *Nil*. Availability of resources must be checked during this admission.

session admitVCRRequest(displaced-start-time, client, movie-id) : As we have already discussed that a client will be displaced from its original *start-time* whenever it issues a VCR request that is the *displaced-start-time*. This procedure will search for an existing session having the original *start-time* within interval [*displaced-start-time*, *displaced-start-time* + *threshold*] and returns the *session*. If there is not such session and resources are available then a new session will be created. In every case resource availability is checked. We have discussed the details of the procedure in section 3.3.

distance(*shortestPathTree*, C_i, C_k).: The procedure returns the shortest path distance from client C_i to the client C_k in the *shortestPathTree*.

MFQLF(*batchList*[]): This procedure implements the Maximum Factored Queue Length First policy [12, 24]. During a batch interval the requests that cannot be patched are queued to *batchList*[ ] corresponding to *movie-id*. The procedure sorts the array *batchList*[ ] which is an array of batched customer requests. Sorting is done according to the largest batch-size weighted by the factor (associated access frequency)$^{-1/2}$ in descending order. It is worth mentioning that the requests are not sorted in the list of batched requests for each movie. Generally there are limited number of movies in a VoD system. That is way *Bubble Sort* is considered a reasonable sorting technique [11].

```
Procedure Admission-Control
   /* The core procedure used by the Admission Controller */
      Init-Admission-Controller()
      Start Online requests processing
      Start Batched requests processing
end Procedure Admission-Control

Thread Online-Requests-Processing
      while true do
         request ← Receive (requests) /* A non-blocking call to receive a request */
            case (request)
               MOVIE-REQUEST:
                     Process-Movie-Request()
               VCR-REQUEST:
                *Process-VCR-Request*()
            SESSION-END-REQUEST:
                *Process-SessionEnd-Request*()
            SELECT-PATCH-PARENT:
                *Select-Patch-Parent*()
        endcase
    end while
end Thread Online-Requests-Processing

Thread Batched-Requests-Processing
  while true do
    Waiting for Timeout of Batch-Interval
    MFQLF(batchList[ ]) /* Sorting the array of movie batches in descending order */
    for $i \leftarrow 1$ to $m$ do
      for each client $C \in batchList[i]$ do
        $session \leftarrow admitClient(currentTime, C, i)$
        if ($session \neq Nil$) then
          Send MOVIE-ACCEPTED($session$, $Nil$) to $C$ /* Patching parent $Nil$ */
        else
          Send MOVIE-REJECTED to $C$
        endif
      end for
    end for
  end while
end Thread Batched-Requests-Processing

Procedure Init-Admission-Controller(network $N$)
  /* It initializes the multicast trees rooted at each server node */
  /* $N$ is an Enterprise network with its connectivities */
    for $i \leftarrow 1$ to $no\_of\_servers$ do /* Shortest path trees rooted at each server node */
      $multicastTrees[i] \leftarrow dijkstra(N,\ serverSource[i])$
    end for
end Procedure Init-Admission-Controller

Procedure Process-Movie-Request(client : $C$, movie-id : $M$)
  /* The procedure will be invoked when MOVIE-REQUEST message has been accepted */
      $session \leftarrow selectPatchSession(C,\ M)$
      $patchingParent \leftarrow selectPatchingParent(session,\ C)$
      if ($session \neq Nil$ and $patchingParent \neq Nil$) then /* Client C is patched */
        Admit $C$ in session /* Allocation of resources for Client C */
        Send MOVIE-ACCEPTED($session$, $patchingParent$) to $C$
      else /* If patching is not possible then batch the request */
        $add(batchList[M],\ C)$ /* Client C is batched */
      endif
end Procedure Process-Movie-Requests

Procedure Process-VCR-Request(client : $C_j$, session : $S$, movie-id : $M$, time : newStartTime)
  /* The procedure will be invoked when VCR-REQUEST message has been accepted */
      $session \leftarrow admitVCRRequest(newStartTime,\ C_j,\ M)$
      if ($session \neq Nil$) then /* If resources are available */
        if ($C_j$ is serving as patchingParent of client $C_k$) then
          $C_n \leftarrow selectPatchingParent(session,\ C_k)$ /* Managing patching parent */
          if ($C_n \neq Nil$)
              Send PATCH-PARENT($C_n$) to $C_k$
          else /* Service interruption occurs for $C_k$ */
```

```
                Send PATCH-PARENT($Nil$) to $C_k$
            endif
        endif
        Send VCR-ACCEPTED($session$) to $C_j$
        $releaseResource(S,\ C_j)$ /* Release resources from the previous session */
    else /* Ignoring VCR request if resources are not available */
        Send VCR-REJECTED to $C_j$
    endif
end Procedure Process-VCR-Requests

Procedure Process-SessionEnd-Request($client:\ C_j,\ session:\ S,\ movie\text{-}id:\ M_l$)
  /* The procedure will be invoked when SESSION-END-REQUEST has been accepted */
    if ($C_j$ is serving as patchingParent of client $C_k$ ) then
        $C_n \leftarrow selectPatchingParent(session,\ C_k)$ /* Find patching parent for $C_k$ */
        if ($C_n \neq Nil$)
            Send PATCH-PARENT($C_n$) to $C_k$
        else /* Service interruption occurs for $C_k$ */
            Send PATCH-PARENT($Nil$) to $C_k$
        endif
    endif
    $releaseResource(S,\ C_j)$ /* Release resources from the session */
end Procedure Process-SessionEnd-Requests

Procedure Select-Patching-Parent($network\ N,\ session\ S,\ client\ C_k$)
returns $patchingParent$
    if $C_k$ is the only client for $S$ then /* If only child client left in the session */
        return $server\ of\ S$ /* Returns the regular channel of the server */
    else
```

```
        $newTree \leftarrow dijkstra(N,\ source\ node\ of\ C_k)$
        $min \leftarrow \infty$
        for $each\ client\ C_i\ of S$ do /* Finding shortest path parent */
            $d \leftarrow distance(newTree,\ C_i,\ C_k)$
            if $C_i$ $not\ serving\ other\ client$ and $min > d$ then
                $min \leftarrow d$
                $C_j \leftarrow C_i$
            endif
        end for
        return $C_j$ /* Returns $C_j$ as patching parent of $C_k$ */
    endif
end Procedure Select-Patch-Parent
```

3.4.4 Complexity Analysis

Suppose,

Number of servers = s

Number of movies = m

Number of nodes = V

Number of edges = E

Number of clients that are seeking admission = n_r

Number of clients already admitted in multicast session = n_c

The ratio, $q = n_r/n_c$

Average movie duration = L

Batch interval = d

The maximum number of multicast session in the system = $\frac{L}{d} \times m$

Total server bandwidth $= B$

The maximum server bandwidth required to serve each session $= b$

The average server bandwidth required to serve all sessions, $S_r = m \times \frac{L}{d} \times b$

The total bandwidth of the accepted sessions can be at most, $S_a = B$

Thus, the average percentage of served requests can be estimated as, $\frac{S_a}{S_r} = \frac{B}{m \times (L/d) \times b}$

Thus, the average percentage of served requests, S_a/S_r is inversely proportional to the number of movies in the system.

Average number of users per multicast session, $k = n_c/(\frac{L}{d} \times m) = (n_c \times d)/(L \times m)$

Number of existing sessions of a particular movie, $S = n_c/km = \frac{L}{d}$

Therefore, k is a variable and as the movie length and the batch interval are almost constant for the system so S has become pseudo constant term in this scenario.

The procedure *Admission-Control* consists of mainly two threads: one is *Online-Requests-Processing Thread* and the other is *Batched-Requests-Processing Thread.* So, its runtime or complexity depends on the complexity of theses threads and we present the complexity of those threads first.

Before starting the threads the procedure does some initialization tasks by calling procedure *Init-Admission-Controller.* This procedure computes the shortest path trees rooted at each server and there are s servers in the system. *Dijkstra's* algorithm [11] is used to compute the shortest path trees and it takes $O(E \log V)$ computation. Thus, the Procedure *Init-Admission-Controller* requires total $O(sE \log V)$ computation.

Now we discuss the complexity of *Batched-Requests-Processing Thread.* The batched requests are those requests that cannot be satisfied by patching scheme immediately after requesting. The thread first sorts an array using *Bubble Sort* to get the scheduling order (MFQLF) of movies and there are m number of movies. Thus sorting takes $O(m^2)$ computation. As there are limited number of movies in the system, sorting time of the array will not be significant. After sorting, the thread admits the movie requests one by one according to the scheduled order (MFQLF). The acceptance or rejection depends on the available resources. There can be at most n_r requests for a movie as there are total n_r number of clients who are seeking admission. For each movie request, the procedure *admitClient* finds out a multicast session that has the minimum shortest path distance from the client to the multicast tree of that session and there can be at most S sessions of a movie. Finding the availability of resources for admitting a request requires the cost of traversing a multicast tree which is $O(E)$ computation for each session. Thus, the procedure *admitClient* takes $O(SE)$ computation and the main loop of the thread has total $O(n_rSE)$ computation. Therefore, the total complexity of *Batched-Requests-Processing Thread* is $O(m^2 + n_rSE)$ i.e. $O(m^2 + n_cE)$.

The complexity of *Online-Requests-Processing Thread* has the following components:

- The first procedure *Process-Movie-Request* of the thread again consists of two sub-procedures: *selectPatchSession* and *selectPatchingParent*. So, its complexity depends on the complexity of those procedures. They are discussed below:
  - Procedure *selectPatchSession* finds out a patchable session from S sessions of that movie. It also ensures if resources are available along the path from the client to the multicast tree of that session which requires $O(E)$ computation. Thus, the complexity of the procedure is $O(SE)$ i.e. $O(E)$.
  - Procedure *selectPatchingParent* finds out a patching parent of a client from the current members of a session. It first finds out the shortest path tree of the enterprise network rooted at the client node that initiates the request. So, it takes $O(E \log V)$ computation to compute the shortest path tree as discussed earlier. After computing the shortest path tree, it finds out the shortest path client as a patching parent from the participants of that session. Finding shortest path client in a multicast tree requires $O(E)$ computation and the average number of clients per multicast session is k. Thus it takes another $O(kE)$ i.e. $O(n_cE)$ computation. Therefore, the run time of the procedure

selectPatchingParent is $O(E \log V + n_c E)$.

Therefore, the procedure *Process-Movie-Request* requires $O(n_c E + E \log V + n_c E)$ i.e. $O(E \log V + n_c E)$ computation.

- Procedure *Process-VCR-Request* consists of two sub-procedures: *admitVCRRequest* and *selectPatchingParent.* The procedure *admitVCRRequest* performs in a similar way as the procedure *admitclient* does. As we have discussed earlier in this section, the complexity of those sub-procedures are $O(SE)$ and $O(E \log V + n_c E)$ respectively. Thus, the procedure needs total $O(SE + E \log V + n_c E)$ i.e. $O(E \log V + n_c E)$ computation.
- The complexity of procedure *Process-SessionEnd-Request* has the same complexity of procedure *selectPatchingParent.* So, the complexity of this procedure is $O(E \log V + n_c E)$.

Thus, the complexity of *Online-Requests-Processing Thread* will be the sum total of the complexity of its four procedures. So far we consider the computational complexity of processing one request for these procedures. There can be at most n_r online requests of each type. So, the complexity of the thread is $O(n_r E \log V + n_r n_c E)$ i.e. $O(n_c E \log V + n_c^2 E)$.

The complexity of procedure *Admission-Control* is the sum total of the complexity of its two threads and the initial procedure. So, the complexity of the procedure is $O(sE \log V + m^2 + n_c E + n_c E \log V + n_c^2 E)$ i.e. $O(sE \log V + m^2 + n_c E \log V + n_c^2 E)$. Therefore, the runtime of the procedure increases with the increase of the network size, the number of clients and the number of movies in the system.

3.4.5 Client Buffer Requirement Analysis

In conventional Patching scheme *patching stream* is played out first and the *regular stream* is stored in the client buffer for future playout. The time limit, up to when a newly arrived client will be patched after a multicast session starts, is called *patching window* [17]. The buffer requirement of each patched client for conventional Patching depends on the patching window size. So, if the patching window size is λ time units then the upper bound of buffer requirement of each patched client is λM time units ignoring the part of the buffer necessary for patching stream to download and immediate playout. Here, M is the average amount of movie data stream per time unit termed as *stream rate*. In Client-Assisted patching scheme, a client may need to download and store data from two simultaneous streams – one is regular stream and the other is patching stream. However, in conventional patching scheme, a client may need to download data from two simultaneous streams but store only the regular stream.

In Client-Assisted patching, patching stream makes up the initial part and regular stream piles up the later part of the buffer. Each client will cache the initial part of the movie up to the time period of patching window. No request for patching is granted λ time units later from the beginning of the session. If a session starts at Time t_0, then a later client may be patched, if it requests the same movie in $(t_0,\ t_0 + \lambda)$ time interval. So, beyond the patching window, clients may start to give up the *initial chunks* one by one. A *chunk* is the buffer space required to store the stream during each time unit and it depends on the stream rate M.

To examine the buffer requirement of each client we will consider the following two situations assuming a session of a movie has already started at Time t_0:

(i) A client requests the same movie in the interval $(t_0,\ t_0 + \frac{\lambda}{2}]$

(ii) A client requests the same movie in the interval $(t_0 + \frac{\lambda}{2},\ t_0 + \lambda)$

For Case (i), if a client requests the same movie at Time t_1 and $t_1 - t_0 \leq \frac{\lambda}{2}$, then simultaneous downloading and storing of both the streams are required for already passed away time (i.e., $t_1 - t_0$) from the beginning. So, patching stream makes up the missing portion by the time $t_1 + (t_1 - t_0)$ or $2t_1 - t_0$. Note that in this case, $(2t_1 - t_0) \leq (t_0 + \lambda)$.

After $t_0 + \lambda$ time, the initial chunks are freed one by one and making space for the newly arrived data. Streaming at different time intervals is shown in Figure 3.6. Let b be the cache required for the newly admitted client and L be the average movie duration.

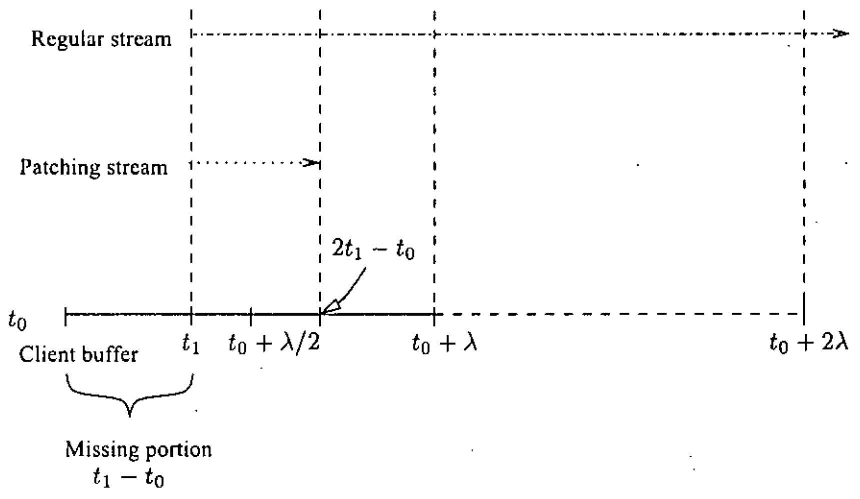


Figure 3.6: Buffer requirement analysis for Case (i).

For Case (i) there may be four states of a client buffer at different intervals as illustrated in Figure 3.6. They are described as follows:

- Downloading and storing both the streams take place in the interval $[t_1, 2t_1 - t_0]$. This is shown in Equation 3.1. After this interval the missing portion is made up and the regular stream continues. At Time $2t_1 - t_0$ the data status of the buffer is $2(t_1 - t_0)M$ and as $t_1 - t_0 \leq \lambda/2$, so the data status of the buffer is at best λM.

- Although the missing portion is made up the initial part needs to be in the cache up to Time $t_0 + \lambda$ i.e. the patching window time. A client may be selected as a patching parent within this time interval. This is shown in Equation 3.2. At the end of interval the data status of the buffer is λM.

- After Time $t_0 + \lambda$ the client begins to relinquish data from the initial part of the buffer and making space for the upcoming streams. If the client is selected as a patching parent just before Time $t_0 + \lambda$ then its patching is ended up at Time $t_0 + 2\lambda$. Thus the data status of the buffer is shown in Equation 3.3 which is λM.

- After Time $t_0 + 2\lambda$ patching by the client is ended up if it is selected as patching parent. So, only the data equivalent to the initial missing time will remain in its buffer during the rest of the time. This is shown in Equation 3.4.

$$\begin{aligned}
b &= 2(t - t_1)M \qquad t \in [t_1, 2t_1 - t_0] && (3.1)\\
b &= 2(2t_1 - t_0 - t_1)M + (t - 2t_1 + t_0)M \qquad t \in (2t_1 - t_0, t_0 + \lambda] \\
&= (2t_1 - 2t_0 + t - 2t_1 + t_0)M = (t - t_0)M && (3.2)\\
b &= 2(t_1 - t_0)M + (t_0 + \lambda - 2t_1 + t_0)M \qquad t \in (t_0 + \lambda, t_0 + 2\lambda] \\
&= (2t_1 - 2t_0 + \lambda - 2t_1 + 2t_0)M = \lambda M && (3.3)\\
b &= (t_1 - t_0)M \qquad t \in (t_0 + 2\lambda, L] && (3.4)
\end{aligned}$$

Considering Case (ii), if a session starts at Time t_0 and a new client requests the same movie at Time t_1 and $t_1 - t_0 > \frac{\lambda}{2}$. In this case downloading and storing both patching and regular streams will be continued up to $2t_1 - t_0$ from Time t_1. Note that in this case, $(2t_1 - t_0) > (t_0 + \lambda)$. As the possibility of selecting a client as a patching parent is up to time interval $(t_0,\ t_0 + \lambda)$, the stored data of the buffer can be freed gradually from the beginning after $t_0 + \lambda$ time. So, after $t_0 + \lambda$ Time, earlier chunks will be freed one by one making space for regular stream. Streaming at different time intervals is shown in Figure 3.7.

Now we explain the buffer states for Case (ii). There may be four states of a client buffer at different intervals as illustrated in Figure 3.7. They are discussed as follows:

- Downloading and storing both the streams take place in the interval $[t_1, t_0 + \lambda]$. At the end of this interval the data status of the buffer will be at best λM. This is shown in Equation 3.5.

- A client may be selected as a patching parent up to Time $t_0 + \lambda$ that is up to the patching window time. So, initial data needs to be in the buffer is up to this time.

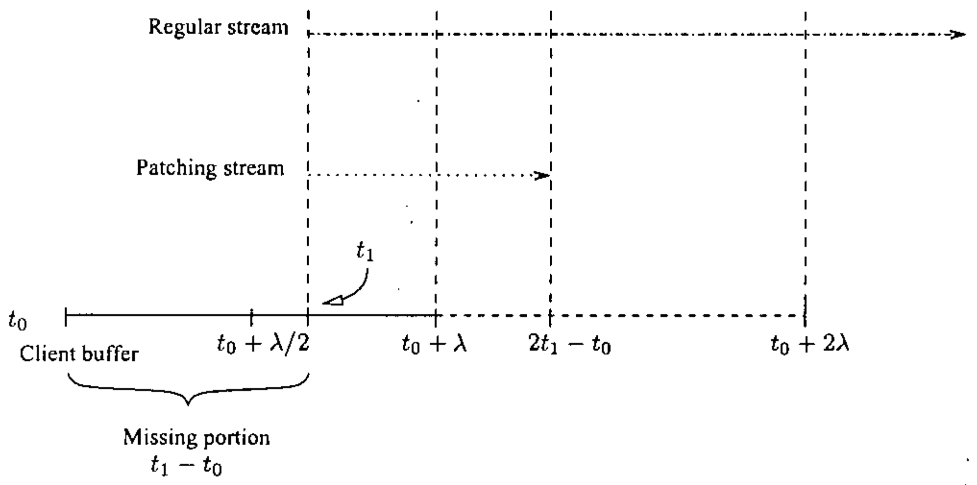


Figure 3.7: Buffer requirement analysis for Case (ii).

After that the client begins to relinquish data from the initial part of the buffer. However, downloading and storing from both the streams will continue up to time $2t_1 - t_0$. So, downloading, storing and evacuating simultaneously take place during this time interval. The combined situation is shown in Equation 3.6 and the data status never crosses the maximum size of the buffer i.e. λM.

- The missing portion is be made up at Time $2t_1 - t_0$ and only the patching stream continues henceforth. If the client is selected as a patching parent just before Time $t_0 + \lambda$ then its patching will be complete at Time $t_0 + 2\lambda$. This is shown in Equation 3.7 and the size of the buffer is adequate hold the required data.

- There is no chance of patching after Time $t_0 + 2\lambda$. So, only the data equivalent to the initial missing time will remain in its buffer during the rest of the time. Again the data status of the buffer is at best λM and this is shown in Equation 3.8.

$$b = 2(t - t_1)M \qquad t \in [t_1, t_0 + \lambda] \tag{3.5}$$

$$\begin{aligned} b &= 2(t_0 + \lambda - t_1)M + (t - \lambda - t_0)M \qquad t \in (t_0 + \lambda, 2t_1 - t_0] \\ &= 2(t - t_1)M \end{aligned} \tag{3.6}$$

$$
\begin{aligned}
b &= 2(\lambda - t_1 + t_0)M + (2t_1 - t_0 - t_0 - \lambda)M \qquad t \in (2t_1 - t_0, t_0 + 2\lambda] \\
&= 2(\lambda - t_1 + t_0)M + (2t_1 - 2t_0 - \lambda)M = \lambda M \qquad (3.7) \\
b &= (t_1 - t_0)M \qquad t \in (t_0 + 2\lambda, L] \qquad (3.8)
\end{aligned}
$$

3.4.6 Client Buffer Management

This section presents the mechanism of storing the chunks of movie streams received from multicast stream of server and patching stream of patch parent. We also discuss the data structure of the buffer of a client and the mechanism of discarding chunks when not necessary.

Data Structure and Buffering Policy

The data structure of a client in the proposed system is presented in Figure 3.8. The features of the buffer array is described below:

- This is an array of structure and there are three fields in this structure. They are *chunk#*, *ptr to chunk data* and *chunk data.*
- Free chunks are denoted by "-1" value in the *chunk#* field. So, initially all the these fields will be "-1". The occupied chunks will be denoted by an integer number starting from 0.
- When any stream is downloaded it is stored in the first available free chunk of the buffer array. Chunks from both patching and regular streams will be stored simultaneously. So, the array is not sorted according to *chunk#*.
- When the multicast session expires the patching window time discarding of initial chunks will be started. The chunk with minimum *chunk#* will be freed at each time unit. "-1" will be assigned in *chunk#* field of the corresponding freed chunks.

- Client starts playout from *chunk*0. As the array is not sorted, the next chunk will be searched and played out by the client. This will happen each time after finishing playout of a chunk. *Linear Search* will be used in this case as there will be a limited number of chunks in the buffer. So the search time will not be significant. Moreover, playout of current chunk and searching the next chunk can simultaneously take place.

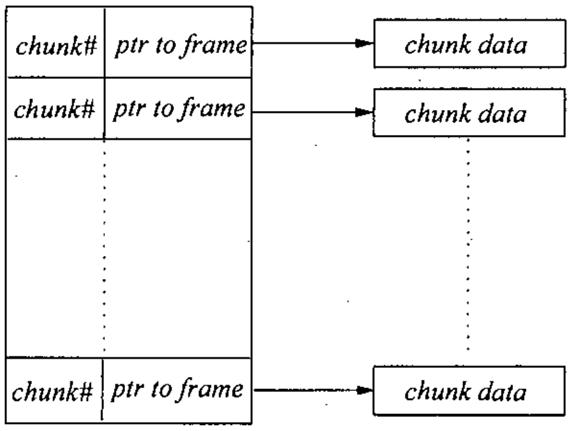


Figure 3.8: Data structure of the buffer of a client.

3.5 Service Interruption

A service interruption may occur for the patched child client. This may happen when the patching parent issues a VCR request or leave the session. When a client issues an interactive action it can no longer be in the current multicast session. So, the leaving client cannot be a patching parent. In this case another patching parent is to be selected by the Admission Controller to deliver the missing portion of the patched child client. The leaving patching parent can send a message to the child client about its departure or the child client will eventually encounter the loss of patching stream. Whatever may the leaving process, delay might be introduced for the patched child client in this situation.

Service interruption is presented in Figure 3.9. If client C_1 is serving client C_3 as a

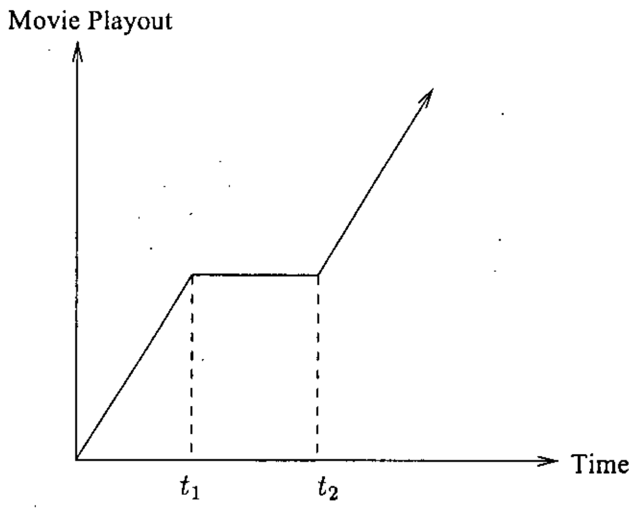


Figure 3.9: Service interruption in client-assisted patching.

patching parent. At Time t_1, client C_1 leaves the session. The Admission Controller finds an alternate patching parent C_2 for client C_3. Now C_2 starts to provide patch stream to client C_3 at Time t_2. So, the service of client C_3 has been interrupted or blocked for $(t_2 - t_1)$ time as shown in Figure 3.9.

We propose a solution to this problem. Each patched client will delay the initial playout for a certain amount of time. However, buffering takes place during this time. But regular clients can start the playout immediately. This delay time might make up the service interruption due to the leaving of patching parent. However, the problem might occur again and again when newly selected patching parents leave the session for some reasons.

Chapter 4

Performance Study

In the previous chapters we have described the architecture and algorithm of the admission controller along with its complexity analysis. In this chapter we describe the simulation results of Client-Assisted Patching in an enterprise network. Through simulation we study the behavior of our approach and evaluate its performance based on some performance metrics. We also compare the simulation results of the proposed scheme with conventional *Patching* [17].

4.1 Simulation Setting

We simulate Client-Assisted patching using Parsec [6], a C-based parallel discrete event simulation language developed in UCLA Parallel Computing Laboratory. The main objective of the simulation is to understand the impact of various parameters on the performance issues of the proposed system. We also simulate conventional *Patching* to make a comparison between our approach and conventional Patching on various metrics. In the following sub sections we describe the various settings of our simulation. Table 4.1 presents the entire simulation setting. Similar parameter setting is considered in simulating conventional Patching scheme.

Table 4.1: Simulation Setting.

Parameter	**Value**
General Settings	
Number of nodes	20
Number of links	32
Number of clients	800 - 1200
Number of servers	4 – 10
Number of replica	1 – 3
Event Reporting	
Reporting type	Discrete Event-driven
Packet type	Request, Response, Query
Request generation process of a client	Poisson process
Inter-arrival of events	Exponential (mean 1000 STU*)
Event generator client selection	Uniform random
Event Timings	
Batch interval	60 – 300 STU
Threshold for VCR request	integer multiple of batch interval
Patching window	300 – 720 STU
Duration of a movie	3600 STU
Simulation time	86400 STU
*STU means Simulation Time Unit	

4.1.1 Simulation Parameters

We run our simulation for an enterprise network of size 20 switch nodes and 32 interconnecting links between them. The network is considered as a connected graph of switch nodes. Number of clients are varied in the rage 800–1200. The clients are randomly distributed to different nodes. Thus, the clients are evenly distributed to different nodes. While getting some particular simulation results a specific parameter is varied keeping other parameters constant. These usual parameter settings are shown in Table 4.2.

Client requests are generated in our simulation according to a Poisson process. We vary the request rate in a range from 0.042 request/second to 1 request/second where

Table 4.2: Usual Simulation Setting.

Parameter	**Value**
General Settings	
Number of clients	800
Number of servers	5
Number of replica	2
Event Reporting	
Request rate of a client	1 request/second
Event Timings	
Batch interval	120 STU
Threshold for VCR request	batch interval
Patching window	600 STU

0.042 request/second indicates a lightly loaded system and 1 request/second indicates a heavily loaded system. The videos are requested with frequencies following a *Zipf-like* [12, 26] distribution with a skew factor z. Specifically, the probability that a video i is selected is $\frac{1}{i^z \sum_{j=1}^{N} \frac{1}{j^z}}$. Higher value of skew factor means that there are few popular movies. Top ten popular movies are replicated in different servers.

We consider continuous VCR actions of pause, fast-forward and rewind. We consider the probability of issuing VCR actions during a playout is higher at the later part of a movie than the earlier part of the movie. We also consider a similar probability distribution about leaving a session during a playout. We do not consider service interruption due to leaving of patching parents in this simulation. However, we think that as patching takes place in the initial part of the movie, both the probabilities of clients' issuing VCR request and session leave request in the earlier part of a movie is much less than later part of the movie. Thus, service interruption due to leaving of patching parent will be negligible.

We consider MPEG-2 streaming which requires network bandwidth in the range of 3 10 Mbps. However, we consider 5 Mbps for MPEG-2 in our simulation. We consider a flat revenue policy which means that any movie (cool or hot) requires the same amount of money to be paid by the customer [24].

4.1.2 Performance Metrics

The performance metrics that we consider in our simulation are *Server Bandwidth Requirements, Percentage Served, Execution Time* and *Percentage of VCR Action Blocking.* We observe these performance metrics of Client-Assisted Patching by varying different parameters like number of servers, number of replica of popular movies and so on. We also compare the observed results with conventional *Patching.* It is observed that the proposed approach outperforms conventional Patching in all respects. But the new approach assumes that the clients in the system are cooperative and hence each client requires a minimum buffer space that will be used for patching other clients. However, we believe that the availability of small buffer space is quite common in any kind of multimedia devices and due to rapid technology advancement, incorporating buffer space in any device is not expensive now a days.

4.1.3 Simulation Technique

We simulate both Client-Assisted Patching and conventional *Patching* using Parsec. Parsec is an entity based discrete event simulator. In Parsec, we have two kind of objects: **entity** - a processing node that performs certain operations and **message** – information passed from one entity to another to coordinate functions among the entities. In Parsec, messages trigger discrete events to the entities. One entity can send message to and can receive message from another entity. At the very beginning of the simulation, all entities are created and then throughout the simulation, they pass messages among themselves to coordinate their operations. The simulation is ended by setting up a total simulation time. The results reported in the figures indicate the situation of the simulation running for the whole simulation time as shown in Table 4.1. When the simulation is ended necessary statistics are collected.

In our simulation, we build the following entities:

- **ClientEntity:** This entity represents a client. The operations that are to be performed by a client are specified in this entity.
- **ServerEntity:** This entity represents a server. The operations that are to be performed by a server are specified in this entity.
- **ADCEntity:** This entity represents the admission controller. It is the important entity of our simulation. All coordinating operations with clients and servers are specified in this entity.
- **Driver:** This is the main entity that every simulation program written in Parsec has. This is the entity that is activated at the beginning. Driver entity creates other entities, initialize all entities with their initialization routine and starts the simulation.

We construct the following messages:

- **StartSimulation:** This message is sent by Driver entity to other entities to start the simulation.
- **Request and Response:** All requests and responses of different entities.
- **Multimedia Data:** These are multimedia data related messages.

4.2 Performance Analysis of Client-Assisted Patching

In this section we discuss the simulation results obtained for *Client-Assisted Patching.* We discuss the performance analysis of the proposed system by varying different simulation parameters. The parameters we consider are the number of servers, number of replica of popular movies, number of movies, number of network nodes, patching window size and total number of clients in the system.

4.2.1 Number of Servers

We have considered enterprise network with multiple servers. The number of servers in the network has a direct influence over the performance of the system. The movies are distributed randomly in the servers. And only the popular movies are replicated to different servers. If we increase the number of servers then the movies are further dispersed and there are more alternative sources of movies. This definitely increases the percentage of served requests of the system. Figure 4.1 shows the percentage of served requests for different request rates. For low request rate the system accepts almost all the requests and the system remains underutilized. We find the effectiveness of using more servers for higher request rates where the system is totally congested with lower acceptance rate by the Admission Controller and hence resulting lower percentage of served requests.

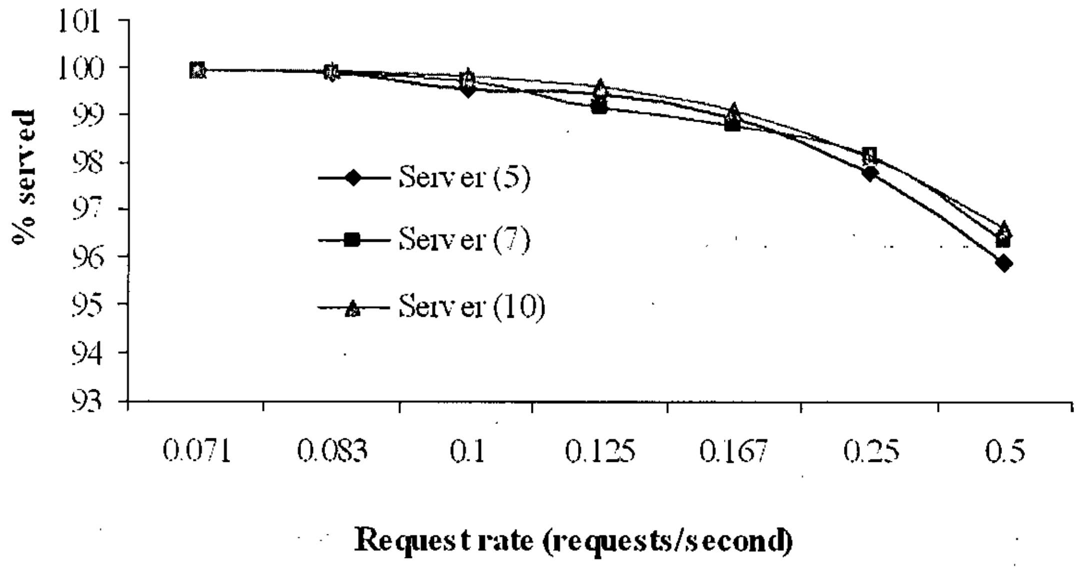


Figure 4.1: Percentage of served requests for different request rates in Client-Assisted Patching with different number of servers.

4.2.2 Number of Replica

The movies which experience most of the requests are called *popular* or *first round* movies [24]. We consider the replication of popular movies only to satisfy the high demand of such movies. The popular movies are replicated to different servers and the number of replication has been changed to observe the performance without changing other parameters of the system. Increasing the number of replica of popular movies thereby increases the number of alternative sources of popular movies. Thus, the system will be able to admit more requests of popular movies and this in turn increases the percentage of served requests of the system. Figure 4.2 shows the effect of replication of movies with the increase in request rates. We observe almost the same behavior as reported for the curve for different number of servers shown in Figure 4.1. The percentage of served requests increases significantly if replication is used for higher request rates. The effect of replication is higher than that of using more servers on the percentage of served requests.

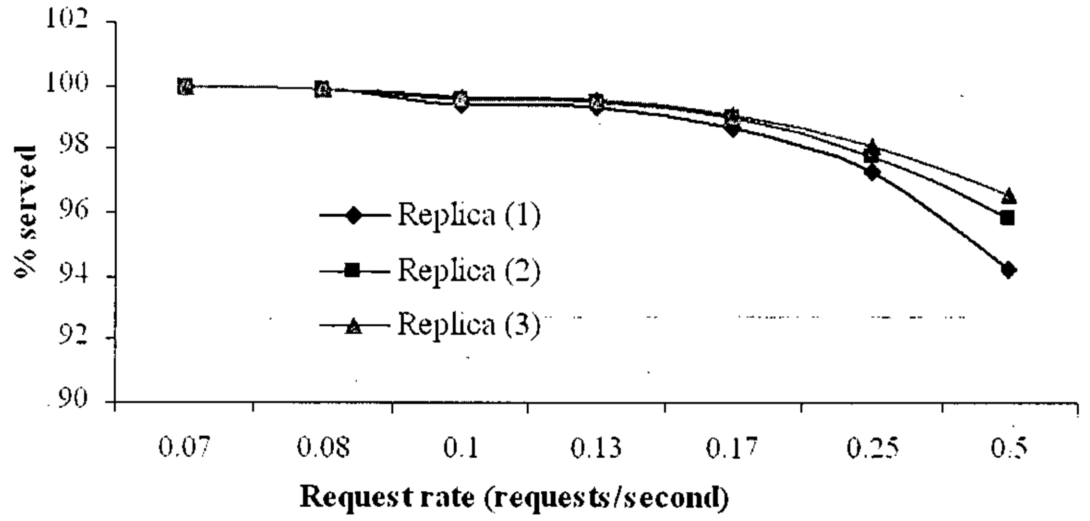


Figure 4.2: Percentage of served requests in Client-Assisted Patching with 5 servers for different request rates and replication.

4.2.3 Number of Movies

In this section we will show the effect of increasing the number of movies in the proposed system. As the number of movies increases then client requests are generated for increased number movies by the clients. This increases the number of different groups of requests (i.e. sessions to be served) in the system which puts an increased load on the server and network bandwidths of the system. This observation complies with the analysis presented in Section 3.4.4. Multicast VoD system shows better performance when the number of similar requests increases in the system. As increasing the number of movies decreases the number of similar requests in the system, the performance degrades at increasing request rates. This is observed in our simulation results as shown in Figure 4.3. The percentage of served requests decreases as the number of movies increases in the system.

104364

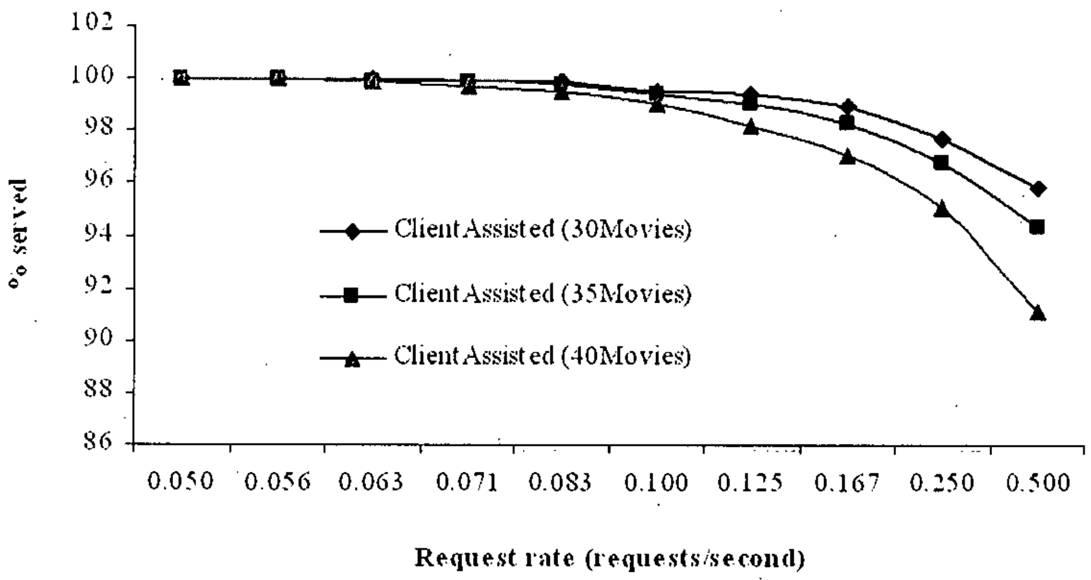


Figure 4.3: Percentage of served requests in Client-Assisted Patching with increasing the number of movies in the system.

4.2.4 Number of Network Nodes

In this section we will discuss the performance of the system when the size of the Enterprise Network is increased. The performance of the system is dependent on a number of parameters of the system. Among the parameters, the bandwidth of the servers and interconnecting links of the network nodes is the most significant. Increasing the network size i.e. the number of network nodes and connecting links, increases the bandwidth resource of the system. Thus, the system will show an overall better performance when the size of the network is increased. This is also observed in our simulation as shown in Figure 4.4. The increase in percentage of served requests is insignificant because the number of servers has been kept unchanged in this experiment.

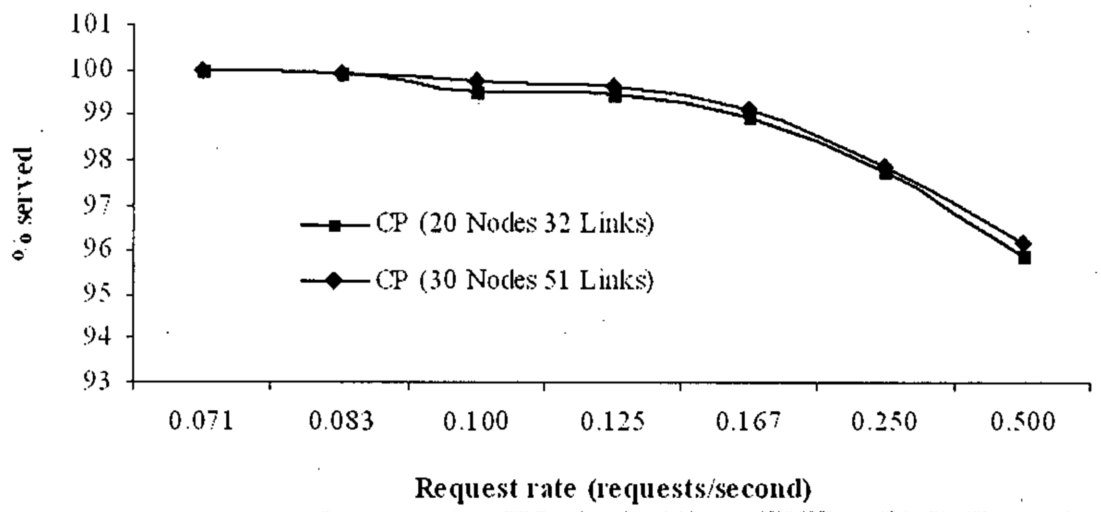


Figure 4.4: Percentage of served requests in Client-Assisted Patching with increasing the network size of the system.

However, increasing the number of nodes and connecting links of the Enterprise Network has a significant impact on the execution time of the admission control algorithm as explained in Chapter 3, Section 3.4.4. This is observed in our simulation as shown in Figure 4.5.

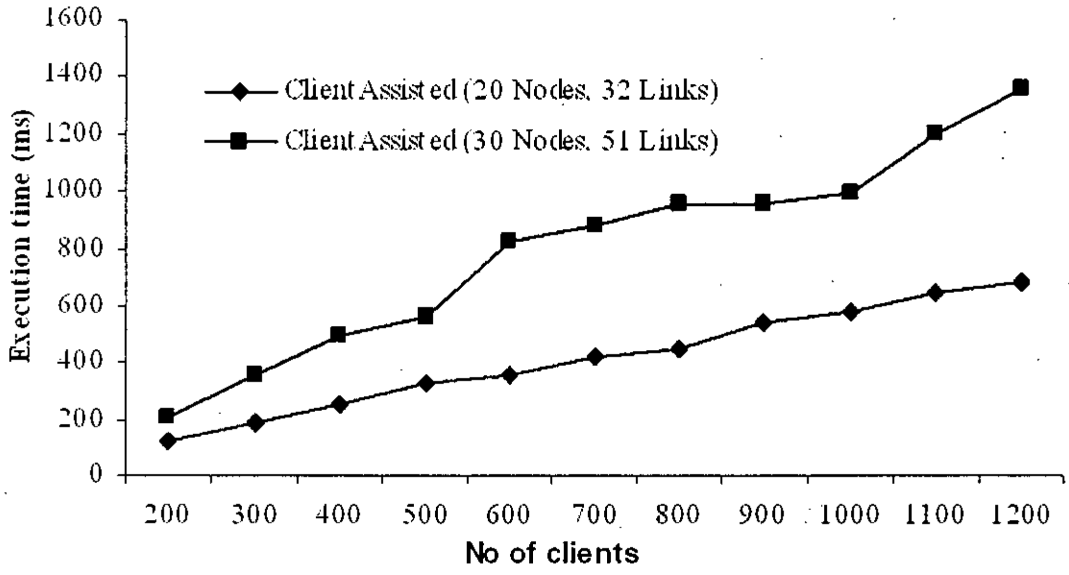


Figure 4.5: Execution time of admission control algorithm of Client-Assisted Patching with increasing the network size of the system.

4.2.5 Patching Window

In this section we will show the effect of patching window in the proposed system. As the patching window size increases, more and more client requests are likely to be patched. Thus patching will be more effective at increased window size and thereby increasing the percentage of served requests of the system. This is observed in Figure 4.6. We find similar behavior of the percentage of served requests as reported in the earlier cases. In the curves (Figure 4.1 – Figure 4.6) showing the effect of different parameters (number of servers, replication, patching window) on the percentage of served requests, we observe that the effect of the performance of the system diminishes with the increase in the parameter values.

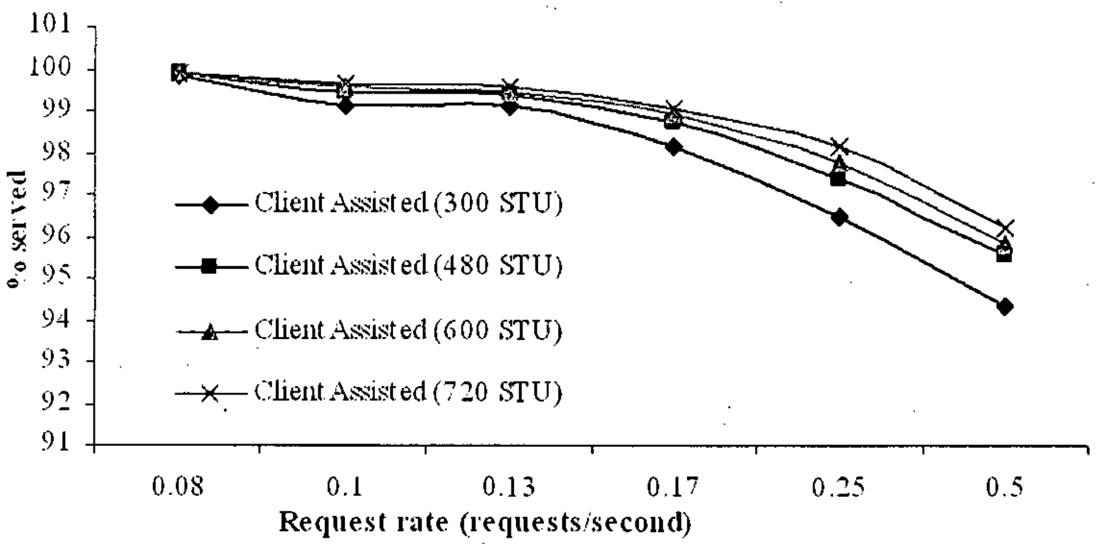


Figure 4.6: Percentage of served requests in Client-Assisted Patching with different patching window sizes at different request rates.

4.3 Comparison with Conventional Patching

In this section, we compare various results of *Client-Assisted Patching* with the results obtained for conventional *Patching* scheme. The reason why we do not compare with *Cooperative Patching* is that it is a periodic multicast approach where ours is a multicast approach. It is a single server approach and ours is a multiple server approach. Moreover, this thesis proposed a technique which alleviates the server load incurred by conventional Patching scheme.

We observe and compare several performance metrics such as bandwidth requirement of servers, the percentage of served requests of the system, VCR action blocking rate and time requirement from the simulation results obtained by varying different parameters.

4.3.1 Bandwidth Requirement of Server

Server bandwidth is very important resource in multicast VoD system because only a single server stream is required to satisfy a group of clients. No server stream is required to patch

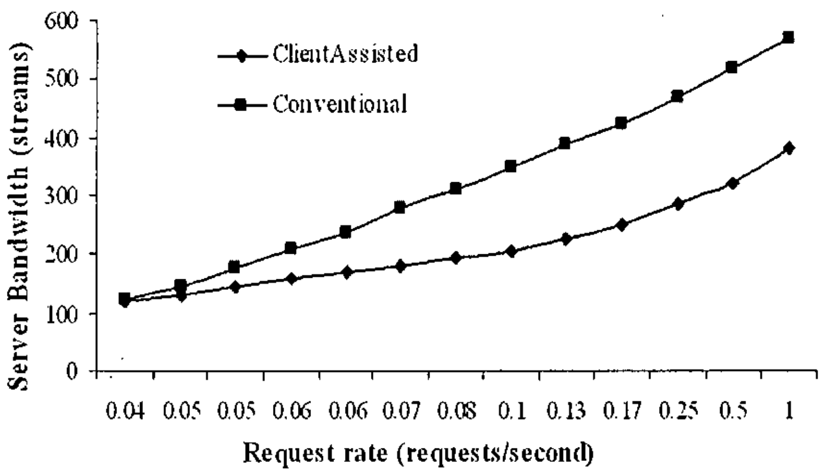


Figure 4.7: Server Bandwidth Requirements for different request rates of conventional and Client-Assisted Patching schemes.

any client in the proposed system. Thus the conserved bandwidth of the servers can be used to satisfy more multicast groups. On the other hand, conventional Patching requires each patched client to assign a dedicated server stream to supply the missing portion of the movie. So, server bandwidth will be exhausted as the server serves more and more patching requests within a short time. Therefore, Client-Assisted Patching will be more scalable than conventional Patching. We measure the server bandwidth requirements by varying the server bandwidth until the system satisfies all customer requests. The server streams required at the varying request rate has been shown in Figure 4.7. As patching stream is supplied by the cooperative clients, the server load is significantly reduced. Our simulation results show that similar load can be carried out by the proposed system with 20% to 30% less server bandwidth than conventional Patching system.

4.3.2 Percentage of Served Requests

In this section we compare the percentage of served requests of the proposed scheme with conventional Patching. We have already discussed that Client-Assisted Patching will be more scalable than conventional Patching because it conserves the server bandwidth

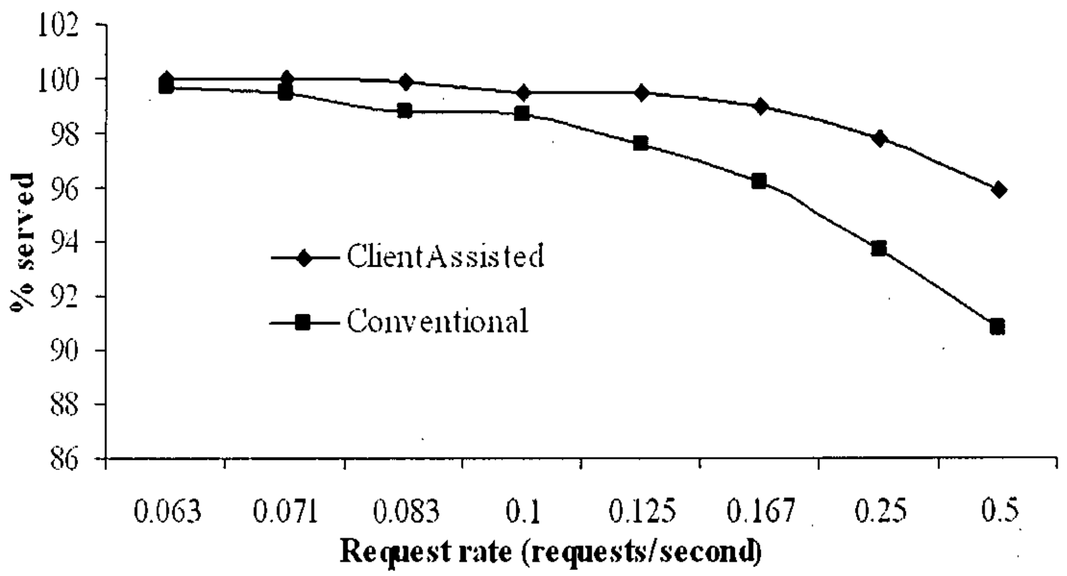


Figure 4.8: Percentage of served requests for different request rates using conventional and Client-Assisted Patching schemes.

that was otherwise necessary to supply the dedicated patching channels to the patched clients. The preserved bandwidth can be used to satisfy more multicast groups and thereby increasing the serve-rate of the system. Simulation results have been shown in Figure 4.8. We observe that serve-rate of our system has been increased as the request rate has been increased. For low request rate the servers are under utilized and thus both the scheme shows the same performance. Client-Assisted Patching starts showing better performance when the servers become utilized with higher request rates.

4.3.3 Patching Window Effect

We have already presented the results regarding patching window size of the proposed system. Increasing patching window size has an adverse effect in VoD system which considers conventional *Patching* [24]. This is due to the fact that the patching channels are dedicated for the patched clients. So, when we increase the patching window size, we are actually increasing the duration of the existence of these dedicated channels. Thus the VoD system considering conventional Patching has a negative impact over increasing patching window size. After a certain limit, the performance of conventional Patching degrades while increasing the patching window size. If we consider the VoD system with Client-Assisted Patching, patching window enlargement has only adverse effect of increasing the service latency. The results have been shown in Figure 4.9.

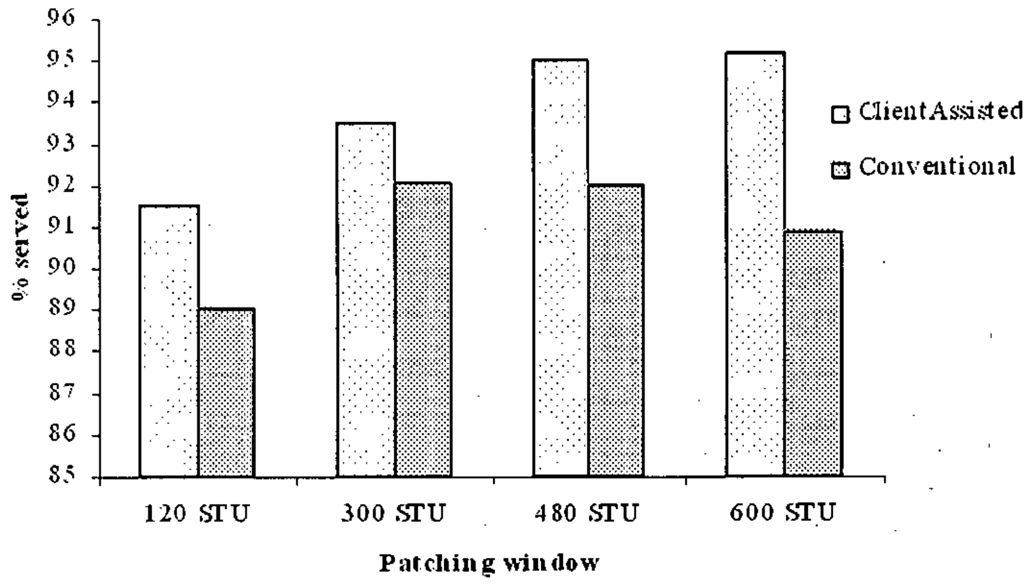


Figure 4.9: Percentage of served requests of conventional and Client-Assisted Patching schemes with different patching window sizes keeping other parameters constant.

4.3.4 VCR Action Blocking

In this section we discuss the percentage of VCR requests that are ignored by the system also called *VCR action blocking*. In most of the cases VCR requests are satisfied by dedicated channels from the server. As our scheme conserves useful server bandwidth, more interactive requests can be satisfied through utilizing this bandwidth. Figure 4.10 shows the percentage of blocked VCR requests for different request rates. Client-Assisted Patching shows better performance than conventional Patching with less rate of blocking VCR requests. Client-Assisted Patching shows competitively better performance for higher request rate. The reason for this behavior has already described in the experimental results shown in previous sections.

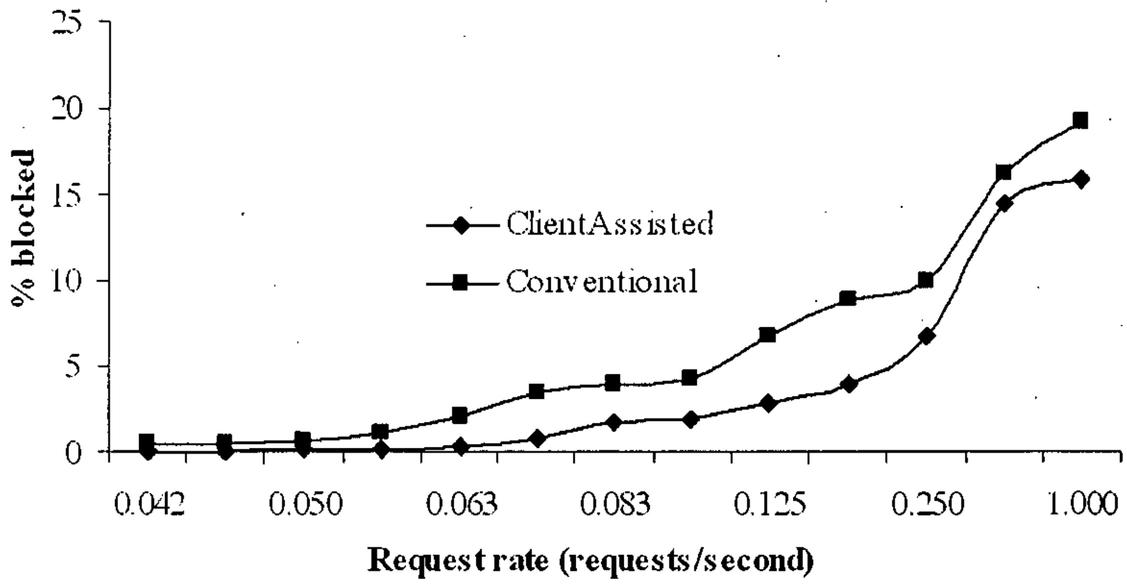


Figure 4.10: Percentage of blocked VCR requests of conventional and Client-Assisted Patching schemes at different request rates.

4.3.5 Execution Time Comparison

As we have already discussed in Chapter 3 that the *Admission Control* procedure of the proposed system needs $O(sE \log V + m^2 + n_c E \log V + n_c^2 E)$ computation where s is the number of servers, V is the number of switch nodes, E is the number of connecting links i.e. edges, m is the number of movies and n_c is the number of clients presently enjoying movies in the system. So, the time complexity depends on these parameters. However, as the computation time increases in quadratic manner with the increase in the number of clients currently enjoying multimedia data in the system. We observe this profile of computation time in the graphs of Figure 4.11.

We assumed the same simulation parameters and environment for conventional Patching and found its complexity as $O(sE \log V + m^2 + n_c E)$. So, it requires less time for conventional Patching because no computation is required for selecting patching parent for each client request. Thus, the proposed system needs more computation than conventional Patching and hence takes more CPU time. This is observed in our simulation results as shown in Figure 4.11.

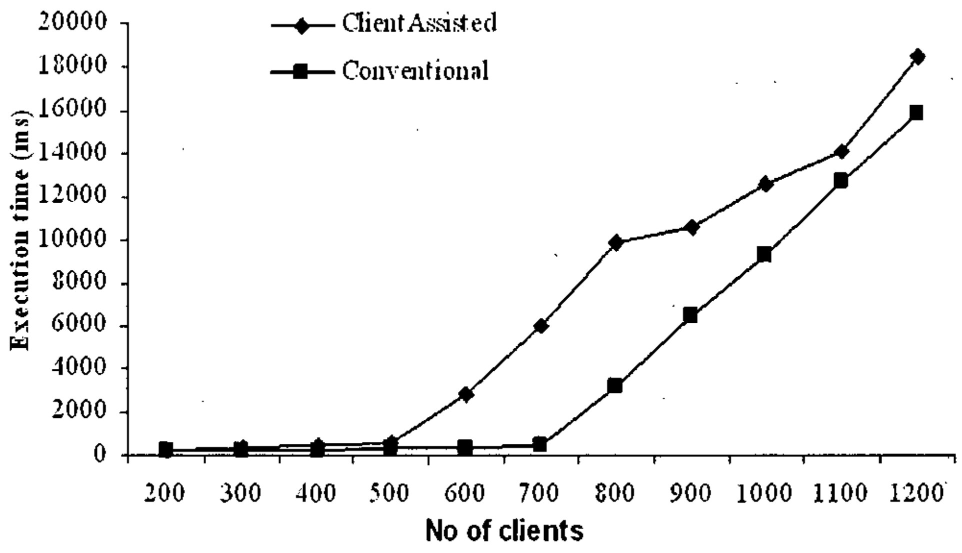


Figure 4.11: Computation time of conventional and Client-Assisted Patching schemes with varying number of clients in the system.

Chapter 5

Conclusion

In this last chapter, we draw the conclusion of our thesis by describing the major contributions made by the research works associated with the thesis followed by some directions for future research over the issue.

5.1 Major Contributions

Multicast Video-on-Demand (MVoD) systems are scalable and cheap-to-operate. Over the past few years extensive research has been done on MVoD. Even though the significant progress has been made, it is still regarded as challenging research domain in VoD service. Our thesis work contributes to this challenging area.

The contributions that have been made in this thesis can be described as follows:

- In this thesis, we consider a multicast video-on-demand service in an Enterprise Network with multiple servers. The network is controlled and owned by an organization. To the best of our knowledge, considering multicast in an Enterprise Network with multiple media servers has been examined little so far. Most of the previous works of this kind is mainly devoted to the Internet architecture.

- We presented the architecture of a new admission controller along with proposals for data transmission protocols in different stages of the admission controller. We designed an admission control methodology for the architecture and we also analyzed the complexity of the controller.

- We also proposed a new patching technique called Client-Assisted Patching which outperforms conventional Patching. In Client-Assisted Patching, client side resources such as I/O bandwidth and buffer are utilized to promote the system performance through patching. This, significantly alleviates the server load. As server bandwidth is a very important resource in any VoD service, it greatly enhances the system performance.

- Client-Assisted Patching does not require cache size larger than does require for conventional Patching. However, it requires each client should have at least a certain amount of buffer space and I/O bandwidth to cooperate other clients. Which is an exception from the conventional Patching scheme where buffer space is only required for the patched clients. But we think that incorporating buffer space is no longer a costly design issue now a days.

- We combine two techniques in the proposed system. They are Batching and Patching. Thus the system combines the advantages of two different techniques which makes the proposed system more robust and highly scalable. We also adopted a fair and efficient scheduling policy in the proposed system.

- We proposed a new technique about rejoining an ongoing session due to multicast group change request. Multicast group change request occurs when a customer issues a VCR action. Our proposed technique will decrease VCR action blocking rate of the system.

- We not only present the multicast video-on-demand system, but also simulate the system using Parsec to make closer observations into the VoD system. We make a rigorous simulation based study of various performance issues of the proposed approach and analyze the simulation output against the expected behavior.

- We also make a simulation of our counter scheme, conventional *Patching*, to compare our approach with it. Simulation reveals that Client-Assisted Patching outperforms conventional *Patching* with a very sharp margin in various important aspects like bandwidth requirement of servers.

5.2 Future Directions of Further Research

No research work can ever end. Any research on any topic always makes a way to further research on the way. Ours is not an exception. Based on our current design and the results of simulations presented in this thesis, we can look into the extension of our works in future in the following directions:

1. We designed the admission controller in a small scale, for an Enterprise Network, a network with limited number of nodes and edges. Further study is necessary to extend the architecture for Internet or interconnected multiple Enterprise Networks.

2. Our admission controller acts as a central moderator in the VoD system. Like any centralized system, our system is also prone to the problem of single point of failure. A distributed system can be established where multiple admission controllers will try to optimize their respective revenues from users' requests which requires data transmission among different networks.

3. We consider that a movie or its replication is entirely kept on a single server. We do not consider keeping different part of a movie on different servers. Collecting and combining data from different sources and transmitting it to a particular destination is a very complex problem. A different methodology has to be developed to handle segmented storage of multimedia data.

4. The system rejects some clients' requests due to resource shortages. Rejected clients simply leave the system. But users sometimes prefer to make future reservations. Thus the admission controller and different protocols need to be redesigned.

5. In our system, patching takes place only at the beginning of initial customer requests. However, one can consider to patch a customer when it wants to rejoin a different multicast group due to an interactive action. Further study is required in efficient management of VCR action for significantly reducing interactive action blocking rate.

Bibliography

[1] C. C. Aggarwal, J. L. Wolf and P. S. Yu, "A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems," *IEEE Int'l Conference on Multimedia Systems*, June 1996.

[2] M. M. Akbar, "Distributed Utility Model Applied to Optimal Admission Control and QoS Adaptation in Distributed Multimedia Server System and Enterprise Networks," *PhD Dissertation, Department of Computer Science, University of Victoria*, August 2002.

[3] M. M. Akbar, E. G. Manning, R. K. Watson, G. C. Shoja, S. Khan and K. F. Li, "Optimal Admission Controllers for Service Level Agreements in Enterprise Networks," *SCI Conference*, pp. 14-18, Orlando, July 2002.

[4] K. C. Almeroth and M. H. Ammar, "On the Performance of a Multicast Delivery Video-on-Demand Service with Discontinuous VCR Actions," *International Conference on Communication (ICC1995)*, Seattle, Washington, June 1995.

[5] K. C. Almeroth and M. H. Ammar, "The Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service," *IEEE Journal in Communications*, vol. 14, no. 5, pp. 1110-1122, August 1996.

[6] R. Bagrodia, R. Meyerr et al., "PARSEC: A Parallel Simulation Environment for Complex System," *UCLA Technical Report*, 1997.

[7] L. Bajaj, M. Takai, R. Ahuaja, K. Tang, R. Bagrodia and M. Gerla, "Glomosim: A Scalable Network Simulation Environment," *Technical Report 990027*, UCLA Computer Science Department, May 1999.

[8] Y. Cai, K. A. Hua and K. Vu, "Optimizing Patching Performance," *Multimedia Computing and Networking, MMCN'99*, January 1999.

[9] Y. Cai and K. A. Hua, "An Efficicent Bandwidth-Sharing Technique for True Video on Demand Systems," *ACM Multimedia'99*, pp. 211-214, Orlando, November 1999.

[10] S. W. Carter and D. D. E. Long, "Improving Video-on-Demand Server Efficiency Through Stream Tapping," *Sixth International Conference on Computer Communications and Networks, (ICCCN1997)*, pp. 200-207, Las Vegas, NV, USA, September 1997.

[11] T. Cormen, C. E. Leiserson and R. Rivest, "Introduction to Algorithms," *Prentice-Hall*, India.

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

[13] L. Gao and D, Towsley, "Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast," *IEEE ICMCS'99*, pp. 117-121, Florence, Italy, June 1999.

[14] M. Guo and M. H. Ammar, "Scalable Live Video Streaming to Cooperative Clients Using Time Shifting and Video Patching," *IEEE Infocom*, Hong Kong, March 2004.

[15] M. Guo and M. H. Ammar, "Cooperative Patching: A Client Based P2P Architecture for Supporting Continuous Live Video Streaming," *IEEE International Conference on Computers*, Chicago, IL, October 2004.

[16] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," *ACM SIGCOMM*, September 1997.

[17] K. A. Hua, Y. Cai and S. Sheu, "Patching: A multicast Technique for True Video-on-Demand Services," *Sixth ACM Multimedia Conference*, pp. 191-200, Bristol, UK, September 1998.

[18] K. A. Hua and D. A. Tran, "Range Multicast for Video on Demand," *Multimedia Tools and Applications Archive*, vol. 27, pp. 367-391, May 2005.

[19] M. M. Islam, M. M. Akbar, H. Hossain and E.G. Manning, "Admission Control of Multimedia Sessions to a Set of Multimedia Servers Connected by an Enterprise Network Communications," *Computers and Signal Processing (PACRIM2005), IEEE Pacific Rim Conference*, pp. 157-160, August 2005.

[20] S. Jin and A. Bestavros, "Cache-and-Relay Streaming Media Delivery for Asynchronous Clients," *NGC*, 2002

[21] J. Jung and D. Lee, "Harmonic Broadcasting for Video-on-Demand Service," *IEEE Transactions on Broadcasting*, vol. 43, no. 3, pp. 268-271, 1997.

[22] T. Little and D. Venkatesh, "Prospects of Interactive Video-on-Demand," *IEEE Multimedia*, pp. 14-23, August 1994.

[23] H. Ma and K. G. Shin, "A New Scheduling Policy for Multicast True VoD Service," *PCM2001*, vol. 2195, pp. 708-715, October 2001.

[24] H. Ma and K. G. Shin, "Multicast Video-on-Demand Services, *ACM Computer Communication Review*, vol. 32, no. 1, pp. 31-43, 2002.

[25] H. Ma, K. G. Shin and W. Wu, "Best-Effort Patching for Multicast True VoD Service," *Multimedia Tools and Applications Archive*, vol. 26, pp. 101-122, May 2005.

[26] V. Poosala, "Zipf's Law," *cs.wisc.edu.*

[27] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *http://search.ietf.org/internet-drafts/draftietf-mpls-arch-06.txt*, 1999.

[28] N. J. Sarhan and C. R. Das, "A Simulation-Based Analysis of Scheduling Policies for Multimedia Servers," *36th Annual Simulation Symposium, Part of the Advanced Simulation Technologies Conference*, pp. 183-190, March 30 - April 2, 2003.

[29] S. Sheu, K. A. Hua and W. Tavanapong, "Chaining a Generalized Batching Technique for Video-on-Demand Systems," *IEEE ICMCS'97*, pp. 110-117, Ottawa, 1997.

[30] A. S. Tanenbaum, Computer Networks, 4th Edition, 2003.

[31] S. Viswanathan and T. Imielinski, "Metopolitan Area Video-On-Demand Service Using Pyramid Broadcasting," *IEEE Multimedia Systems*, vol. 4, pp. 197-208, 1996.

[32] "Real Time Streaming Protocol", *http://en.wikipedia.org*, September 14, 2007.

[33] "Real-time Transport Protocol", *http://en.wikipedia.org*, September 14, 2007.