M.Sc. Engg. Thesis

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

by

Munima Jahan

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

The thesis titled "**Multicast Video-on-Demand Service in Enterprise Networks with Distributed Client Assisted Patching**," submitted by Munima Jahan, Roll No. 100505031P, Session October 2005, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment 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 August 9, 2011.

Board of Examiners

1. _____

Dr. Md. Mostofa Akbar — Chairman
Associate Professor — (Supervisor)
Department of CSE
BUET, Dhaka 1000

2. _____

Dr. Muhammad Masroor Ali — Member
Professor & Head — (Ex-officio)
Department of CSE
BUET, Dhaka 1000

3. _____

Dr. M. Kaykobad — Member
Professor
Department of CSE
BUET, Dhaka 1000

4. _____

Dr. Md. Humayun Kabir — Member
Assistant Professor
Department of CSE
BUET, Dhaka 1000

5. _____

Dr. Mohammad Rashedur Rahman — Member
Assistant Professor (External)
Department of EECS

North South University, Bashundhara, Dhaka

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.

Munima Jahan

Candidate

Contents

List of Figures

List of Tables

Acknowledgments

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

This master thesis work was assigned and supervised by Dr. Md. Mostofa Akbar, Professor, Department of CSE, BUET, Bangladesh. And it would not have been possible to accomplish without the support and guidance of him.

First and foremost, I express my great gratitude to my supervisor Dr. Md. Mostofa Akbar for giving me this thesis work opportunity. I want to give my deepest gratitude to him for his constant guidance and help through all the phases of this work. I always get a strong support from him when I was struggling with the programming. I express my heart-felt gratitude to my supervisor for his illuminating instruction and kind help through all the stages of writing the report.

I would also like to express my gratitude to Dr. Md. Humayun Kabir for his kind assistance and suggestions. He helped me a lot in different aspect of this work and guided me with proper directions in different stage of my work.

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 Rashedur Rahman.

I would also like to thank my family and friends, for their moral support along the way.

Abstract

Multicast Video-on-Demand (VoD) services have become some of the most popular real-time multimedia applications available via the Internet for last few years. Multicast communication with patching enables clients to join an existing multicast session with out any service latency. In this research, we propose a new distributed patching technique DCAP (Distributed Client Assisted Patching) where the initial portion of a movie is distributed to multiple clients to store and provide as patch stream to other clients interested to join an ongoing session within a short time. This scheme significantly reduces the server load without requiring larger client cache space than the similar existing systems such as Client Assisted Patching. We present detailed algorithms for the admission control of patching clients in this research. The policy of distributing the initial part of the movie among different clients is also formulated. The analysis of time requirement for admission controlling, buffer requirement for the patching clients and the bandwidth requirement of the server and link connecting the servers are presented in this thesis. To validate the theoretical results we have done simulation of the proposed system using Parsec, a parallel simulator suitable for simulating different entities in the VoD systems. The detailed analysis on the simulation results reveals that the new system outperforms the previous systems in terms of number of requests served and average waiting although it requires more time in admission controlling for finding suitable patch client during patching. Moreover the system is more scalable and cost effective than many other existing systems.

Chapter 1

Introduction

Video on demand is a technology that provides entertainment on demand to all the subscribers of the service. Video on demand provides customers with informative and entertaining streams of multimedia and video information. A multicast Video-on-Demand (VoD) system allows clients to share a server stream by batching the user requests. Multicast extends the traditional unicast communications with efficient multipoint communications in which data can be sent to a set of destinations simultaneously. Given the rapid development and deployment of multimedia applications and the multireceiver nature of video programs, real-time video distribution has emerged as one of the most important IP multicast applications. It is also an essential component of many current and emerging Internet applications, such as videoconferencing and distance learning.

1.1 Multicast VoD Techniques

A multicast VoD system allows clients to share a server stream by batching their requests, and hence, improves channel utilization. Multicast communications is one of the critical techniques to enhance the VoD service scale by sharing the communication bandwidth.

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.

However, in the typical multicast communication, all receivers are expected to access the same multicast stream at approximately the same time. Therefore, only few customers can be served in the same multicast stream and additional multicast streams are required since most requests issued at different time.

1.2 Batching and Patching

Customer requests arriving within a short time can be batched together and serviced by a single stream is called Batching [2]. 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 in this short interval of time. However, it increases initial service latency that may cause some impatient customers to renege.

Patching is a multicast technique that enables a server to transmit only the beginning of the entire video data to clients and ensures that clients download the rest data of the video from an ongoing stream. By making multiple clients share an ongoing stream, Patching can reduce server network bandwidth requirements for TVoD services. Double Patching [3] ensures that a long patching stream delivers not only essential data for the current client but also extra data for future clients, so it significantly reduces the total amount of video data delivered by all streams.

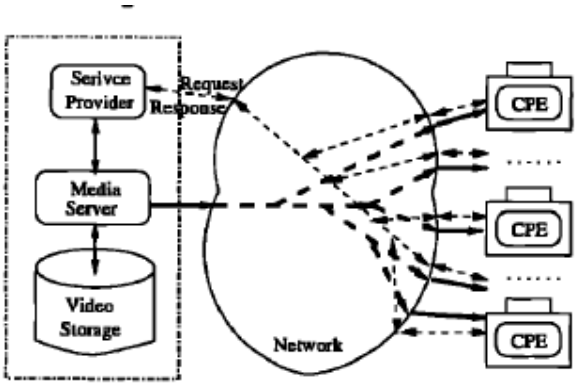


Figure.1.1 A multicast VoD system

1.3 Problem Definition and Previous Work

Multicast networks are well suited for delivery of video. Rather than transmitting duplicate frames to multiple clients, a video source sends one frame and lets the duplication of data occur in a distributed nature throughout the switches of the network. Hence it allows a broad range of applications including entertainment and information services, distance learning, corporate telecasts and narrowcasts etc. to clients across a high-speed network. However, due to the bandwidth intensive nature (usually larger than 1Mbps) of high quality digital video, and the long-lived nature (tens of minutes to a couple of hours) of video content, server and network bandwidths are major limiting factors in the widespread streaming of such videos over the Internet. The problem is further complicated by the fact that clients are plentiful and heterogeneous, and that clients asynchronously issue requests for the same media stream. Particularly for popular clips, a large number of client requests may arrive close together in time relative to the duration of the stream.

Patching [4] eliminates the service latency imposed by the Batching scheme [2]. The objective of Patching is to substantially improve the number of requests each channel can 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.

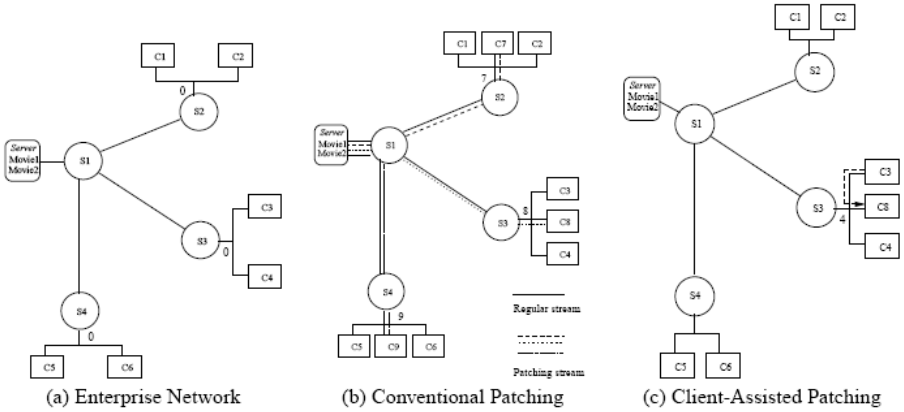


Figure. 1.2. Client Assisted Patching and conventional Patching in an Enterprise Network.

Client Assisted Patching [5], a newly proposed system uses client side cache to reduces the server load. In this approach, all patching channels are provided by the cooperative clients rather than the server itself. Thus, the system alleviates server load and the conserved bandwidth can be used to satisfy more multicast groups. It also increases the throughput and scalability of the system. The minimum buffer requirement in the intervals is same as the conventional patching scheme requires. Client Assisted Patching technique is illustrated in Fig. 1.2. 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.

Client Assisted Patching reduces the server load by using the client side cache. However client cache is limited and only a small portion of the video can be stored in a single client to patch. In our newly proposed system we considered that different clients will store different portion of the movie. Thus we can provide a larger amount for patching. The new scheme proposed in our research is defined as Distributed Client-Assisted Patching.

1.4 Scope and Focus of the Thesis

Since streaming of any multimedia object like high quality video consumes a significantly large amount of network resources, network bandwidth limitation is the major constraint in most of the multimedia systems. So request-to-service delay,

network traffic, congestion and server overloading are the main parameters to be considered in video streaming over the communication networks that affect the quality of service (QoS). Providing VoD service over the internet in a scalable way is a challenging problem. 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. The new patching technique is proposed in this thesis called Distributed Client-Assisted Patching. In this patching technique multiple clients are assumed to serve a requesting client by dedicating their patched portion.

Distribution of storing the initial portion of a multicast session will be based on the arrival time of a new patching client. Thus it is most likely that the earlier clients will store the earlier part and the later client will store the later part of the patched stream.

This research also presents architecture of an Admission Controller for an Enterprise Network to deliver the multicast VoD service. 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.

Testing the scheme in a real world scenario or in a prototype is out of scope of this research. Discrete event simulation model is used to simulate the VoD requests in the proposed system. The simulation results will be validated by comparing the analytical performance measures.

1.5 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 details of video-on-demand service have 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 mentioned in this chapter as well.

Chapter 3, the main chapter of this thesis, illustrates our proposed patching technique and various adaptation techniques in Enterprise Network. The admission control methodologies and client buffer requirement analysis are also presented in this chapter.

Chapter 4 consists of the simulation results and comparative study against Client Assisted 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 and Literature Review

The VoD services allow customers to access the video program with the VCR-like interactive operations over the networks. In normal operation, the customer subscribes a request to the video providers and the providers deliver the selected video program to the customer via the networks. In general, individual video stream is delivered for each customer to provide the true VoD services. Unfortunately, this scheme requires lots of communication bandwidth and is not practical in the real applications. One important strategy to share the communication cost is the multicast communications, in which many customers can share the same video stream. However, this scheme assumes all receivers accessing the same multicast stream at approximately the same time, and hence restricts the number of customers served in the same multicast group.

2.1 Multicast Video-on-Demand Services

A multicast Video-on-Demand system allows clients to share a server stream by batching their requests, and hence, improves channel utilization. Multicast offers efficient one-to-many data transmission and thus provides the foundation for various applications that need to distribute data to many receivers in a scalable manner. It reduces both the server-side overhead and the overall network load. Thus, multicast VoD has good scalability and excellent cost/performance efficiency (See [6] for an excellent survey of multicast VoD services). However, it is difficult to support VCR-like interactivity with multicast VoD and, at the same time, improve service efficiency. There are several proposals [7, 8, 9] to solve this problem.

Multicast communication is one of the critical techniques to enhance the VoD service scale by sharing the communication bandwidth. However, in the typical multicast communication, all receivers are expected to access the same multicast stream at approximately the same time. Therefore, only few customers can be served in the

same multicast stream and additional multicast streams are required since most requests are issued at different time.

2.2 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 Broad casting [2]. 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 [10]. 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 [11, 12, 13] 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 present this approach in the following section as it is related to our research.

2.2.1 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…, 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: up stream link congestion and 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.

2.3 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 [2]. Customer reneging behavior has been discussed in [2]. To eliminate the service latency, several dynamic multicast techniques have been proposed in [16, 4, 1, 17, 18]. In subsequent section we will describe some of the works related with this research.

2.3.1 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 2.1. The scheduling policies shown in Table 2.1 and some of their variants are discussed in [2, 19]. As MFQLF policy maximizes the system throughput without compromising fairness, we have adopted this policy in this work.

Table 2.1: Classification of batching polices

Batching Policies	Working Principle	Objective
Maximum Queue Length First (MQLF)	Requests for the video with the largest number 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 to serve next.	A throughput close to that of MQLF without compromising fairness

2.4 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.

2.4.1 Patching

Patching was introduced in [4] 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.

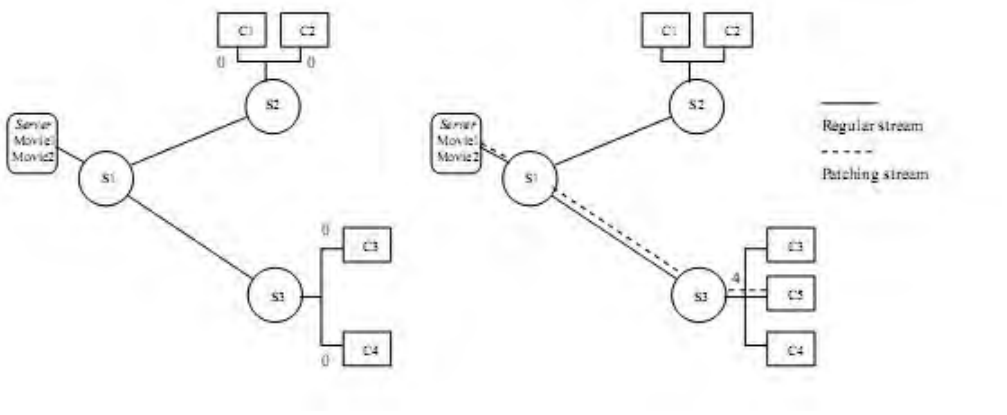


Figure 2.1. Patching: A dynamic multicast technique.

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 [20, 4]. Some modifications of patching technique are discussed in [21, 17]. Two simple approaches of setting the patching window are discussed in [4]. 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 [4]. 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 [1]. 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.

2.4.2 Client Assisted Patching

Client Assisted Patching [5] 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 system alleviates server load and the conserved bandwidth can be used to satisfy more multicast groups. It also increases the throughput and scalability of the system.

Here a service interruption may occur when the patching parent changes multicast group through a VCR request or leaves the session. In this case another patching parent is to be selected by the Admission Controller to deliver the missing portion of the patched 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 be the leaving process, delay might be introduced for the patched client in this situation. The minimum buffer requirement in the intervals is same as the conventional patching scheme requires.

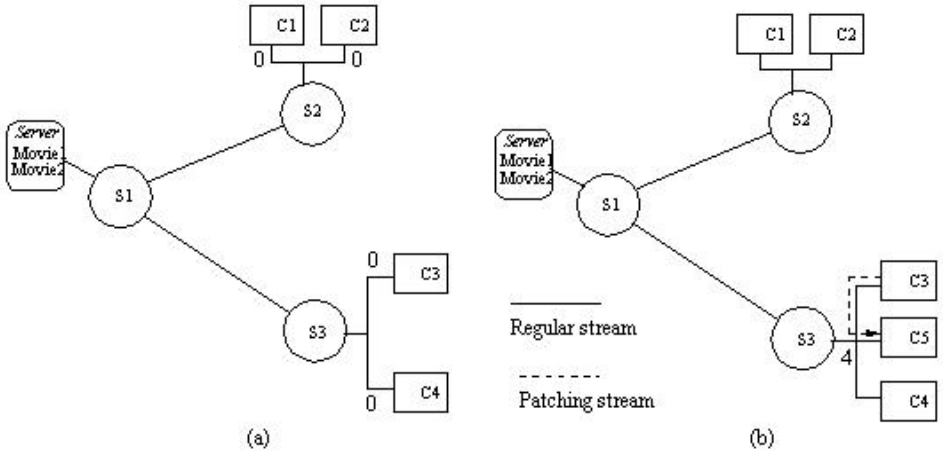


Figure 2.2. Client Assisted Patching.

Client Assisted Patching technique is illustrated in Fig. 2.2. A session is assumed to be started at Time 0 with six clients as shown in Fig. 2.2(a). Patching window is assumed to be 5 time units long. At Time 4, Client C_8 requests the same movie. As the

request time is within the time interval of the patching window, the Admission Controller will select a nearby client to supply the patching stream. At this time, Client C_3 is selected as shown in Fig. 2.2(b) to supply the patching stream to the newly arrived Client C_8. 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.

2.4.2.1 Architecture of the Client Assisted Patching

The architecture of the proposed video-on-demand service in an Enterprise Network is shown in Figure 2.3. The system components are described below.

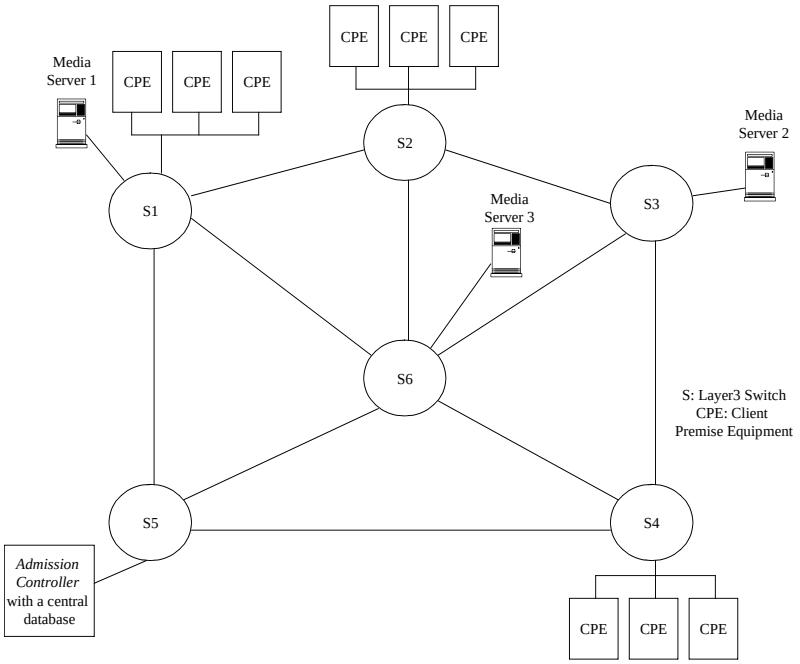


Figure 2.3. The architecture of a VoD system in Enterprise Network.

- **Enterprise network:** We consider a privately owned small enterprise network with network switches (nodes) and links between them. In Figure 3.3, S_1 through S_6 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 play out 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 whereabouts 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.

2.4.2.2 Basic principles of the Client Assisted Patching

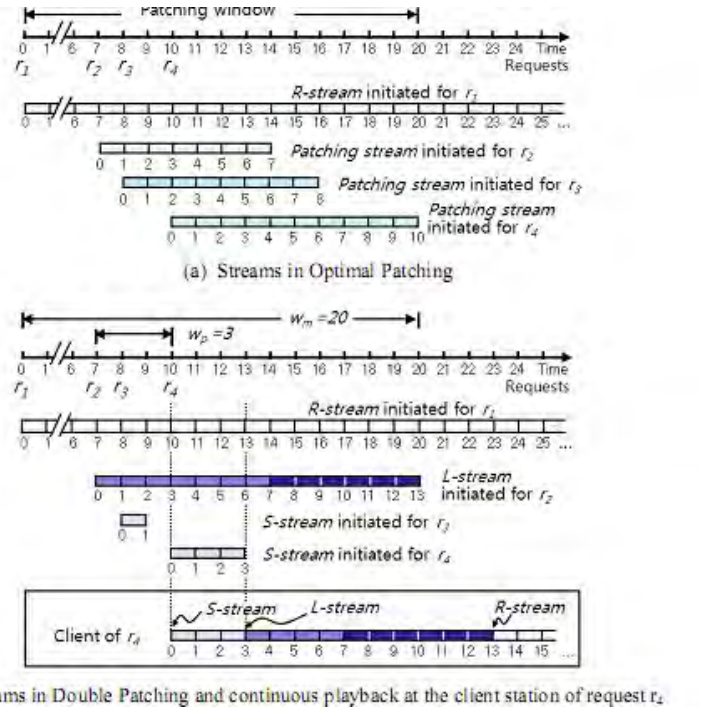
The architecture of the Client Assisted Patching is described in the previous section where the Enterprise Network is composed of several layer 3 switch nodes, servers, clients and a powerful Admission Controller. Basic principles of the proposed system are demonstrated in the following points:

- Clients will place their requests to the Admission Controller and subsequently served by the video servers.
- After receiving a request the ADC will batch it for a fixed time defined as batch window if there is no ongoing multicast session for the requested video.
- Each client joined in the multicast session will store the initial portion of the movie.
- A request will be patched if there exist a multicast session for the same movie and it comes within the interval 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*.
- The newly joined client will also store the initial portion of the movie according to the distribution technique described in next section depending on its arrival time.
- If a client requests movie after the patch window limit but with in two patch window the ADC will select two patch parents for the client. The first window size will be provided by the first parent and the rest will be provided by the next parent.
- In our proposed system, a client will supply patching stream to at best a single client at a time. Thus, clients in the proposed system have a limited responsibility that makes the management task easier and efficient.
- 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.
- If service interruption is occurred ADC will select a new patch parent for the client.

2.4.3 Double Patching

In double patching [3] technique two patch streams are used, a long stream (L-stream) and a short stream (S-stream). As the skew between the latest regular stream (R-stream) and a new request becomes longer, the length of a patching stream also becomes longer. In order to shorten the length of a patching stream, Double Patching introduces a long patching stream (L-stream). A client receiving an L-stream has to share the latest R-stream; the L-stream can be shared with future clients. It uses two time thresholds: a multicast window and a patching window. The multicast window (W_m) is the minimum interval between two sequential R-streams; the patching window (W_p) is the minimum interval between two sequential L-streams. The patching stream in Optimal Patching is called a short patching stream (S-stream) in Double Patching. The algorithm that a server uses to schedule a new stream for the current client is as follows:



(a) Streams in Optimal Patching

(b) Streams in Double Patching and continuous playback at the client station of request r_4

Fig. 2.4 Optimal Patching vs. Double Patching

- If the skew between the current client and the latest R-stream is greater than W_m, it schedules a new R-stream.
- Otherwise, it schedules a new L-stream /S-stream as follows:
    - If the skew between the current client and the latest R-stream/L-stream is less than or equal to W_p, it schedules a new S-stream to deliver the beginning of the entire video data that the client has not received from the latest R-stream/L-stream. In this case, to play back the entire video data, the client plays back data in the following order:
      S-stream, L-stream if the client has to share it, and R-stream.
    - Otherwise, it schedules a new L-stream to deliver the beginning of the video that the client has not received from the R-stream and the following data of the video that will be continuously played back during $2 \times Wp$ time units after playback of the beginning. In this case, the client first plays back the data from the L-stream and then the data from the R-stream.

Fig. 2.4 shows how to schedule streams for client requests in Optimal Patching and Double Patching. As shown in Fig. 2.4, the length of an L-stream is $(2 \times W_p)$ longer

than that of a corresponding patching stream in Optimal Patching. However, because W_p is much smaller than that in Optimal Patching, S-streams that are scheduled for clients sharing the same L-stream become very short. As a result, Double Patching can decrease the total amount of transmitted video data by 50% compared with Optimal Patching, and significantly reduce the server network bandwidth requirements [3].

2.4.4 Expanded Patching Technique using Four Types of Streams (XP4S)

A multicast technique that completely prevents a server from transmitting unnecessary video data like double patching and uses four types of streams: regular stream (R-stream), patching stream (P-stream), short patching stream (S-stream) and linking stream (LK-stream). An R-stream, an S-stream, W_m, and W_p in the proposed XP4S [22], have the same meanings as in Double Patching.

A P-stream is scheduled in the same manner that Double Patching schedules an L-stream, except that it does not deliver extra data. In other words, a P-stream delivers the beginning of the entire video data that the current client has not received from the latest R-stream and it does not deliver extra data for possible future clients. If a client request is within the patching window of the latest R-stream/P-stream, an S-stream is scheduled to make it share the R- stream/P-stream. In XP4S, a VoD server can multicast a single LK-stream to all clients that have received their respective S-streams and shared the same P-stream. Using LK-streams, the proposed XP4S completely prevents the server network bandwidth wastage that can be generated by the extra data of Double Patching. As a result, server network bandwidth requirements for TVoD services can be reduced [22]. Using the same server network bandwidth, our technique always has better average service latency and client defection rate compared with Double Patching.

2.5 Range Multicast

Range Multicast is a new communication paradigm for VoD applications [1]. 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 2.5. 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 2.5 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.

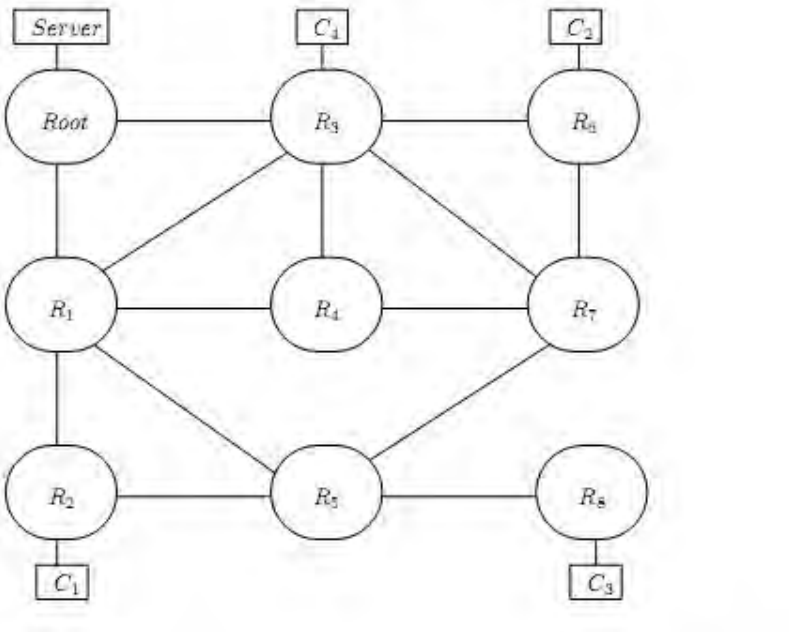


Figure 2.5: Overlay topology of range multicast enabled routers

2.6 Client-To-Client Streaming Scheme for VoD Applications

Client-to-client streaming scheme for VoD application is an [23] is an efficient client-to-client streaming approach to cooperatively stream the video using chaining technique with unicast communication among the clients. This approach considers two major issues of VoD:

i) Prefix caching scheme to accommodate more number of videos closer to client, so that the request-service delay for the user can be minimized.

ii) Cooperative proxy and client chaining scheme for streaming the videos using unicasting.

This approach minimizes the client rejection rate and bandwidth requirement on server to proxy and proxy to client path. The simulation results show that the proposed approach achieves reduced client waiting time and optimal prefix caching of videos minimizing server to proxy path bandwidth usage by utilizing the client to client bandwidth, which is occasionally used when compared to busy server to proxy path bandwidth.

The main goal of this streaming scheme is to make each client act as a server while it receives the video, so that the available memory and bandwidth of the clients can be utilized more efficiently. The un-scalability of traditional client-server unicast VoD service lies in the fact that the server is the only contributor and can thus become flooded by a large number of clients submissively requesting the service.

2.7 Some other related research on Multicast VoD services

Many other researches have been done on multicast Video on Demand services. Some of them are described shortly in the following section:

An enhanced client-centric approach for efficient video broadcast is described in [24]. P2Cast—an architecture that uses a peer-to-peer approach to cooperatively stream video using *patching* techniques, while only relying on unicast connections among peers. The following two key technical issues are addressed in P2Cast: (1) constructing an application overlay appropriate for streaming; and (2) providing continuous stream playback (without glitches) in the face of disruption from an early departing client. P2Cast can serve many more clients than traditional client server unicast service, and that it generally out-performs multicast-based patching if clients can cache more than 10% of a stream's initial portion. Disruptions are handled by delaying the start of playback and applying the shifted forwarding technique. The threshold in P2Cast, i.e., the length of time during which arriving clients form a single session, can serve as a "knob" to adjust the balance between the scalability and the clients' viewing quality.

MegaDrop: A Cooperative Video-on-Demand System in a Peer-to-Peer Environment is a new technique proposed in [25]. This paper describes a fully decentralized VoD service via P2P techniques, which is referred to as the MegaDrop

system. The MegaDrop system not only takes active peers into consideration but also provides mechanisms for discovering inactive peers that contain desired media objects. Evaluation results of the MegaDrop system show that the architecture performs more efficiently when the more inactive peers involving to provide media blocks.

A New Zero-Delay Video-on-Demand Scheme is a technique proposed to solve the server delay of high-performance static streaming scheme [26]. It contains the outstanding thoughts from two efficient schemes: GEBB and patching scheme. The scheme allocates (optional) the number of channels to minimize the bandwidth consumption. The scheme can provide zero-delay VoD service with sending the first part of video via unicast channel when the client request arrived. To improve the performance, the residual part of video is delivered by using revised GEBB. Since most services require the server to deliver only a small leading part of the video, the server can serve many more clients per time units. Simulation results show that the schemes can minimal bandwidth consumption effectively.

P-chaining: a practical VoD service scheme autonomically handling interactive operations proposes a service scheme based on chaining [27], in which clients as well as the server provide streaming services. In the proposed scheme, services are provided by unicast and managed locally using node lists. In addition, proposed scheme can support frequent VCR operations without incurring significant overhead in the server workload. The proposed scheme reduces server workload significantly and the frequent VCR operations can be served smoothly without causing too much overhead.

Proxy-assisted scalable periodic broadcasting of videos for heterogeneous clients is proposed in [28]. In this paper, a scheme is proposed to significantly reduce the waiting time of all heterogeneous clients, without the need for any additional backbone bandwidth. This scheme uses a proxy buffer within video-on-demand systems using PB. In the proposed system, the server broadcasts a video using one of the traditional PB protocols. Simultaneously, the proxy receives the stream from the server and stores it in its local buffer, then broadcasts the stored data to the clients in its local network. Because the proxy provides extra, transparent channels to the server,

clients are likely to reduce their reception bandwidth requirements through the use of efficient reception schedules using the extra channels.

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 [29] (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 modelled by time stamped 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 [30] 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 (applicationtype, 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

Distributed Client Assisted Patching

Client Assisted Patching [5] reduces the server load by using the client side cache to store the initial portion of the movie. In this approach, all patching channels are provided by the cooperative clients rather than the server itself. Thus, the system alleviates server load and the conserved bandwidth can be used to satisfy more multicast groups.

The patch window during the multicasting is defined by the duration when a particular client can join the multicast group. The number of users in a multicast group depends on the size of window. But increased window size requires larger buffer in the clients and client buffer is limited. So storing the same portion to all clients can provide only a limited portion as patch to other clients. If the storage required for storing the initial portion of the movies can be distributed among a group of clients then it is possible to support a larger window size with the same amount of buffer in the clients. Thus it requires less buffer and a large multicast group can be supported with smaller network bandwidth.

3.1 Distributed Client Assisted Patching

In the Distributed Client Assisted Patching storing the initial portion of the session is distributed among the participating clients. Some observation about the new scheme can be described as follows:

- The clients are allowed to join an ongoing multicast session with in $2W$ time from the beginning of the session. Where W is the patch window in client assisted patching. Thus the effective patching window is doubled.
- Each client in a multicast session will store a portion of the video with length W depending on its arrival time. The distribution policy is described in the later section.

- If a client requests for a movie after starting of multicast session the ADC will select one or two patch parents for the client. Different parts of the patching stream may be supplied by different clients.
- In the new scheme the patch streams are distributed among the clients. It means that a client will not store the whole patch stream and the ADC must find multiple clients for patching. The main problem here is the reduced probability of finding a patch client. Thus in some cases waiting time will be increased and the percentage of served will be decreased. But as we are getting larger patch window we can effectively get more clients to be served with single movie stream.

3.2 Distribution policy

In our proposed system each client in a multicast session will store a portion of the initial part of the movie with length W depending on its arrival time. The distribution policy of the part of the movie to be stored in patched clients is described in the following table.

Table 3.1: Distribution policy used in the proposed system.

Arrival time, t	**Part of the Movie Stored, B**
$0 \leq t < W/2$	$0 \leq B < W$
$W/2 \leq t < W$	$W/2 \leq B < W+W/2$
$W \leq t \leq W+W/2$	$W \leq B \leq 2W$
$W+W/2 < t \leq 2W$	*None*

the distribution policy is explained with proper reasoning as follows

- The clients coming within the time 0 to $W/2$ will store the initial part of the movie from starting to W time duration. Here we should mention that our scheme is a combination of both batching and patching. So the regular multicast stars with a group of clients in the batch who will also store the first part of the initial portion. So even if no patch requests come within 0 to $W/2$ time interval the first part of the patch stream is always available to some of the clients. That is why the initial part indicating time duration from 0 to $W/2$ is not stored by other group of clients coming later.

- Again the clients joining with in the time interval $W/2 \leq t < W$ will store the initial part from $W/2$ to $W+ W/2$ and clients coming with in the time interval $W/2 \leq t < W+W/2$ will store the part from $W \leq B < 2W$. Here we can see that the initial part from $W/2$ to W and W to $3W/2$ are stored by the two different groups of clients. This has been done to ensure that there will be less chance to miss any part of the initial portion.
- Part of the movie from $3W/2$ to $2W$ is stored by only one patch client group with arrival time $W \leq t < 3W/2$. This part of movie is required only by the clients arriving $3W/2 < t < 2W$ during the patching. It is assumed that this is sufficient to serve this part as it is required by less number of clients compared to other patching clients.
- The client coming at t ($t > 2W$) need not to store any part of the movie because clients arriving after this interval will not be allowed for patching. These clients will be batched for next multicast session.
- If no clients joins the ongoing session with in the interval $W/2 \leq t < W$ no clients will store the part from $W/2$ to $W+W/2$. But still we can get the portion $W/2$ to W from the first group and the portion W to $W + W/2$ from the second group.
- Again if no client comes within $W \leq t < W+W/2$ we can get the initial part 0 to $W+W/2$ from the first two groups but we may miss the portion $W+W/2$ to $2W$ as no client stores this part. In this case patching will be served directly from the server.
- If no client joins with in the time interval $W/2 \leq t \leq W+W/2$ then only the initial part from 0 to W is available. Though this would be a very rare situation if this happens then patching will be allowed for 0 to W interval and rest of the portion will be patched directly from the server.

3.3 An Illustrative Example

Suppose a multicast session starts with 5 clients C_0, C_1, C_2, C_3 and C_4. All these five clients will store the first 10 min of data if the client buffer is limited to store 10 min of movie data.

Time *t*=0:

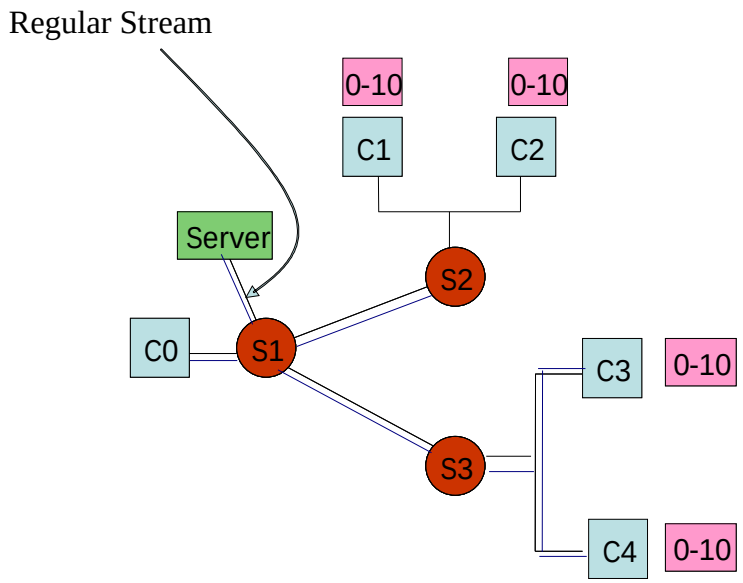


Figure 3.1 Example of the Distributed Client Assisted Patching at time *t*=0.

Time *t*=5:

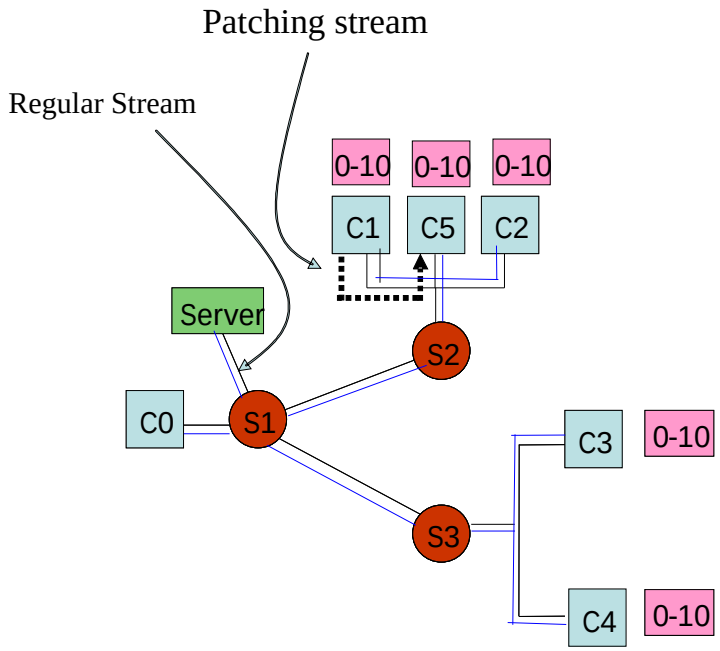


Figure 3.2 Example of the Distributed Client Assisted Patching at time *t*=5.

Suppose a client C_5 joins at time *t*=5. C_5 will get the patch stream from C_1 and will store first 10 min of data.

Time *t*=8:

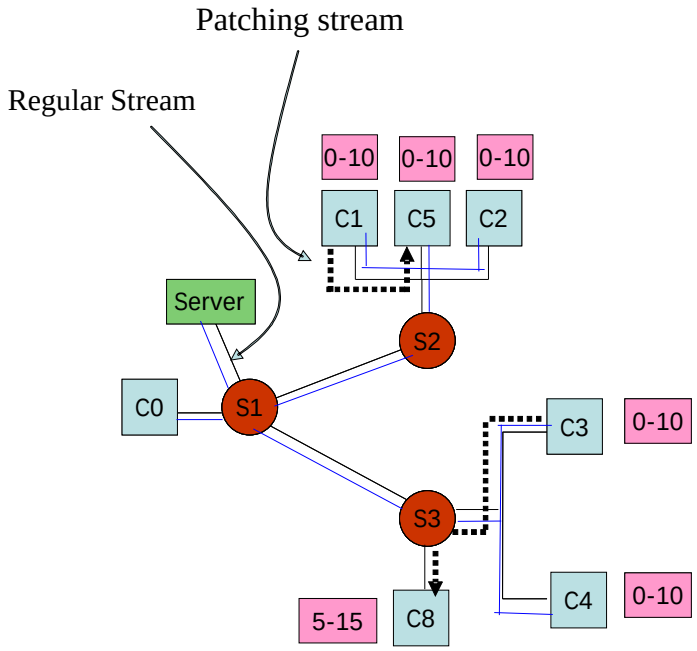


Figure 3.3 Example of the Distributed Client Assisted Patching at time *t*=8.

C_8 requests for the movie at time *t*=8. C_8 will get patch from C_3 and will store 5 to 15 min of data.

Time *t*=11:

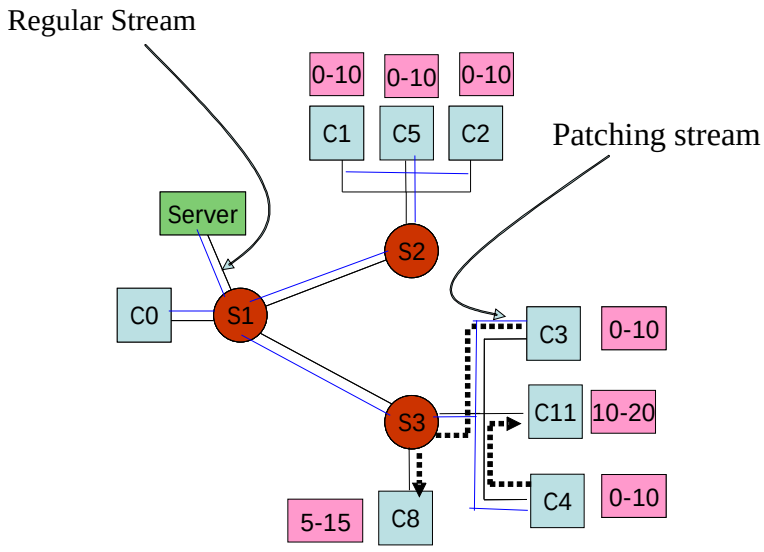


Figure 3.4 Example of the Distributed Client Assisted Patching at time *t*=11.

C_{11} comes at time 11 min. As the client C_{11} comes at time $t>W$ but $t<2W$ it needs two patch parents. C_4 will be selected as patch parent for the first 10 min and then C_8 will

be selected as patch parent for next one min data. Again C_{11} will store the initial stream from 10 to 20 min.

Time *t*=21:

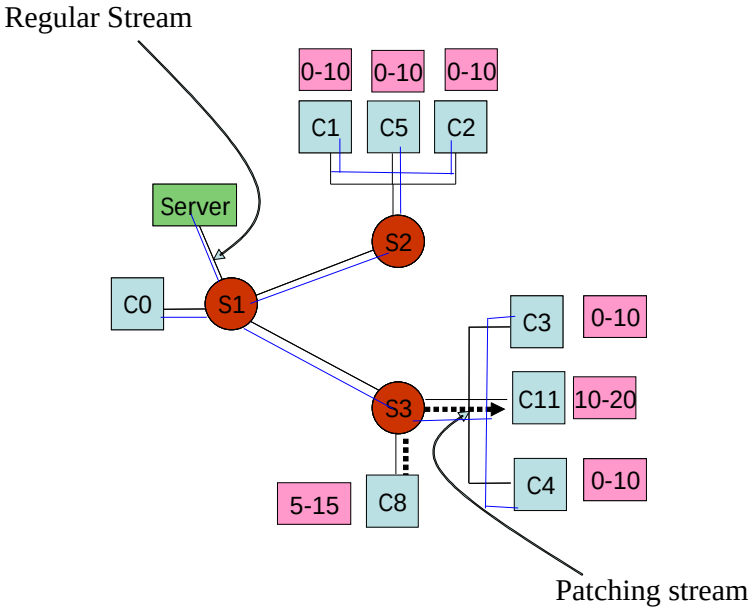


Figure 3.5 Example of the Distributed Client Assisted Patching at time *t*=21.

At time *t*=21 C_3 will complete the patching stream for C_8. Also C_4 will complete patching of the first part for C_{11} and C_8 starts giving the second patch stream to C_{11} as shown in the Figure 3.5.

Time *t*=22:

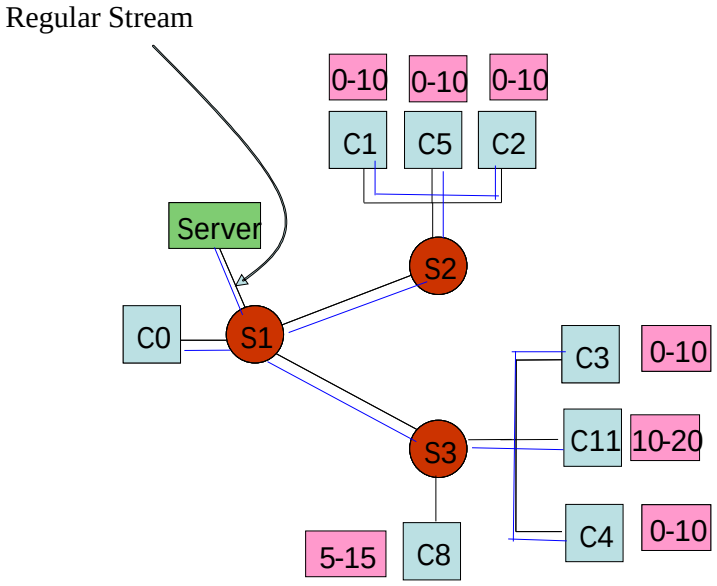


Figure 3.6 Example of the Distributed Client Assisted Patching at time *t*=22.

At time t=22 C_8 completes the patch stream for C_{11}. Also the window size is over and no patch request will be accepted any more.

3.4 Motivation Example

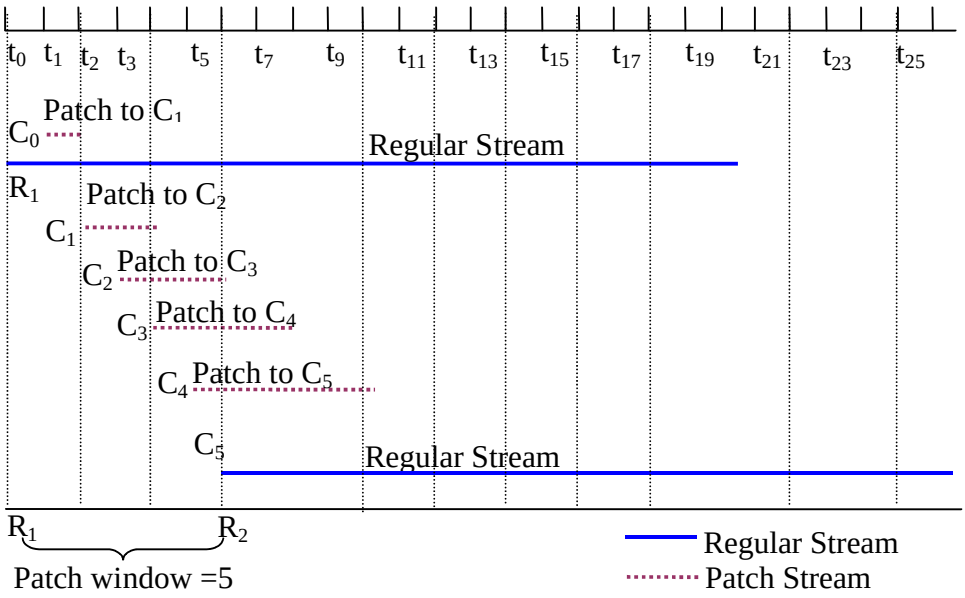
Let us consider an example. The length of the video is 20 minutes. Each client has enough disk space to store necessary patch stream. The arrival rate is one request per minute. Suppose the client buffer size is 5 minute.

Client Assisted Patching:

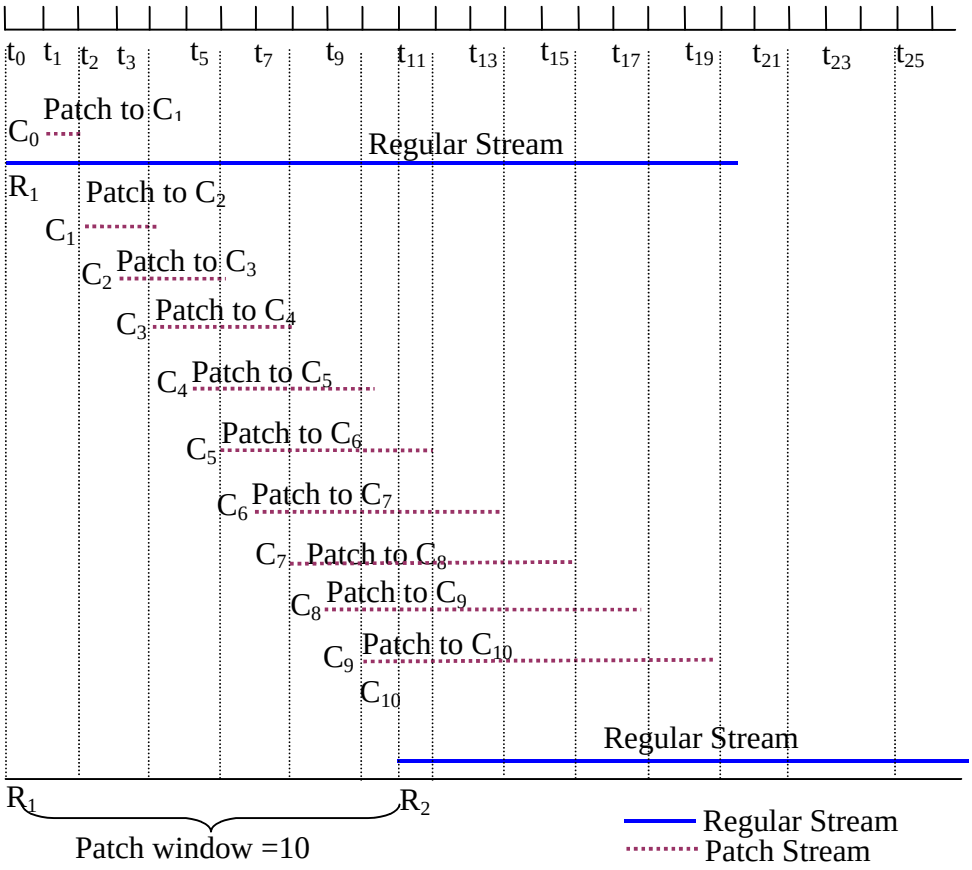
The patch request will be allowed till W=5 min.

Suppose the first client requests for the movie at time 0. So a regular multicast will be initiated at time zero and will be continued for next 20 min. this is the only data stream provided by the server. The next 4 users will be given patch stream from different clients.

The data delivered by server D_s=20 min and the data delivered by the client is D_c= $\sum_{i=1}^{5} i$ =15 min. This data is shared by 5 users for one multicast session. Again with in 20 min there will be four multicast sessions which needs to send 4×20 =80 min of data by the server to serve 20 users. So the average data required for individual client is 4 min from server and 3 min from patch client.



(a) Data transmitted by Client Assisted Patching (CAP).



(b) Data transmitted by Distributed Client Assisted Patching (DCAP).

Figure 3.7 Comparing CAP and DCAP

Distributed client assisted patching:

In this system first client will initiate the regular stream and rest 9 user will be patched within $2W$ =10 min the effective window size. The data delivered by the server will be D_s=20 min and the data delivered by the clients will be $\sum_{i=1}^{9} i$ =45 for 9 clients in one multicast session. Again within 20 min there will be only two regular multicasts which need only 2×20=40 min of data to be delivered by the server to serve 20 users. So we can see that the same number of user can be served with half of the server bandwidth. Here the average data required for each client is 2 min from server and 5 min from client.

Thus we can see that the distributed Client Assisted Patching requires 50% less data from the server. So it is obvious that the server bandwidth requirement is reduced 50% in our system.

3.5 Analysis for Server Bandwidth Requirement

In our analysis we will refer to the amount of data in terms of their play back duration. Let us make the following assumptions:

- Both batching and patching are combined in this system.
- The batching window size is W_B and the patching window size is W_P and all clients have enough buffer to store necessary patch stream.

During the patch window of a particular multicast session there will be only one regular multicast stream transmitted by the server. The rest will be patched by the client. During the batching no new clients will be served and they will get the stream in the next multicast session. So the total data delivered by the server during one multicast session for the *i*th movie M_i is

$$D_s = L_i \tag{3.1}$$

Where, L_i is the length of the *i*th movie M_i. The patch streams for the late clients will be provided by the cooperative clients.

If *k* patching streams are initiated between *t* and $t+\Delta t$, then total data transmitted by *k* patch stream can be approximated as kt if Δt is negligible. If the probability of initiating *k* patch stream during Δt is $P(k, \Delta t)$ then the total data delivered by the clients between *t* and Δt is $\sum_{k=1}^{\infty} ktP(k, \Delta t)$. To calculate the mean total amount of video data delivered by a multicast group we can partition (0, W_p) into $W_p/\Delta t$ small time segment. Then total data transmitted by the client

$$D_c = \sum_{t=1}^{\left\lfloor \frac{w_p}{\Delta t} \right\rfloor} \sum_{k=1}^{\infty} ktP(k, \Delta t) \tag{3.2}$$

To determine the expression of the probability we assume that the multicast initiation process is Poisson with rate λ. The probability density function is $f_x = \lambda e^{-\lambda x}$, where *x* indicates the first time of the patch client arrival and f_x indicates the probability that the first client arrives at time *x*. Now we can derive $P(k, \Delta t) = (\lambda \Delta t)^k \frac{e^{-\lambda \Delta t}}{k!}$. D_c can be derived as follows:

$$\sum_{k=1}^{\infty} ktP(k, \Delta t)$$

$$= \sum_{k=1}^{\infty} kt(\lambda\Delta t)^k \frac{e^{-\lambda\Delta t}}{k!} = t\sum_{k=1}^{\infty}(\lambda\Delta t)^k \frac{e^{-\lambda\Delta t}}{(k-1)!} = te^{-\lambda\Delta t}\sum_{k=1}^{\infty}\frac{(\lambda\Delta t)^k}{(k-1)!} = te^{-\lambda\Delta t}\lambda\Delta te^{\lambda\Delta t} = t\lambda\Delta t$$

If we set Δt equals to 1 second then we get

$$D_c = \sum_{t=1}^{W_P} t\lambda = \lambda\frac{W_P(W_P+1)}{2} \tag{3.3}$$

Since the client request rate is λ, the patch window is W_p and the batch window is W_B the mean interval between two multicast group is $\tau = W_B + W_p$.

Now consider N multicast session in a time interval T. Thus $N = T/\tau$. Total data stream supplied by these N streams will be $D_{st} = L_{avg} \times N$

Thus the server bandwidth requirements

$$= D_{st}/T = \frac{L_{avg}\times N\times b}{T} = \frac{L_{avg}\times b}{\tau} = \frac{L_{avg}\times b}{W_B + W_P} \tag{3.4}$$

Again the patching by the client will be continued within patch window W_p. So the link bandwidth required for patch streams which is provided by the clients to other clients can be measured as

$$\text{link bandwidth} = \frac{D_c}{W_p}b = \lambda\frac{(W_P+1)}{2}b \tag{3.5}$$

Where b is the video play back rate and L_{avg} is the average movie length.

If we represent the patch window of Client Assisted Patching as W_{cap} we can write

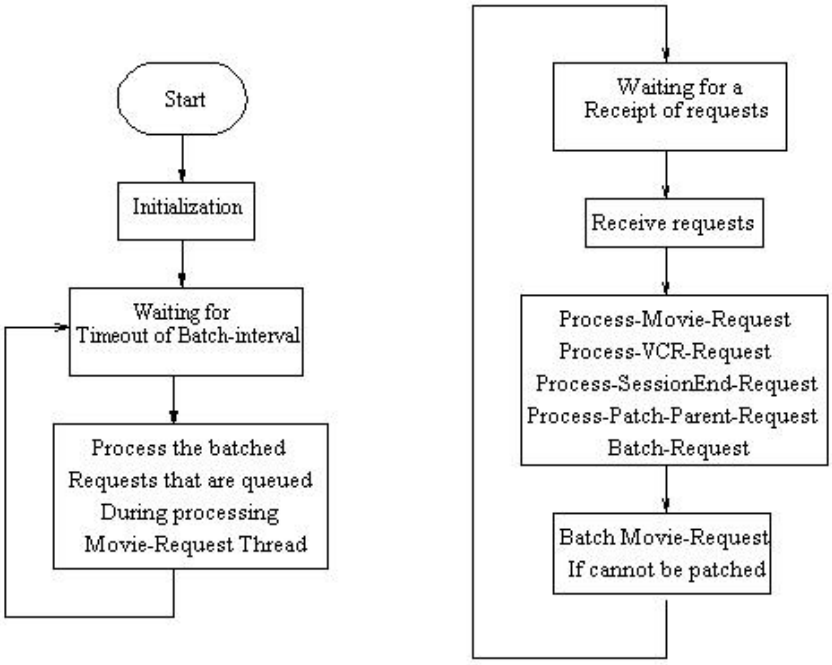
$$W_p = 2W_{cap} \tag{3.6}$$

From the equation (4) we can see that the server bandwidth requirement is inversely proportional to the patch window. So the larger the patch window the less will be the requirement of server bandwidth. Again distributed Client Assisted Patching supports a patch window that is double of the patch window supported by the Client Assisted Patching system. So we can say that our system saves about 35% bandwidth than that of Client Assisted Patching if $W_B = W_{cap}$.

3.6 Algorithm for Distributed Client Assisted Patching

The Admission Controller plays a very important role in the proposed VoD system. The basic algorithm of admission control is same as the admission controller of Client Assisted Patching system [5].The flow chart of the admission control procedure is illustrated in Figure 3.8 which is already presented in the Client Assisted Patching [5].

Here we will not describe the admission control procedure in details. We will only describe the data structure and algorithm that we have added in our system.



(a) Thread for batch processing (b) Online request receiving and processing thread

Figure 3.8 The flow chart of the Admission Control.

3.6.1 Data Structures used in the proposed system

Some of the data structures used in our program are described bellow:

clientPatchInfo:This is a structure data type to store the information about the patch session of individual clients. That means it stores the required patch information and the node selected as patch parent.

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

session_Time: a double value representing the time when a specific client joins a multicast session which is a member of the structure *clientPatchInfo*.

WindowSize Wp: The threshold value up to which a client can join an ongoing session. Here window size is double of the initial part stored in each client.

3.6.2 Procedures and Algorithms

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 is used to construct these shortest path trees.

clientPatchInfo FindClientPatchDetail (*sessionTime, movieId, graph*[*NO_OF_SERVERS*][*NODES*], *clientSource, clientId*)*:* this function checks whether the client can be given patch stream. That means it checks whether the request came within path window time. It also checks if single patch is enough or two parents are required. The function returns the suitable patch session with necessary resources including patching parents if patch session is available. Otherwise it returns a null session to indicate that patch is not possible.

Void StoreInitialPart(*clientId, nodeId, sessionTime*)*:* the procedure is used to store the specific part of the initial portion to the client buffer depending on the session time that means the time when the client is joining the session.

patchClientNodeResource selectPatchingParent(*network, source, sessionTime, clientId*)*:* this procedure returns the patch parent that has the necessary patch stream depending on the session time. If there is more than one patch parent available then it returns the one which requires less resources.

Procedure Process-Movie-Request(*client : C, movie-id : M, session_Time: T*)

/* *The procedure will be invoked when MOVIE-REQUEST message has been accepted* */

session ← *FindClientPatchDetail*(*C, M, T*)

patchingParent ← *session.PatchParents*

if (*session* ≠*Nil and patchingParent* ≠*Nil*) then /* *Client C is patched* */

Admit C in session /* *Allocation of resources for Client C* */

Distribute_Patch_Stream(*C, Current_Timet, M*)

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 FindClientPatchDetail:(*Network N, client C, movie-id M, session_Time T*)

returns *patchSession*

if $T < W_p$

// *single patch_parent is required*

patchSession .PatchParents[0]← *selectPatchingParent*(*N, C, M, T*)

else if $Wp < t < 2*W_p$

// *double patch_parent*

patchSession .PatchParents[0]← *selectPatchingParent*(*N, C, M,* W_p)

patchSession .PatchParents[1]← *selectPatchingParent*(*N, C, M, T*)

end if

resourcse←Find_Resource for patchSession .PatchParents

if *resource ≠Nil* and *patchSession .PatchParents ≠Nil*

return *patchSession*

else

request rejected

end if

end Procedure FindClientPatchDetail

Procedure selectPatchingParent:(*Network n, client* C_k *, movie-id M, sessionTime t*)

returns *patchingParent*

newTree ← $dijkstra(n, source\ node\ of\ C_k)$

min $\leftarrow \infty$

for *each client Cp of n* **do** /* *Finding shortest path parent* */

$d \leftarrow distance(newTree, C_p, C_k)$

Select a client Cp such that

1. C_p *has the necessary patch stream.*

2. C_p *is not serving as a patch parent. // To reduce the load of a patching client.*

3. *min>d.*

set C_p *as patch parent*

min $\leftarrow d$

$C_j \leftarrow C_p$

end for

return C_j.

end Procedure selectPatchingParent

Procedure StoreInitialPart (*client Ck, SessionTime t, movie-id M***)**

if $0 \leq t < W/2$

//Ck will store from 0 to *W*

Store the initial part of M from 0 to *W in the client buffer*

else if $W/2 \leq t < W$

//Ck will store from *W*/2 to *W*+*W*/2

Store the initial part of M from W/2 to *W*+*W*/2 *in the client buffer*

else if $W \leq t \leq W+W/2$

//Ck will store from W to 2W

Store the initial part of M from W to 2*W in the client buffer*

end if

end Procedure StoreInitialPart

3.7 Performance Analysis

In this section we will discuss the complexity analysis of our program and also the buffer requirements of the clients in the subsequent section.

3.7.1 Complexity Analysis

The complexity of Admission-Control in Client Assisted Patching is $O(sE \log V + m^2 + n_c E \log V + n^2_c E)$ which is the sum of the complexity of two threads Batched-Requests-Processing and Online-Requests-Processing and the procedure Init-Admission-Controller . Here, *s* is the number of servers, *m* is the number of movies, *V* is the number of nodes, *E* is the number of edges, n_r is the number of clients that are seeking admission and n_c is the number of clients already admitted in multicast session.

In the Distributed Client Assisted Patching unlike the Client Assisted Patching the users do not store the same initial portion. Rather three different groups of users store

the different portion of the initial part. Also in this system the users are allowed to join an ongoing multicast session within a patch window which is double of the patch stream stored in a single client. That means the system allows a window size which is twice as much as used in the client assisted patching.

In distributed Client Assisted Patching a user coming within the single patch window requires only one patch parent for the first part of the initial portion of the movie stored in one client buffer. Here single patch window refers to the half of the supported threshold with in which a client is allowed to join an ongoing multicast session. In this case admission controller invokes the procedure *selectPatchingParent* once to get a single patch parent. Where the complexity of the procedure is $n_c^2 E$. In this case our system takes the same computational time as client assisted patching.

But a client requesting for the same movie after the single patch window time but within the double window size which is the effective patching window in our system requires two patch parents. The first parent will provide the first part of the required patch and the second parent will provide the rest. So the admission controller has to invoke the procedure *selectPatchingParent* twice to get two different patch parents. Thus the complexity of this procedure will be $2n_c^2 E$. So the total time complexity of the proposed system can be described as: $O(sE \log V + m^2 + n_c E \log V + 2n_c^2 E)$. So we can see that the time complexity of our system looks same as that of the client assisted patching. But as the procedure *selectPatchingParent* needs to be invoked twice in case of two different patch parents the time requird to execute the program will be higher than that of Client Assisted Patching.

3.7.2 Buffer Requirements

In conventional patching scheme patching stream is played back first and regular stream is stored in the client buffer for the future play out. The limit up to when a newly arrived client will be patched after a multicast session starts is called patching window [4]. If patching window size is W time units then the upper bound of buffer requirement is WM. where, M is the average amount of movie data stream per time unit.

In Client Assisted Patching scheme each client caches the initial part of the movie stream up to the time period of patching window to provide patch stream to future

clients. The upper limit of the required buffer is still *WM* in Client Assisted Patching [5].

In our scheme users are allowed to join an ongoing multicast session with in 2*W* time after the session starts. The initial part of the movie is distributed to different client such that each client will store maximum *W* time unit for serving future clients.

To estimate the buffer requirements let us consider the following situations assuming a multicast session has started at time t_0.

i) A client requested the same movie in the time interval ($t_0, t_0 + W/2$).

ii) A client requested the same movie in the time interval ($t_0 + W/2, t_0 + W$).

iii) A client requested the same movie in the time interval ($t_0 + W, t_0 + 2W$).

For Case (i), if a client requests the movie at time t_1 and $t_1 - t_0 \leq W/2$, 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. The patching stream makes up the missing portion by $(2t_1 - t_0)$ where $(2t_1 - t_0) \leq (t_0 + W)$. Streaming at different time interval can be shown in the figure. If *b* is the cache required for the newly admitted client and *L* be the average movie duration the possible equations for different situations are shown in the Figure 3.9.

- A client coming at time t_1 where $t_1 < t_0 + W/2$ needs to store both the initial part and the regular stream till $2t_1 - t_0$. The buffer requirement for this interval is shown by the equation 3.7 which can be at best *WM*.
- After the time $2t_1 - t_0$ the missing portion will be made up but the client needs to store the initial part 0 to *W* for patching. This situation is described by the equation 3.8.
- A client will not make the buffer free till there is a chance to be selected as patch parent. In the worst case a client will join as patch client at $t_0 + 2W$ and will be able to get the stream up to $t_0 + 3W$. This is shown by the equation 3.9.
- After $t_0 + 3W$ the client buffer used for storing the initial part can be freed. Only the missing portion will be stored in the client buffer. The buffer required for this state is shown by the equation 3.10.

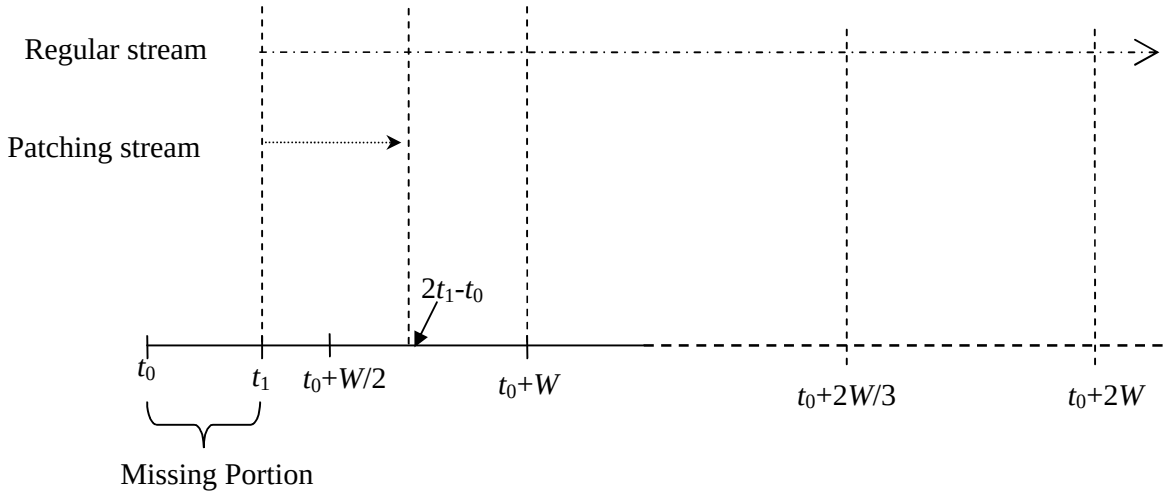


Figure 3.9 Buffer requirement analysis for Case (i).

$$b = 2(t - t_1)M \qquad t \in [t_1, 2t_1 - t_0] \tag{3.7}$$

$$b = 2(2t_1 - t_0 - t_1)M + (t - 2t_1 + t_0)M \qquad t \in (2t_1 - t_0, t_0 + W]$$

$$= (2t_1 - 2t_0 + t - 2t_1 + t_0)M = (t - t_0)M \tag{3.8}$$

$$b = 2(t_1 - t_0)M + (t_0 + W - 2t_1 + t_0)M \qquad t \in (t_0 + W\ ,\ t_0 + 3W]$$

$$= (2t_1 - 2t_0 + W - 2t_1 + 2t_0)M = 3WM/2 \tag{3.9}$$

$$b = (t_1 - t_0)M \qquad t \in (t_0 + 3W, L] \tag{3.10}$$

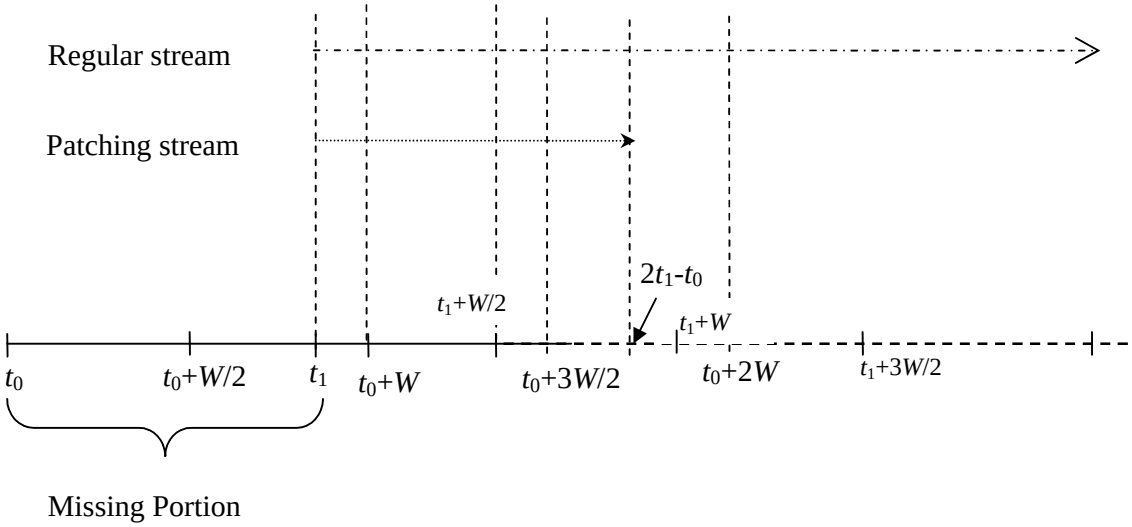


Figure 3.10 Buffer requirement analysis for Case (ii).

Considering Case (ii), if a session starts at time t_0 and a new client requests the same movie at time t_1 and $t_1 - t_0 > W/2$. In this case downloading and storing both patch stream and regular streams will be continued up to $2t_1 - t_0$ from time t_1 where $(2t_1 - t_0) > (t_0 + W)$. Any client can be selected as patching parent with in $t_0 + 2W$ time interval

and after that the initial chunk can be freed one by one to make free space for regular stream.

There may be five states of a client buffer at different intervals at described in Figure 3.10. This can be discussed as follows:

- A client joined with in the time duration ($t_0+\frac{W}{2}$, t_0+W) will store the initial part after $t_0+\frac{W}{2}$. So in the time interval (t_1, $t_1+\frac{W}{2}$) only the storing of regular stream will take place. At the end of this interval the buffer will be at best $\frac{WM}{2}$. This is shown in Equation 3.11.
- After the time $t_1+\frac{W}{2}$ both initial part and the regular stream will be stored in the client buffer and continued till 2 $t_1 - t_0$. The combined situation is discussed in the equation 3.12. The maximum buffer size can be $\frac{3W}{2}M$.
- The missing portion will be made up at time $2t_1 - t_0$ and only the patching stream continues till $t_1+\frac{3W}{2}$. The maximum buffer required for this interval is $2WM$. This is shown in Equation 3.13.
- A client may be selected as a patching parent up to time $t_0 + 2W$. So, initial data needs to be in the buffer is up to this time. If a client is selected as a patch parent at t_0+2W it will take $t_{0+}3W$ time to finish the patch stream. This is shown in Equation 3.14 and the data status can be of maximum size the $2WM$.
- After Time $t_0 + 3W$ 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 3.15.

$$b=(t- t_1)M \qquad t\in [t_1, t_1+W/2] \qquad (3.11)$$

$$b=(t_1+W/2-t_1)M+2(t-t_1-W/2)M \qquad t\in [t_1+W/2, 2t_1- t_0] \qquad (3.12)$$
$$= 3WM/2$$

$$b=(t_1+W/2-t_1)M+2(2t_1- t_0- t_1 -W/2)M$$
$$+(t- 2t_1+ t_0)M \qquad t\in [2t_1- t_{0,}\ t_1+3W/2] \qquad (3.13)$$
$$= WM/2+2(t_1- t_0-W/2)+(t_1+3W/2-2\ t_1+ t_0)M$$

$= WM/2 + WM + (3W/2 - (t_1 - t_0))M$

$= WM/2 + WM + WM/2$

$= 2WM$

$b = (2t_1 - t_0 - t_1)M + (t_1 + 3W/2 - t_1 - W/2)M \quad t \in [t_1 + 3W/2, t_1 + 3W]$ (3.14)

$= (t_1 - t_0)M + WM$

$= 2WM$

$b = (2t_1 - t_0 - t_1)M \quad t \in (t_0 + 3W, L)$ (3.15)

$= WM$

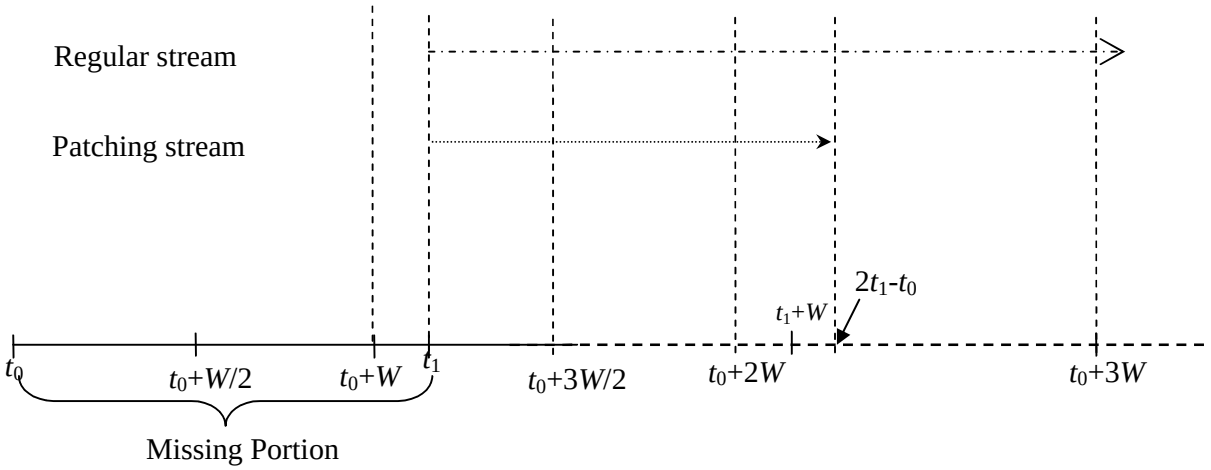


Figure 3.11 Buffer requirement analysis for Case (iii).

Considering Case (iii), if a session starts at time t_0 and a new client requests the same movie at time t_1 and $t_1 - t_0 > W$.

There may be different states of a client buffer at different intervals as described in Figure 3.11. This can be discussed as follows:

- A client joined with in the time $(t_0+W, t_0+2W/3)$ will store the initial part after t_0+W. So in the time interval (t_1, t_1+W) only the storing of regular stream will take place. At the end of this interval the buffer will be at best WM. This is shown in Equation 3.16.
- At the time interval $[t_1+W, 2t_1-t_0]$ both initial stream and regular stream are stored. Here we can see that $t_1+W > t_0+2W$ and our system allows a new client to join only with in t_0+2W time. But a client joined with in t_0+2W can request for second parent till t_0+3W. So the client will store the initial part from t_1+W

to t_1+2W. The situation can be expressed by Equation 3.17. The maximum buffer size can be 2WM.

- The missing portion will be made up at time $2t_1 - t_0$ and only the patching stream continues till t_1+2W. Here $2t_1 - t_0 > 2W$ and a new client are allowed to join an ongoing session till t_0 +2W time. But a client admitted at t_0+2W needs W time for the first patch. So any client can be selected as patch parent till t_0+3W for the second patch stream. This is shown in Equation 3.18. In this interval the maximum buffer size can be 2WM.
- A client selected as a patching parent at time t_0 +3W needs to complete the patch stream till t_0+4W. Then the initial chunk can be freed for regular stream. This situation is shown in Equation 3.19 and the data status can be of maximum size the 2WM.
- For the remaining time interval only the regular stream will be remained at the client buffer which is of 2WM length and described by the equation 3.20. So the upper bound of the client buffer requirement is 2WM in our scheme where the effective window size is 2W.
- If a client comes after the time t_0 +3$W/2$ time it will only store the missing portion and no other buffering is required. The situation can be expressed as in Equation 3.21. Where the maximum buffer requirement can be 2WM.

$$b=(t-t_1)M \qquad t \in [t_1, t_1+W] \qquad (3.16)$$

$$b=(t_1+W-t_1)M+2(t-t_1-W) \qquad t \in [t_1+W, 2t_1-t_0] \qquad (3.17)$$
$$=(t_1+W-t_1)M+2(2t_1-t_0-t_1-W)$$
$$=2WM$$

$$b=(2t_1-t_0-t_1)M+(t-t_1-W)M \qquad t \in [2t_1-t_0, t_0+3W] \qquad (3.18)$$
$$=(t_1-t_0)M+(t_0+3W-t_1-W)M=2WM$$

$$b=(2t_1-t_0-t_1)M+(t_0+3W-t_1-W)M \qquad t \in [t_0+3W, t_0+4W] \qquad (3.19)$$
$$=2WM$$

$$b=(2t_1-t_0-t_1)M \qquad t \in [t_0+4W, L] \qquad (3.20)$$
$$=(3W/2)M$$

$$b=(2t_1-t_0-t_1)M \qquad t \in [t_1, L] \qquad (3.21)$$
$$=2WM$$

3.8 Comparison of Patching Effort

In this research we define a new parameter called patching effort to justify the performance of our system with respect to the Client Assisted Patching. The Patching effort is defined as the duration of patching multiplied by the average buffer requirements during this interval. This parameter expresses the measure of the effort that a client needs to provide as a patch parent during a single multicast session.

First we have determined the time at which a client can act as a patch parent and we measure the minimum buffer required at this time. Then we measure time duration how long a client needs to store the initial portion of the movie to provide as patch and also calculated the maximum buffer required during this interval. Then we get the average buffer requirements from the minimum buffer and maximum buffer. Finally we have calculated the patching effort by multiplying the average buffer requirement with the duration and find the total patching effort. The summary of the measurement of client efforts in different case are given in Table 3.2.

Table 3.2: Calculation of patching effort for both CAP and DCAP.

DCAP (Window size $2W$)							
Case	Min buffer Requirement	Max buffer Requirement	Average buffer	Start as patch parent	Relinquish the initial buffer at	duration	Client efforts
Case (i)	0	$3W/2$	$3W/4$	t_0	t_0+3W	$3W$	$9W^2/4$
Case (ii)	$W/2$	$2W$	$5W/4$	t_0+W	$t_0+7W/2$	$5W/2$	$25W^2/8$
Case (iii)	W	$2W$	$3W/2$	t_0+2W	t_0+4W	$2W$	$6W^2/2$
Total client effort=						$8.375W^2$	
CAP (Window size $2W$)							
Case (i)	0	$2W$	W	t_0	t_0+4W	$4W$	$4W^2$
Case (i)	0	$2W$	W	$t_0+W/2$	t_0+4W	$7W/2$	$7W^2/2$
Case (i)	0	$2W$	W	t_0+W	t_0+4W	$3W$	$3W^2$
Case (i)	0	$2W$	W	$t_0+3W/2$	t_0+4W	$5W/2$	$5W^2/2$
Total client effort=						$13W^2$	

In this analysis we have considered the patch window of same size for both the system to compare the effort. Here we have considered three different cases for Distributed Client Assisted patching and four different cases for Client Assisted Patching. Because in our system the user coming after the time $t_0+3W/2$ need not to store any part of the movie and need not act as a patch parent. But in the case of Client Assisted Patching any client coming within the patch window stores some part of the movie and need to store till double of the window size. Because a client selected at the end of the patch window needs another patch window time to complete the total stream.

From the above analysis we can see that our system requires less patching effort than that of Client Assisted Patching when both of the system has the same window size. That means in our system a client has to give less effort as a patching parent to provide the same patch stream.

Chapter 4

Performance Study

We have described the architecture and algorithm of the admission controller along with its complexity analysis in the previous chapter. In this chapter we describe the simulation results of Distributed Client Assisted Patching in an enterprise network. We study the behaviour of our approach and evaluate its performance based on the simulation results. We also compare the simulation results of the proposed scheme with Client Assisted Patching [4].

4.1 Simulation Technique

We simulate Distributed Client Assisted Patching using Parsec [29], 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 Client Assisted Patching to make a comparison between our approach and Client Assisted Patching on various metrics. In Parsec, we have two kinds 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. For details of the simulation entities and messages the interested readers are referred to [4]. In the following sub sections we describe the various settings of our simulation. Table 4.1 presents the entire simulation settings. Similar parameter settings are considered in simulating Client Assisted Patching scheme.

Table 4.1: Simulation Setting.

Parameter	Value
General Settings	
Number of nodes	20
Number of links	32
Number of clients	500 – 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 120 STU
Patching window	300 – 720 STU
Duration of a movie	3600 STU
Simulation time	86400 STU
*STU means Simulation Time Unit which is equivalent to almost 4.5 µSecond.	

4.1.1 Simulation Parameters

We run our simulation for an enterprise network as a connected graph of switch nodes. Number of clients is varied in the rage of 500–1200. 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 (which are kept as contant) 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 0.2 request/second to 2 request/second.

Table 4.2: Usual Simulation Setting.

Parameter	Value
General Settings	
Number of clients	500
Number of servers	5
Number of replica	1

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

Here 0.2 requests/second indicates a lightly loaded system and 2 requests/second indicates a heavily loaded system.

4.1.2 Simulation Assumptions:

The necessary assumptions for the simulation are listed as follows:

- We considered two different types of requests in the simulation. Firstly the regular requests for movies or videos. Secondly the VCR requests made by the clients of the system.
- The regular movie requests are either batched for the next regular multicast session or it is patched in the current multicast session.
- It is assumed that the VCR requests are generated after a random time which is sometime larger than the patch time of a client when there is a patch request. This is because we consider our scheme as a less interactive system. The number of VCR requests is less than 5% of the regular movie requests.
- Only pause, fast forward and rewind are considered as VCR actions in simulation. The requests are directly served from the multimedia server. No batching or patching scheme is applied for these types of requests.
- We consider the probability of issuing VCR actions during a play out is higher at the later part of a movie than the earlier part of the movie. A similar probability distribution is also considered for the event of leaving a session during a play out.
- Service interruption due to leaving of a patching parent is not considered in this simulation.
- 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 have considered a fixed play back rate in our system.

4.2 Some Probability Distribution Used in the System

We have already mentioned that we have considered the request generation process of a client in our system as a Poisson process. We have also considered that the videos are requested with frequencies following a Zipf-like distribution. In this section we discuss some of the distributions.

Poisson process:

A Poisson process is a stochastic process in which events occur continuously and independently of one another.

The Poisson process is a collection $\{N(t) : t \geq 0\}$ of random variables, where $N(t)$ is the number of events that have occurred up to time t (starting from time 0). The number of events between time a and time b is given as $N(b) - N(a)$ and has a Poisson distribution.

Poisson distribution:

Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event.

If the expected number of occurrences in this interval is λ, then the probability that there are exactly k occurrences (k being a non-negative integer, k = 0, 1, 2, ...) is defined as

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Zipf-Law :

Zipf-Law states that the probability of occurrence of words or other items starts high and tapers off. Thus, a few occur very often while many others occur rarely.

We have used Zipf-like distribution to select the movie with a skew factor z.

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.

4.3 Performance Analysis of Distributed Client-Assisted Patching

In this section we discuss the simulation results obtained for Distributed 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, the number of replica for each content, the patching window size and the total number of clients in the system.

4.3.1 Number of Servers

We have considered that the movies are distributed randomly in multiple servers in an enterprise network. The number of servers in the network has a positive impact on the performance of the system. If the number of servers is increased the movies are further dispersed. 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. But the percentage of served is not doubled when the number of server is doubled. Increase of number of servers causes the increase of server source but the link bandwidths connecting the servers and users are not proportionally increased. This is the main reason behind this observed behaviour of the percentage of served request with the increase in number of servers in the system.

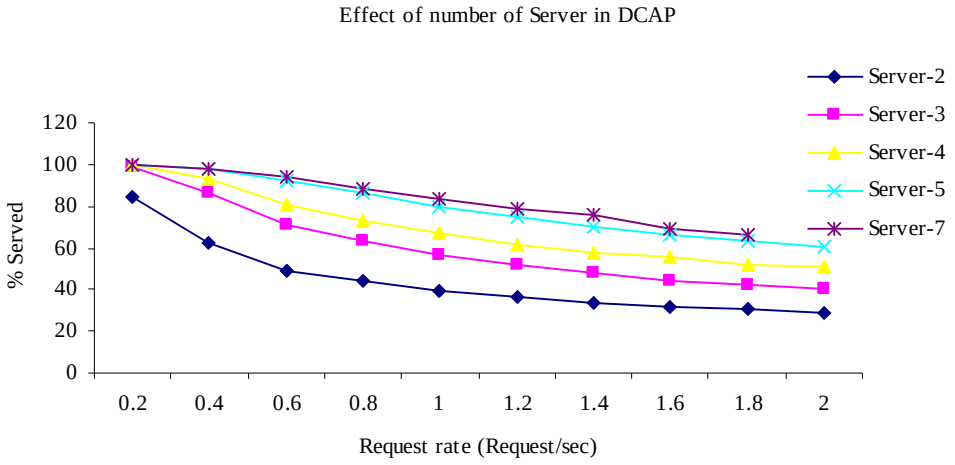


Figure 4.1: Percentage of served requests for different request rates in Distributed Client Assisted Patching with different number of servers. (The numbers in the legend indicate the number of server.)

4.3.2 Number of Replica

We consider that the popular movies are replicated to different servers in our system. The popular movies are those which experience most of the requests. These are also called as first round movies [31]. Replication of popular movies is considered only to satisfy the high demand of such movies. Here replication means to make copies of the same movie in more than one server. No replica means a movie is stored in a single server. Number of replica is always less than the number of server.

We have changed the number of replication to observe the performance of the scheme without changing other parameters. 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 behaviour 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. But the effect of replication on the percentage of served requests is not as significant as that of using more servers. Replication only makes more alternative source but other resources are still limited. It ensures better utilization of the server and link resources but does not increase the resources.

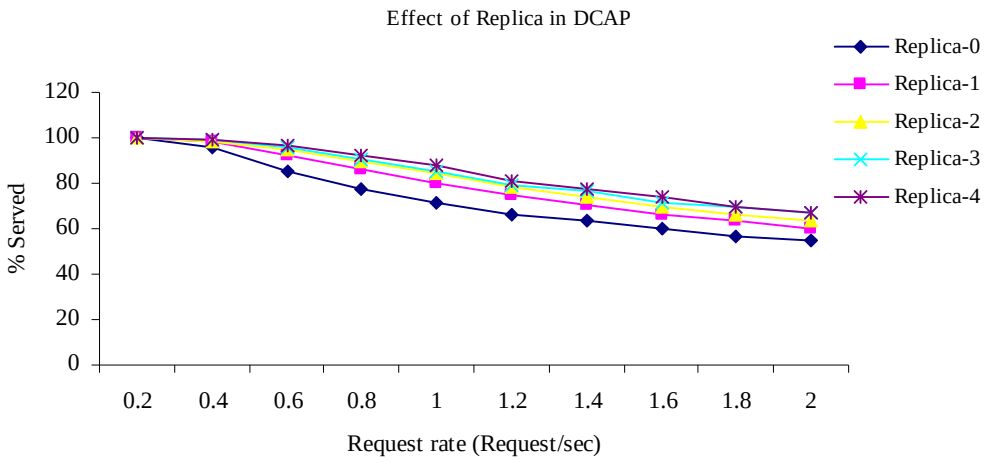


Figure 4.2: Percentage of served requests in Distributed Client Assisted Patching with different request rates and replication.

4.3.3 Patching Window

This section describes 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. The patch clients share a single movie stream among themselves to utilize the server resources. 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 4.3.

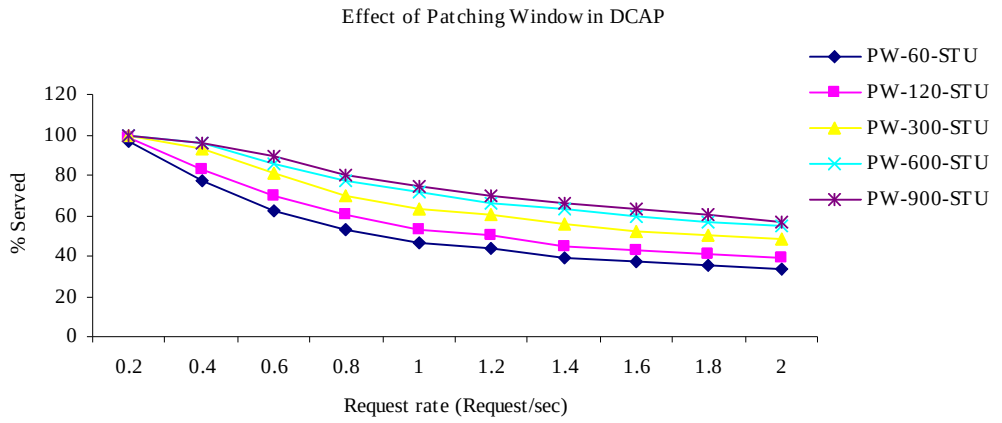


Figure 4.3: Percentage of served requests for different request rates in Distributed Client Assisted Patching with different patching window size. (The number in the legend indicates length of the patching window.)

4.3.4 Average waiting time

Figure 4.4 shows the average waiting time of a client with varying server bandwidth. Here we can see that the waiting time is almost same for different request rate for a fixed server bandwidth. A particular multicast movie stream can serve unlimited number of users if the underlying network has the required capability. Thus more requests in the system will not consume more server bandwidth. That is why the request rate (i.e. the number of users) has very insignificant effect on the waiting time for a particular server bandwidth.

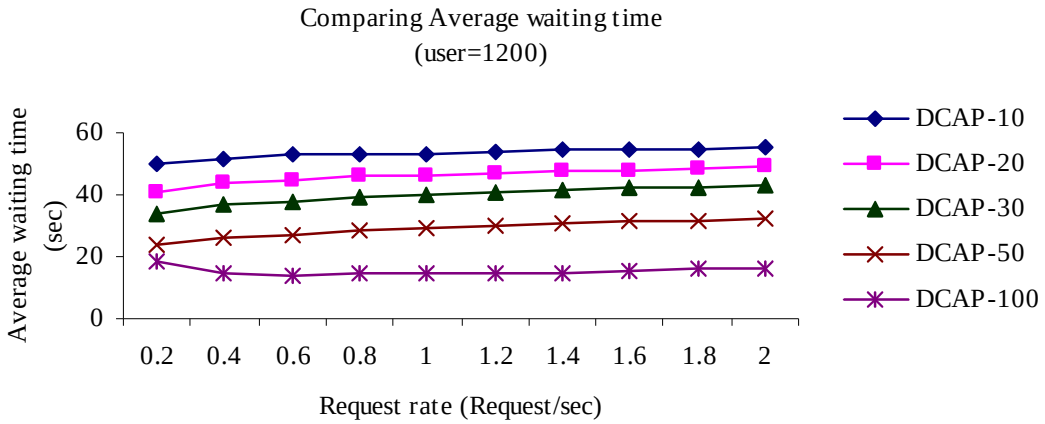


Figure 4.4: Average waiting time with different server bandwidth. (The number in the legend indicates server bandwidth.)

From Figure 4.4 we observe that the waiting time is higher when the available server bandwidth is low. Because increasing the server bandwidth will allow the system to run more patching and more multicast stream as well. Thus the users need not wait for subsequent batches for a particular request and hence reduce the average waiting time. But when the request rate is very low (request rate=0.2) with a high available server bandwidth (BW=100) all the requests seem to be different requests for different movies, so the requests must wait until the new batch starts. Eventually the patching is not effective here. But if the request rate is little bit increased to 0.4 the patching will be in effect thereby decreasing the average waiting time. For higher request rate with this server bandwidth the contribution of congestion occurs rejection of requests resulting the increase of average waiting time. Such behaviour is also observed for higher server bandwidth. Or low bandwidth the rejection for congestion occurs even for low request rate. Thus we do not observe the high average waiting time for low server bandwidth with low request rate.

4.4 Comparison with Client Assisted Patching

In this section, we compare various results of Distributed Client Assisted Patching with the results obtained for Client Assisted Patching scheme. We observe and compare several performance metrics such as bandwidth requirement of servers, the percentage of served requests of the system, the percentage of patched requests of the system and time requirement from the simulation results obtained by varying different parameters. The parameter settings for the simulations are given in Table 4.3.

Table 4.3: Simulation Setting.

Parameter	**Value**
General Settings	
Server Bandwidth	20-100
Link Bandwidth	500-700
Number of clients	500 – 1200

4.4.1 Percentage of Served Requests

In this section we compare the percentage of served requests of the proposed scheme and that of the Client Assisted Patching scheme.

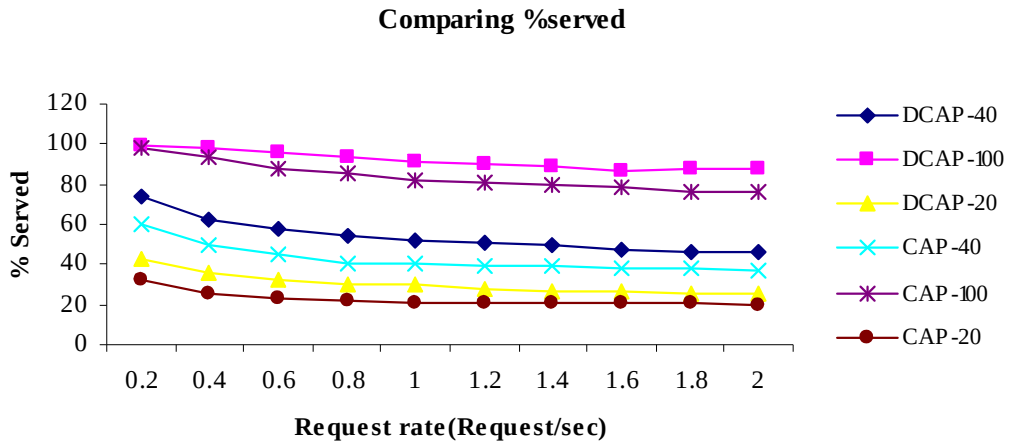


Figure 4.5 Percentage of served requests for different request rates and different server bandwidth. (The number in the legend indicates server bandwidth and the abbreviation DCAP means Distributed Client Assisted Patching and CAP means Client Assisted Patching schemes.)

Figure 4.5 compares the percentage of served requests of Client Assisted Patching and distributed Client Assisted Patching with different server bandwidth. We observe that distributed scheme outperforms the previously proposed Client Assisted Patching scheme as the distributed Client Assisted Patching scheme provides the scope of more patching streams. From the figure we observe that for the server with higher bandwidth and low request rate both the systems get enough bandwidth for regular multicast and patching as there will be less number of users in the system. So percentage served is almost 100% and we get similar performance for both the schemes. But when the server bandwidth is very low with high request rate both the system will perform worse as there will be contention for resources. This results almost the same percentage of served requests in both the systems. Though the distributed Client Assisted Patching allows more patches, this is also limited and insignificant for a specific multicast session.

4.4.2 Percentage of Patched Requests

The number of requests patched is a very important factor in both Client Assisted Patching and our proposed scheme. If a user is patched from the cooperative clients the server bandwidth can be saved. The patching window is doubled in distributed Client Assisted Patching scheme compared to Client Assisted Patching scheme. So it can satisfy a larger user group by providing patches to the clients using only the link

bandwidths. Server bandwidth is only used to initiate the regular multicast and it is possible to save server bandwidth further by delaying the regular multicast.

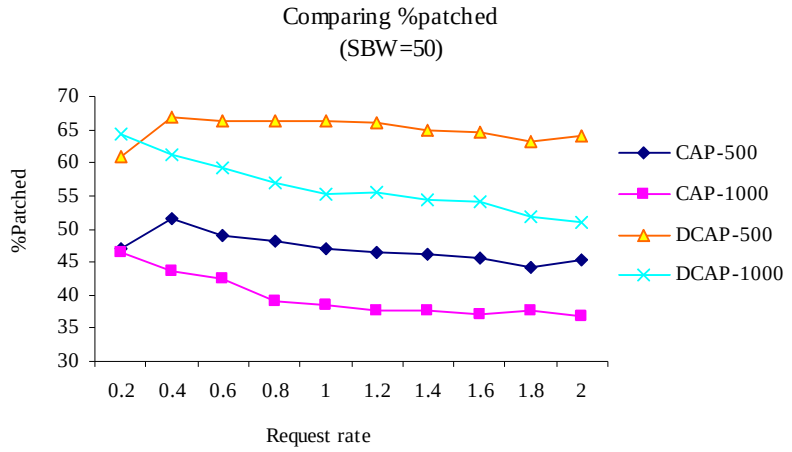


Figure 4.6(a) Showing percentage of patched requests for different request rates with medium server bandwidth and different number of clients. (The number in the legend represents the number of users.)

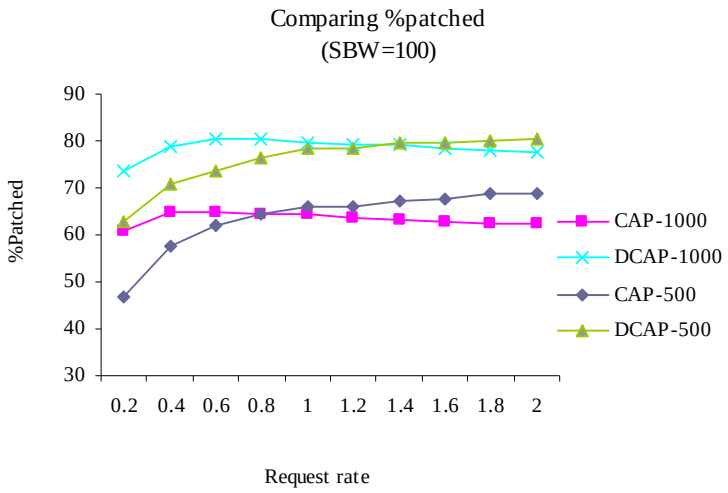


Figure 4.6(b) Showing percentage of patched requests for different request rates with higher server bandwidth and different number of clients. (The number in the legend represents the number of users.)

It is observed from Figure 4.6(a) that percentage of patching degrades with the increase in request rate. For a system with low server bandwidth there will be less number of multicast movie streams and the large number of clients will be competing for patching. This will create contention in link bandwidth and the clients will be rejected for patching when there are not enough resources in links.

We do not observe this degrading behaviour of the percentage of patched clients in Figure 4.6(b) as there are sufficient resources for multicast movie stream and the patched clients will be better served because of low contention. Here we observe that when both the systems have enough server bandwidth the distributed Client Assisted Patching performs 30% better (on average) than the client assisted patching. This is because our system provides a larger patching window. But when the available server bandwidth is lower as shown in the Figure 4.6(a) the performance of the Client Assisted Patching further degrades where as our system shows almost the same performance as before. It is because our system requires less server bandwidth and the number of patch request depends only on the available link bandwidth.

4.4.3 Average waiting time

Figure 4.7 shows the average waiting time of a client with respect to different request rate. We have collected the simulation result with varying different parameters like server bandwidth and number of users. In general we can say that the higher the request rate the more will be the waiting time. This is because when the request rate is higher there may be shortage of resources. So some clients may need to wait for the necessary resources.

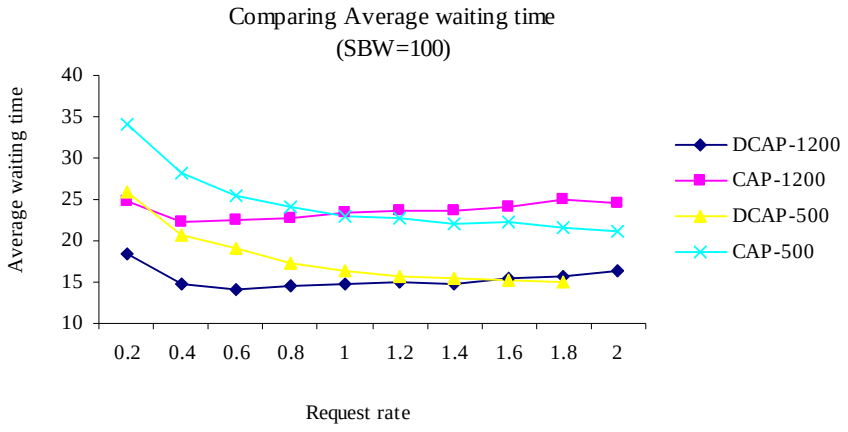


Figure 4.7(a): The average waiting time for a client with different request rate. (The number in the legend represents the number of users.)

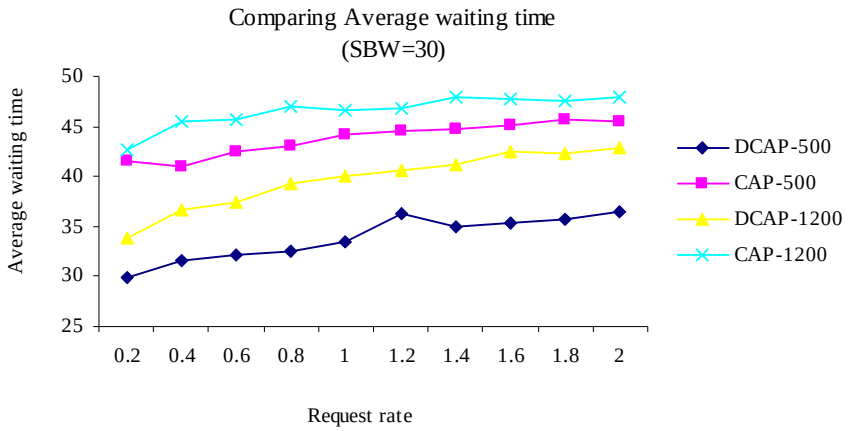


Figure 4.7(b): The average waiting time for a client with different request rate. (The number in the legend represents the number of users.)

When comparing with Client Assisted Patching we observe that our system outperforms with less waiting time as shown in Figure 4.7(a) and 4.7(b). Actually the lack of server bandwidth may not allow frequent batching for a multicast movie stream and clients need to wait for a long time to get available resources. This waiting time for batching plays a significant role in average waiting time. The Distributed Client Assisted Patching overcomes this problem by allowing the broader scope of patching and hence reduces the average waiting time which is observed in the figures. The observed profile of Client Assisted Patching are almost the same as shown in Fig 4.7(a) and 4.7(b). The explanation of this profile is already presented in Section 4.3.4.

4.4.4 Bandwidth Requirement of Server

In multicast VoD system only a single server stream is provided to a group of clients for a regular multicast and patching stream is also provided from the server. But in the proposed system patching stream is provided by the clients and the server only needs to transmit the regular stream. In both Client Assisted Patching scheme and our proposed scheme patching is done by the cooperative clients. But as our system supports a patch window double of the window supported by Client Assisted Patching more clients can be supported by a single multicast session. Also the less bandwidth requirement allows the server to delay batching and incorporate more multicast session.

The following figure compares the bandwidth requirement of the client assisted scheme and distributed client assisted system and also justifies information on favour of the fact we have discussed earlier. Here required server bandwidth is measured by summing up the total bandwidth used in the system during the simulation time.

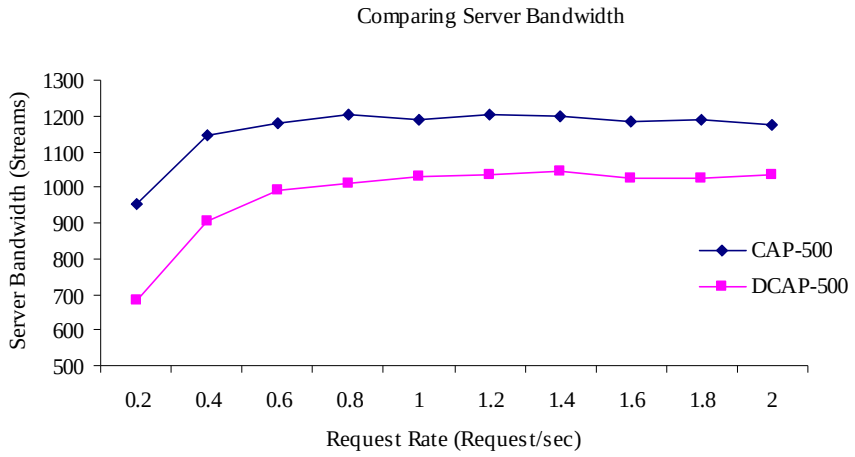


Figure 4.8(a) Server Bandwidth Requirements for different request rates of Client-Assisted and Distributed Client Assisted Patching schemes. (The number in the legend represents the number of users.)

From the figure 4.8(a) we can see that our scheme needs about 45% less bandwidth than that of Client Assisted Patching.

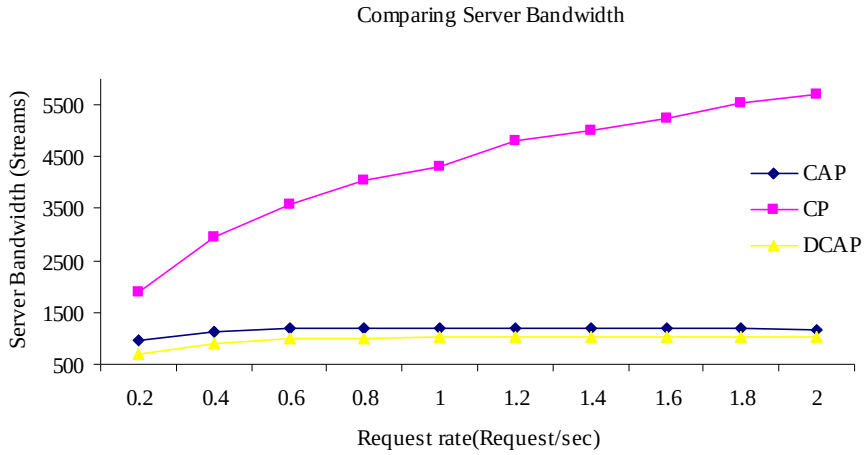


Figure 4.8(b) Server Bandwidth Requirements for different request rates of Client-Assisted, Distributed Client Assisted and Conventional Patching schemes (CP means Conventional Patching.).

Figure 4.8(b) shows the comparison between Client Assisted Patching, Distributed Client Assisted Patching and Conventional Patching scheme. Here we can see that Conventional Patching scheme requires huge bandwidth with compare to other two

schemes because in conventional patching all patch streams are provided from the server.

4.4.5 Execution Time Comparison

We have already discussed the time complexity of our system in Chapter 3. We have shown that the Admission Control procedure of the proposed system needs $O(sE \log V + m^2 + n_c E \log V + 2n_c^2 E)$ 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.

For the same simulation parameters and environment the complexity of Client Assisted Patching is $O(sE \log V + m^2 + n_c E \log V + n^2 c\ E)$. This means that, Client Assisted Patching requires less time for execution. This is because in Client Assisted Patching scheme only single patching parent is required to serve the patch stream and as all clients store the same part of the initial portion only a free patch parent finding is enough. But in our system different groups of clients stores different part of the initial portion. So when selecting patching parent for a client we have to find a parent not serving any user which has the required portion of the patching stream. As all clients do not store the same portion of the initial part it may require long time to find a patch parent. Again when two patch parents are required, we have to perform the search twice. Thus, the proposed system needs more computation to find a patch parent than Client Assisted Patching. This is observed in our simulation results as shown in Figure 4.9. The quadratic pattern of time requirement curve justifies the complexity analysis expression.

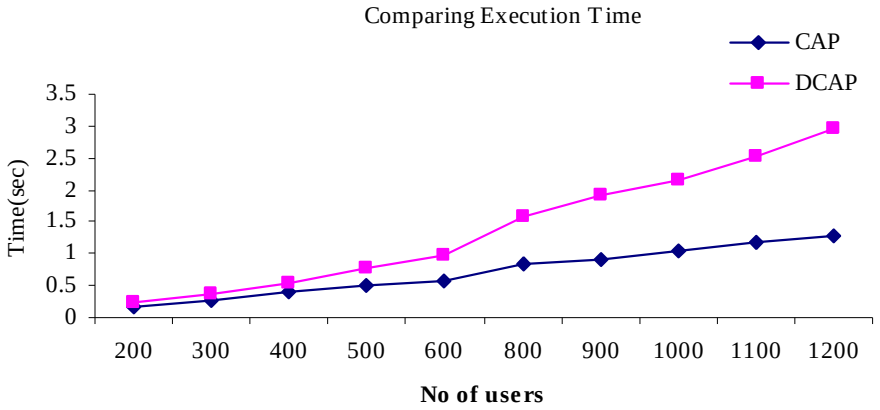


Figure 4.9 Computation time of conventional and Client Assisted Patching schemes with varying number of clients in the system.

4.5 Trade off between Server Bandwidth and Execution Time

From the analysis of server bandwidth requirements described in the Figure 4.8 we can see that our system requires less server bandwidth compared to Client Assisted Patching. But on the other hand the Figure 4.10 shows that our system requires more execution time than that of the Client Assisted Patching. So there may arise some question about how we can make the trade off between the server bandwidth and the execution time. In that case we must say that server bandwidth is the key parameter for any VoD system. And the cost for the server bandwidth is a recurring one. So if we can save the server bandwidth we will require less recurring cost. But when the question is about the performance or execution time we can say that only a few second extra waiting is required in our system which can be negligible. Or even if we are concern about the speed it is possible to set up a high speed system which will cost only for once. So we can say that it is better to save the server bandwidth even if it incurs some extra execution time.

4.6 Comparison with CAP with the Doubled Window size

In this section we compare our system with the Client Assisted Patching when the window size is the same in both of the scheme. To make the window size same we just doubled the patching window of CAP so that it equals to the effective window size of DCAP and tried to compare the performance of the two schemes.

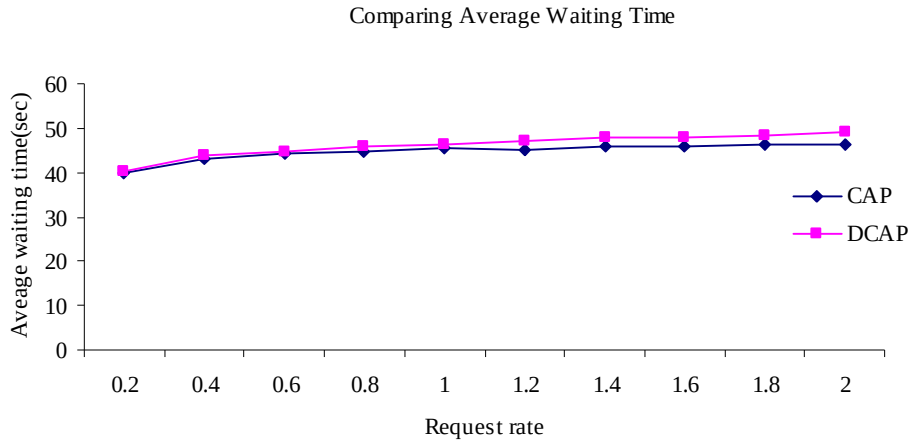


Figure 4.10: Comparing the average waiting time when both CAP and DCAP has a patch window of 1200 STU.

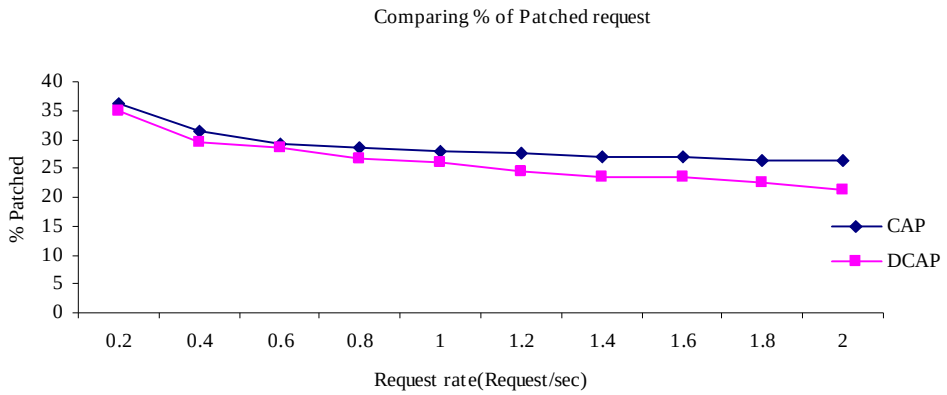


Figure 4.11: Comparing the percentage of patched requests when both CAP and DCAP has a patch window of 1200 STU.

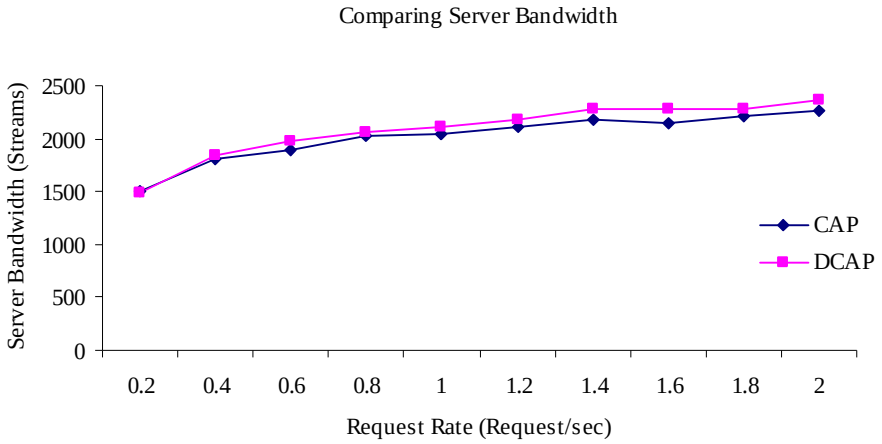


Figure 4.12: Comparing the server bandwidth requirement when both CAP and DCAP has a patch window of 1200 STU.

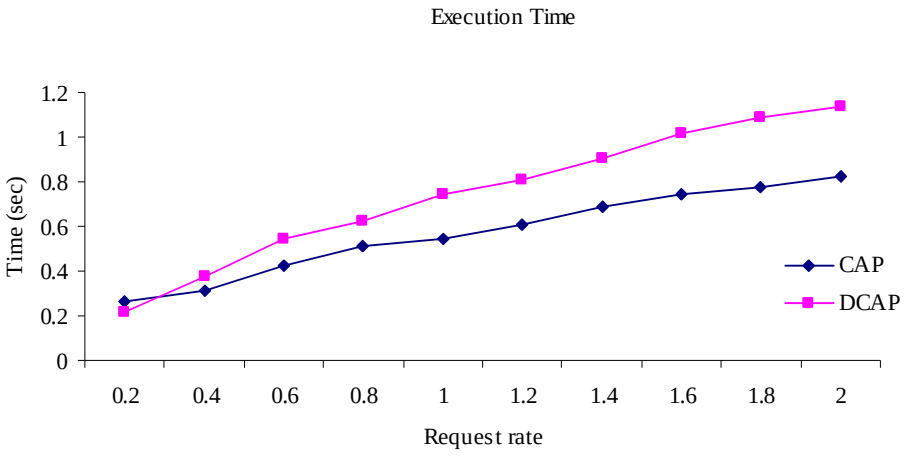


Figure 4.13: Comparing the execution time when both CAP and DCAP has a patch window of 1200 STU.

Figure 4.10 to Figure 4.12 compares the average waiting time, percentage of the patched request and server bandwidth requirement respectively of the two schemes when the patch window is same.

Figure 4.13 compares the execution time of the both scheme when they have same patching window. From the Figure 4.13 we see that our system requires extra time to execute with respect to CAP when the patch window size is same. So from the above observations we can say that both the system has a similar performance when they have a patch window of same size. But in the previous chapter we have observed that our system requires a less client effort to serve this performance. So we can say that our system has a similar performance with a less client effort.

4.7 Observations from the Simulation Results

From the simulation result we have discussed in the previous subsections we get the following observations that Distributed Client Assisted Patching outperforms Client Assisted Patching:

- It significantly alleviates the server load.
- It is more scalable.
- Our system is cheaper to operate.
- It reduces the service latency introduced by batching technique.
- The scheme supports larger multicast group.

Chapter 5

Conclusion

In this concluding chapter, we have summarized the major contributions made by our research work and also focused on some directions for future research over the issue.

5.1 Major Contributions

Since streaming of any multimedia object like high quality video consumes a significantly large amount of network resources, network bandwidth limitation is the major constraint in most of the multimedia systems. Multicast Video-on-Demand (MVoD) systems with Patching are scalable and cheap-to-operate. Under Standard Patching, requests arriving within a patching window are able to share the same multicast using a patching stream. As time elapses, these patching streams need to "patch" more data, and therefore incur a higher communication cost. To optimize system performance, 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 have proposed a new patching technique called Distributed Client Assisted Patching where the initial part of the movie is distributed among groups of client for patching.
- In this system a client can get a larger amount of patch stream combinedly from two different patch parents. Thus our system provides larger patch window.
- As the system is providing a larger patching window a larger multicast group can be supported in a single multicast session. Hence it increases resource sharing and saving server bandwidth.

- In our research we have derived some probabilistic mathematical model to analyse the server bandwidth requirements as well as link bandwidth requirement.
- Distributed Client Assisted Patching and conventional Client Assisted Patching require the same size buffer. But the effective patch window is higher.
- 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 behaviour.
- We also do simulation of our counter scheme, Client-Assisted Patching, to compare our approach with it. Simulation reveals that Distributed Client Assisted Patching outperforms Client Assisted Patching with a very sharp margin in various important aspects like bandwidth requirement of servers.

5.2 Future Directions of Further Research

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 have considered only three groups of clients who will store the initial part. This distribution can be further studied with the analysis of the optimal strategy.
2. In this research we used a specific distribution policy to store the initial part. Different distribution policy can be applied to find the impact.
3. 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.
4. 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.

5. 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.
6. Further study is required in efficient management of VCR action for significantly reducing interactive action blocking rate where patching is in effect. In this research we have totally ignored patching during VCR actions.

Bibliography

[1] Hua, K. A. and D. A. Tran. Range Multicast for Video on Demand. *Multimedia Tools and Applications*, Volume 27, pp. 367–391, May 2005.

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

[3] Cai, Y., W. Tavanapong and K. Hua, "A Double Patching Technique for Efficient Bandwidth Sharing in Video-on-Demand Systems", *Multimedia Application and Tools*, Volume 32, Issue 1, pp.115-136, January 2007.

[4] 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, September1998.

[5] Farhad, S.M., Akbar, M.M. , Kabir, M.H., "Multicast Video-on-Demand Service in an Enterprise Network with Client-Assisted Patching", *Multimedia Tools and Applications*, Volume 43, Issue 1, pp 63 – 90, May 2009.

[6] Ma, H., Shin, K. G., "Multicast video-on-demand services," *ACM Computer Communication Review*, Volume 32, Issue 1, pp. 31-43, ACM Press, 2002.

[7] Emmanuel, L. Profeta A., Shin, K. G., "Providing unrestricted VCR capability in multicast video-on-demand systems,"*Conference on Multimedia Computing and Systems'98*, June-July 1998.

[8] Almeroth, K.C., Ammar, M.H., "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE J. Select. Areas Commune*, Volume 14, Issue 6, pp. 1110–1122, Aug. 1996.

[9] Liao W., Li V.O.K., "The split and merge protocol for interactive video-on-demand," *IEEE Multimedia*, pp. 51–62, Oct.-Dec.1997.

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

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

[12] Jung, J., Lee, D., "Harmonic Broadcasting for Video-on-Demand Service," *IEEE Transactions on Broadcasting*, Volume 43, Issue 3, pp. 268-271, 1997.

[13] Viswanathan, S., Imielinski, T., “Metopolitan Area Video-On-Demand Service Using Pyramid Broadcasting,” *IEEE Multimedia Systems*, Volume 4, pp. 197-208, 1996.

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

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

[16] Carter,,S. W., Long, D. D. E., “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.

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

[18] Sheu, S., Hua, K. A., Tavanapong W., “Chaining a Generalized Batching Technique for Video-on-Demand Systems,” *IEEE ICMCS’97*, pp. 110-117, Ottawa, 1997.

[19] Ma, H., Shin, K. G., “Multicast Video-on-Demand Services”, *ACM Computer Communication Review*, Volume 32, Issu 1, pp. 31-43, 2002.

[20] Cai, Y., Hua, K. A. , Vu, K., “Optimizing Patching Performance,” *Multimedia Computing and Networking, MMCN’99*, January 1999.

[21] Cai Y., Hua, K. A., “An Efficicent Bandwidth-Sharing Technique for True Video on Demand Systems,” *ACM Multimedia’99*, pp. 211-214, Orlando, November 1999.

[22] Ha, S.J., Bae I.H., Kim J.G., Park Y.H. and Oh S.J., “An Expanded Patching Technique using Four Types of Streams for True VoD Services”, *KSII Transactions On Internet And Information Systems*, Volume 3, Issue 5, pp 444-460 October 2009.

[23] Dakshayini[1], M., Nair[2], Dr T R G. K., “Client-To-Client Streaming Scheme For Vod Applications,” *The International Journal of Multimedia & Its Applications (IJMA)*, Volume 2, Issue 2, May 2010.

[24] Guo, Y., Suh, K., · Kurose, J., Towsley, D., “An enhanced client-centric approach for efficient video broadcast,”, *Multimedia Tools and Applications*,Volume 33, pp. 109–129, 2007.

[25] Chen, J. M., Leu, J. S., Chen, Y. C., Wei, H. W., Shih, W. K., "MegaDrop: A Cooperative Video-on-Demand System in a Peer-to-Peer Environment", *Journal Of Information Science And Engineering*, Volume 27, pp. 1345-1361, 2011.

[26] Wang, Y., Zhang, Y., Hu L., Huang, X., "A New Zero-Delay Video-on-Demand Scheme", *Journal of Information & Computational Science* , Volume 7, Issue 10 pp. 2122–2129, 2010, Available at http://www.joics.com.

[27] Kim, H., Yeom, H. Y. ,"P-chaining: a practical VoD service scheme autonomically handling interactive operations", *Multimedia Tools and Applications*, Volume 39, pp. 117–142, 2008.

[28] Kwon, J. B., "Proxy-assisted scalable periodic broadcasting of videos for heterogeneous clients", *Multimedia Tools and Applications*, Volume 51, pp.1105–1125, 2011.

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

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

[31] Islam, M. M., M. M. Akbar, Hossain, H., Manning, E.G., "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.