An Integrated Approach to End-to-End Analysis of Distributed Video-on-Demand Systems

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University.

By

Padmavathi V. Mundur

Bachelor of Engineering, Industrial Engineering
Bangalore University, 1985
Post-Graduate Diploma, Industrial Management
Indian Institute of Science, 1986
Master of Engineering, Systems Engineering
University of Virginia at Charlottesville, 1990

Director: Arun K. Sood
Co-Director: Robert P. Simon
Department of Computer Science

Spring Semester 2000
George Mason University
Fairfax, Virginia

DEDICATION

To my dear dad, Mundur Venkatasubbiah.

ACKNOWLEDGMENTS

I am indebted to my advisor, Arun Sood for his excellent guidance as well as his support and encouragement. He is the quintessential professor and it is my privilege to have worked with him. I would also like to thank my co-advisor, Robert Simon for his accessibility and advice while acting as my sounding board many times. Both have been extremely generous with their time and very patient when I pulled all those difficult paper deadlines. I am fortunate to have had an excellent committee consisting of Daniel Menascé, Edgar Sibley and Jim Chen. I am grateful for their insightful comments and for their valuable time.

Thanks are due to the wonderful staff at CS and IT&E departments who always helped me with a smile. I thank my many friends at GMU for making this long journey tolerable and even enjoyable. In particular, thanks are due to Mitch Potter for his help in LaTeX early on. Special thanks go to Liz White for her friendship and career advice. I would also like to thank a very special friend, Priscilla McAndrews for being there, appreciatively awaiting my modest triumphs.

I would like to acknowledge my mother, M. V. Bhagirathi for her constant support. Finally, I am most grateful to my best friend and husband, Manohar Malayanur for supporting and encouraging me all along.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

AN INTEGRATED APPROACH TO END-TO-END ANALYSIS OF DISTRIBUTED VIDEO-ON-DEMAND SYSTEMS

Padmavathi V. Mundur, Ph.D.

George Mason University, 2000

Dissertation Director: Dr. Arun K. Sood, Co-Director: Dr. Robert P. Simon

Distributed Video-on-Demand (VoD) systems are expected to be one of the most important services supported by the next generation of high-speed networks, video servers, and distributed multimedia file systems. Using a distributed VoD system, a client can request a video from anywhere and at any time. In response to a client's request, VoD systems deliver high quality digitized video directly to client set-top boxes. A typical VoD architecture consists of three subsystems: storage, network, and client. Admission control tests are employed on all subsystems before admitting a new request to check whether resources are available to provide guaranteed service. This architectural approach represents a dramatic break from a best-effort environment. From a research point of view, VoD systems pose great challenges in the areas of retrieval and delivery for real-time performance. Much of previous research restricts the analysis of VoD systems to either the storage or the network subsystem.

The focus of my dissertation research is the end-to-end analysis of a distributed VoD system. I analyze the distributed architecture of the VoD system to design global request handling and admission control strategies and evaluate them using

global metrics. I also analyze and model all subsystems — server, network, and client — involved in providing guaranteed services to an individual request in an integrated framework. The objective of this analysis is to develop a methodology for determining efficient ways of using all resources in the VoD architecture within the constraints of providing guaranteed high quality services to each request.

In this dissertation, I develop an analytical model that integrates the video server retrieval techniques with the network transfer mechanisms by relating buffer management at the server with rate-based scheduling at the network. The significance of this integrated analysis is the development of a method for designing and measuring the effectiveness of end-to-end admission control and request handling policies. I extend the analysis to include the effects of controlling the network delay jitter, where jitter is the difference between maximum and minimum end-to-end delay experienced by packet transfers. I prove that the client buffer requirements will be reduced if the underlying network service discipline provides a non-trivial delay jitter bound.

I propose a new technique called threshold-based admission control for VoD systems and develop an analytical model for computing blocking probabilities efficiently. Using simulation, I show that threshold-based VoD system behavior conforms to the analytical model. Through analytical and simulation results, I show that threshold-based admission control can be used to manipulate VoD performance by restricting access to lower priority request classes.

I analyze the integrated admission control strategy through an extensive simulation of a distributed VoD system consisting of server clusters and networks. My results show that request handling policies based on limited redirection of blocked requests to other resources perform better than load sharing policies. This shows that limited redirection must be incorporated in the reservation or signaling protocols. For providing service at different quality of service levels, I show that the downgrade option coupled with threshold restrictions is a powerful tool for transforming an incoming request mix into a workload that the VoD system can handle.

CHAPTER 1

INTRODUCTION

Evolution of standards and protocols for high speed, integrated services networks, coupled with rapid advances in the storage and compression technology, has spurred interest in distributed multimedia systems. Of the many applications for distributed multimedia systems, video-on-demand (VoD) has much appeal because of its on-demand nature. Distributed VoD systems are expected to be one of the most important services supported by the next generation of high-speed networks, video servers, and distributed multimedia file systems. Using a distributed VoD system, a client will be able to request a video from anywhere and at any time. In response to a client's request, VoD systems will deliver high quality digitized video directly to client set-top boxes or workstations.

From a research point of view, distributed VoD systems pose great challenges in the areas of retrieval and delivery for real-time performance. Typically a distributed VoD system consists of hierarchies of servers and network elements connected to a large client base. The servers transmit the multimedia data periodically over the network in response to a client's request. Because there can be no gaps in the presentation of the multimedia data on the client site, there must be a tight coordination among all components of a VoD system. Guarantees on server retrieval performance and network transfer conditions must be provided to maintain the stringent real-time performance and synchronization requirements of video applications. Current data management practices both at the server and the network levels must be modified to address real-time requirements of multimedia data.

At present, networks and service level technologies provide only best-effort service to conventional data transfer applications. In contrast, VoD applications require guaranteed throughput and bounded delays over a prolonged period of time. To provide these guarantees, it is essential to limit the number and nature of requests being served simultaneously in the VoD system. For each request, we also need reservation processes to allocate resources along its routing path and maintain them for the length of time the request is being served. All these requirements are different from the best-effort principle of present day servers and networks.

Media blocks in a VoD application need to be retrieved at the server and transmitted over the network to the client within strict timing constraints. Unlike traditional data blocks, media blocks are not independent entities but are synchronized with other media blocks and served as a *stream* of blocks. This is also the situation in the network, where data is viewed as a stream of packets, sometimes termed a *flow*. Setting up a flow identifies the set of packets that must receive special guarantees. These continuity requirements over a period of time lead to complex resource management requirements at the heart of which is the concept of admission control. End-to-end admission control strategies determine whether adequate resources are available along the path of the new request so that it can be admitted without violating the continuity requirements of the requests already in service.

1.1 End-to-End Analysis and Admission Control

Tight coordination among subsystems and end-to-end admission control are key to providing real-time performance in VoD systems. If we ignore the coordination and admission control requirements and operate a VoD system on best-effort delivery, much like the current practices on servers and networks, every client is admitted into the system and gets a fair share of the system even under heavy load. This indiscriminate, though fair, distribution of limited resources leads to degraded quality in service under high load. In this scenario, analysis of isolated subsystems is sufficient because we would be primarily interested in learning how a certain subsystem performs under various workload conditions.

For VoD clients, however, the resulting degraded quality in service under high load conditions in a best-effort delivery will not be acceptable, because degraded service means dropped packets resulting in visual artifacts or gaps in display. VoD clients need firm guarantees on throughput and bounded delays. Because of this requirement, if there are not enough resources to provide these guarantees, the client is *denied* service through admission control. In this mode of operation, the objective is to maximize the number of clients being served in the VoD system or minimize the number of requests getting blocked or rejected. This requires careful examination of resource usage on all subsystems taken together. Therefore, there is a strong need for end-to-end analysis and a global strategy for request handling and admission control in a VoD system.

1.2 Problem Statement

The issue addressed in this dissertation is the development of a method for analyzing a distributed VoD system architecture in terms of end-to-end admission control techniques, request handling strategies, and blocking performance. We must analyze the performance of a distributed VoD architecture by designing global request handling and admission control strategies and evaluating them using global metrics through analytical modeling or simulation means. We must also analyze and model all subsystems — server, network, and client — involved in providing guaranteed services to an individual request in an integrated framework. The objective of this analysis is therefore, the development of a methodology for determining efficient ways of using all resources in the VoD architecture within the constraints of providing guaranteed high quality services to each request.

1.3 Previous Work

Previous research in VoD systems has focused predominantly on topics related to video server design, such as disk striping, video block placement, admission control at the level of disks, and disk groups. Much other research has focused on load balancing using replication and placement, either in the context of disk groups or server hierarchies. Other topics, such as fault tolerance and data sharing schemes resulting in *near VoD* have also been explored. On the network side, analyzing network conditions for guaranteed service has received much attention. Several network scheduling disciplines have been proposed. However, the research on the server and the network side has gone on independently, without considering the end-to-end nature of a VoD system. Most of the analytical or simulation studies in VoD systems make simplifying assumptions about the network and do not necessarily model the real-time aspect of request processing on the network.

While there is merit in discussing issues related to individual subsystems, I argue that it is not sensible to neglect the end-to-end characteristic of a complex VoD architecture. This dissertation is different from other related VoD research in that I provide an integrated analysis of all VoD subsystems. I implement that integrated model in an extensive simulation to design and evaluate global request handling policies and admission control techniques.

1.4 Approach

The intent of this dissertation is to provide a method to analyze a distributed VoD system by considering all subsystems in an integrated framework. If desired quality of service cannot be guaranteed, the request will be denied service. In choosing this approach I have kept in mind the commercial mass market aspect of the VoD service, where only the highest quality will be accepted; therefore, the requirements on the system are stringent and do not allow any inconsistencies in the guaranteed quality of service for the duration of a request.

The nature of service considered in this dissertation is a *true* video-on-demand. Though I do not model interactivity in this research, the analysis can be extended to include VCR-like functions. Others have proposed several data sharing schemes, such as batching, stream merging, and multicasting in an effort to maximize the number of requests per video stream. Many of these techniques have analytical appeal but they do not adequately address questions on incorporating interactivity. For that reason, I only consider a *true-VoD* system where each new request if admitted into the system, will be assigned dedicated resources on all subsystems along the request path. Analyzing such an ideal situation gives us a basic understanding of the VoD performance. Any improvements from data sharing techniques can be compared to this basic performance.

Rapid advances on the network side to provide guaranteed services allow us to move away from the best-effort paradigm. It is possible to think of a distributed architecture for a VoD system consisting of clusters of servers and network elements providing true-VoD service. In the context of integrated services, packet switching network, service classes offering enhanced quality of service with regard to bandwidth, queueing delays, and packet loss are being proposed. In response to increasing demand for integrated services on the Internet, the Internet Engineering Task Force (IETF) has set up a working group (intserv) to evaluate proposals on providing guaranteed service (Braden, Clark, and Shenker 1994). Network scheduling schemes such as Weighted Fair Queueing (WFQ) (Parekh and Gallager 1993) which provides firm delays on a per packet basis are already supported by commercial routers. Reservation protocols such as RSVP (resource reservation protocol) (Wroclawski 1997) are being actively researched and evaluated.

The proposed VoD modeling and simulation are general-purpose and include any scheduling schemes at the server and the network as long as the guarantees essential for multimedia transfer are provided. Therefore, the VoD model presented in this dissertation can be used over a variety of underlying protocols and architectures.

1.5 Proposed VoD System Model and Simulation

The proposed VoD system consists of clusters of servers and high speed, integrated services, packet switching networks (see Figure 1.1). A hierarchy is assumed among the clusters of servers: local and remote. Clusters of servers have limited bandwidth and therefore reject requests when there are not enough resources to serve them. Each local cluster is dedicated to a single user population, referred to as the *reference* user population. The local cluster may store a complete or partial set of videos from the video collection. The service from the local cluster is provided over a local distribution network to the reference user population. The local distribution network is assumed to have enough resources to sustain the demand from the reference user population. This is a reasonable assumption, because the local cluster acts as a neighborhood server and as the overall VoD user population grows, more new local clusters will be used. When a new neighborhood local cluster is added to the distributed VoD system, the network capacities can be sized to match the reference user population for that neighborhood. The servers in the cluster however, have limited bandwidth and store a limited set of videos from the video collection. Therefore, the local cluster administers admission control tests before accepting new requests.

Several remote clusters also provide service over high speed networks to many user populations. These remote clusters act as permanent video collection repositories and provide real-time service to client requests. The high speed network does not have unlimited bandwidth and can only serve a limited set of client requests simultaneously. Implementation of reservation protocols and scheduling schemes necessary to provide guaranteed services along the routing path of the requests is assumed. However, the proposed model is independent of any specific protocol or the scheduling schemes. The reference user population generates video requests that are served from the local cluster or the remote clusters based on video availability and policy restrictions.

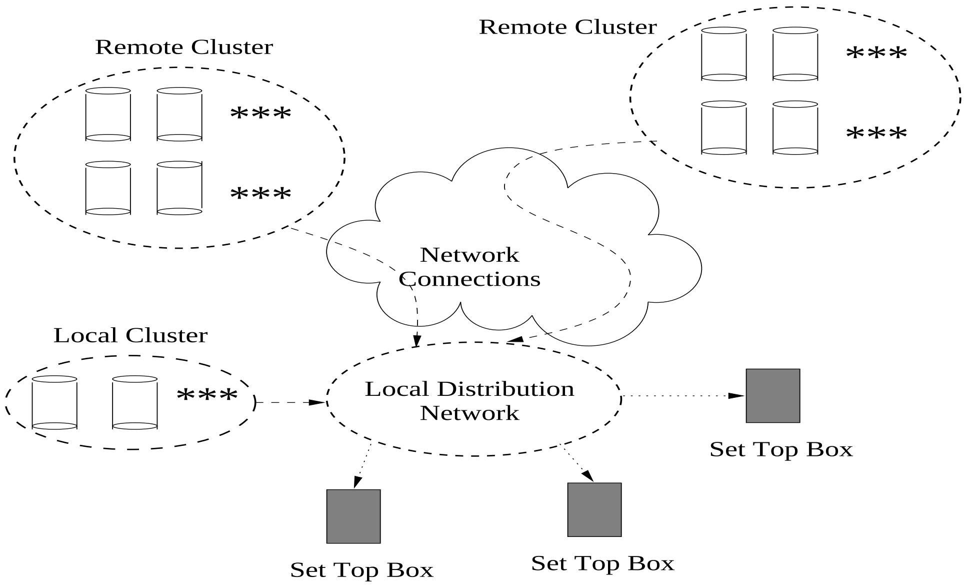

Figure 1.1: Hierarchical VoD architecture

I first develop a remote service model which integrates video server retrieval and network transfer mechanisms by relating the buffer management at the server with rate-based scheduling at the network. The significance of this integrated analysis is the development of a method for designing and measuring the effectiveness of end-to-end admission control and request handling policies. The server model includes a double buffer scheme and the network model assumes a scheduling discipline that provides guaranteed rates and bounded delays. Admission control conditions for preventing starvation of the display process at the client in the absence of network delay jitter are derived. Network delay jitter is defined as the difference between maximum and minimum end-to-end delay in any two packet transfers. The absence of such jitter indicates that the multimedia data will arrive at the client after a constant delay to be consumed by the playback process.

An important aspect of the proposed unified server and network model is that it enables us to study the impact of a server transmission schedule and a guaranteed services network on a client. I provide an analysis that quantifies the client buffer requirements needed to prevent buffer overflows in presence of network delay jitter. In the analysis, I prove that the client buffer requirements are linear in delay jitter. The proposed analysis is a formulation of that well known linear relationship by providing an explicit model of the server retrieval and network transmission schedule. The analysis also shows that controlling delay jitter within the network will enable us to serve heterogeneous clients with different sized buffers.

Admission control as discussed in multimedia research is mostly based on capacity; all requests are admitted into the system if there is available capacity. In this dissertation, I propose a new admission control technique called threshold-based admission control. Proposed admission control technique groups new requests into priority classes; priorities may be based on popularity of videos. For instance, requests for less popular videos are given lower priority. Threshold restrictions are enforced on lower priority classes to make more capacity available for higher priority classes. A threshold is based on the portion of the server capacity allowed for a request class. I develop an analytical model of threshold-based admission control in the VoD system where requests of certain classes are excluded for admission into the system after a threshold. I develop efficient computational techniques to analyze VoD system performance under threshold-based admission control. Threshold-based admission control is shown to be an excellent tool for manipulating the VoD performance to suit the overall objectives of the VoD system.

The integrated framework simplifies the end-to-end admission control process by considering all subsystems together. I construct a simulation study around this integrated framework. The simulation of the distributed VoD system enables us to examine the performance of different admission control techniques and request handling policies. In the simulation, I include features, such as blocking and not queueing, redirection or alternate routing, and downgraded service for blocked requests, all of which are difficult to incorporate in an analytical model. I implement the integrated admission control model at the server clusters and the network subsystems in the simulation. I develop global performance metrics for evaluating the performance of the

VoD system. Request handling policies and admission control techniques are designed and evaluated in their efficient use of the multiple resources in the VoD architecture.

The methodology used is a combination of analytical and simulation modeling. The simulation model is based on discrete-event request processing in a VoD architecture consisting of several server and network resources. Since a request that fails in an admission control test is blocked from service, the performance metrics are based on blocking events. I use traditional measures, such as blocking probability or blocking rate. In addition, I use class-based blocking measures to signify the performance of individual classes of requests and blocking at resources to measure the level of activity at individual resources. Where possible the results of the simulation model and the methodology are validated using analytical results. A Poisson process is used to generate request arrival rate in the absence of a realistic workload. The high level simulation model can easily accommodate actual workload generation as and when it becomes available. A 95% confidence interval analysis is provided to show simulation accuracy.

1.6 Organization

This dissertation is organized as follows. In Chapter 2, an extensive survey of previous and current research into distributed VoD systems is presented. In Chapter 3, I present the integrated admission control model for the distributed VoD system. In Chapter 4, threshold-based admission control techniques for a multi-class VoD server are discussed. Also presented in that chapter is an efficient computational technique to compute blocking probability under threshold-based admission control. An extensive discrete-event simulation of a distributed VoD architecture is presented in Chapter 5. With an analysis of client display requirements in Chapter 6, we come full circle in this end-to-end analysis of the distributed VoD system. I conclude this dissertation in Chapter 7 with some thoughts on future research in this area.

1.7 Contributions

The major contributions of this dissertation are:

- Development of a unified model of a distributed, hierarchical VoD architecture consisting of clusters of video servers and network elements. Using the model, I derive admission control conditions both at the server and the network. I also derive bounds on reserved rates at the network.

- Development of an analytical model to evaluate the performance of a VoD resource such as the video server under threshold-based admission control. In addition, I derive computationally efficient ways of finding blocking probabilities under threshold-based admission control.

- Development of a client model to analyze the impact of the server transmission schedule and the network guaranteed services on buffer overflow and underflow conditions at the client. I analyze the effect of traffic reshaping under different network service disciplines on the client buffer requirements and prove that providing delay jitter control results in reduced buffer requirements.

- Development of an extensive simulation of a distributed VoD architecture by incorporating the scheduling schemes at the server and the real-time network from the proposed unified model. I develop global performance metrics and design and evaluate several request handling policies. I evaluate VoD performance with two service options: all requests at the same Quality of Service (QoS) level and different classes of requests at multi-rate QoS levels.

- Simulation results show that request handling policies based on limited redirection for blocked requests have better performance than policies based on load sharing. I discuss the relevance of different types of request handling policies under various design considerations such as replicated or distributed video collection, non-blocking or shared resources.

- Analytical and simulation results show that threshold-based admission control is a powerful tool for manipulating the VoD system performance. I show that the VoD system behavior under threshold-based admission control in the simulation conforms to the analytical model.

- Simulation results show that threshold-based policies coupled with a downgrade option for serving requests prove extremely effective resulting in good performance for multi-rate QoS option. I show that the downgrade option combined with threshold restrictions is a powerful tool for transforming an incoming request mix over which we have no control into a workload that the VoD system can handle.

The research forms an excellent basis for cost-benefit analyses because of the system-wide approach. The research also, confirms the need for including desirable features, such as multiple trials for blocked requests using redirection to different resources in reservation or signaling protocols for distributed multimedia systems.

CHAPTER 2

LITERATURE SURVEY

In this chapter, I will highlight the issues related to the main components of a VoD system and discuss pertinent literature on each issue. I will evaluate previous and current research and show how this dissertation will address some of the limitations of previous work.

This chapter is organized as follows. In Section 2.1, I briefly discuss distributed multimedia systems. In Section 2.2, research issues related to the storage subsystem such as disk scheduling, data placement and retrieval, admission control, scalability, and fault tolerance are discussed. In Sections 2.3 and 2.4, I present previous research in network and client subsystems. Research in the area of compression techniques is discussed in Section 2.5. Analytical and simulation models as presented in other researches are discussed in Section 2.6. In Sections 2.7 and 2.8, I provide a summary and an evaluation of previous work.

2.1 Distributed Multimedia Systems

Li and Liao (Li and Liao 1997) define a Distributed Multimedia System (DMS) as an integrated communication, computing, and information system that enables the processing, management, delivery, and presentation of synchronized multimedia information with quality of service guarantees. A typical DMS architecture that can provide on-demand, interactive video applications consists of a hierarchical configuration of video servers and network switches. Furht et al. (Furht, Kalra, Rodriguez, and Wall 1995) discuss such a general architecture and provide insight into using ATM-based networks and access technologies such as HFC and HDSL to enable interactive services for thousands of users. The recent trend in multimedia networking however, is towards packet switched, high speed, integrated services network. The list of services that may be offered using a large DMS includes video-on-demand, digital audio, digital libraries, home shopping, and much more. Of the many services, interactive video-on-demand has received the attention of many researchers because of the challenges of providing such a service.

The research into video-on-demand began in the late 1980s with Sincoskie (Sincoskie 1990) examining the technological feasibility of a video-on-demand service. With the technology trends of that time, he concluded that video-on-demand should be a viable application in about ten years. Some of that promise has been realized with the enormous gains in storage technology and high speed network infrastructure. Finding a cost effective means of providing VoD service however, remains an open problem.

A typical VoD system consists of three main subsystems: storage, network, and client subsystems (Little and Venkatesh 1994; Ramanathan and Rangan 1994; Furht, Kalra, Rodriguez, and Wall 1995). In the following sections, I will discuss critical design issues related to these subsystems.

2.2 Storage Subsystem

Continuous media which include audio and video have temporal constraints such as playback deadlines which are not present in textual data retrieval and transfer. Several characteristics distinguish continuous media and merit consideration while designing video servers: real-time storage and retrieval for low-latency real-time performance, large storage space, and high transfer rate. There are two ways to achieve a cost effective solution to the large storage and high transfer rate requirements of VoD systems; first, to maximize the utility of a given storage device by maximizing the number of simultaneous requests that can be handled by video servers and second, to find architectural solutions such as hierarchical configurations of secondary and tertiary devices that will distribute the load within and across different levels. However, the emphasis on tertiary storage devices is diminishing with the recent advances in storage technology. Tertiary storage is still useful as an archival device in building permanent repositories. Hierarchical configuration of servers is still a valid cost-effective solution to provide access to thousands of VoD users.

Prior work in storage subsystems has predominantly focused on finding efficient storage and retrieval techniques that will minimize the overhead of retrieval of multimedia data and maximize the number of concurrent streams. Issues in multimedia server design identified by some of the researches are storage and retrieval of multi-resolution data to serve clients with different computing and communications capabilities, multimedia file systems for efficient continuous-time data access, scalability, and fault tolerance (Jadav and Choudhary 1995; Krikelis 1998; Ozden, Rastogi, and Silberschatz 1995; Shenoy, Goyal, and Vin 1995). I will discuss prior research with respect to these and other issues next.

2.2.1 Storage and Retrieval

Disk transfer rates significantly higher than a single stream real-time playback rate make it possible to serve multiple requests simultaneously; for instance, video servers with more than 100 MBps transfer rates are available and an MPEG II stream requires only about 4-16 Mbps. While serving multiple streams, the server must meet the continuous playback requirements of all streams by scheduling disk requests so that no individual stream is starved. By limiting the number of simultaneous requests, it is possible to provide such guarantees. Disk scheduling, video data placement, and admission control are related issues in determining server real-time performance, (Gemmell, Vin, Kandlur, Rangan, and Rowe 1995; Kunii, Shinagawa, Paul, Khan, and Khokhar 1995; Mourad 1996). These issues are discussed next.

Disk Scheduling

Conventional disk scheduling algorithms such as SCAN where the disk head moves in one direction retrieving data for requests in the order they are stored on the disk or Shortest Seek Time First (SSTF) where requests are serviced in the order of their seek times (Denning 1967) do not have a notion of time or deadlines and are therefore, unsuitable for real-time scheduling. These techniques however, utilize disk arm efficiently and therefore optimize seek times. The best known algorithm for real-time scheduling of tasks with deadlines is Earliest Deadline First (EDF) (Liu and Layland 1973). Scheduling based solely on EDF results in excessive seek time and rotational latency and poor server utilization.

Current research in multimedia retrieval use the notion of a *cycle* or a *round* to refer a time period in which the video server retrieves media blocks of arbitrary length for the set of streams currently being serviced. Processing requests in rounds exploits the periodic nature of video playback. Within each round, a disk scheduling algorithm is employed. Data placement algorithms which by design optimize seek times and rotational latencies are preferred.

Reddy and Wyllie (Reddy and Wyllie 1993; Reddy and Wyllie 1994) propose a scheduling scheme which is a combination of SCAN and EDF algorithms. SCAN-EDF uses SCAN algorithm to serve requests which have the same deadline. Its efficiency depends on the number of requests having the same deadline. Deferred deadlines are adopted to group requests to have the same deadline. This results in increased seek optimizations and hence, accommodate more number of streams. This is achieved at the cost of increased buffer requirements and latencies. Yu et al. (Yu, Chen, and Kandlur 1992; Yu, Chen, and Kandlur 1993) present a disk scheduling scheme called Grouped Sweeping Scheme (GSS) which is a hybrid between round-robin and SCAN scheduling. Using GSS the number of streams being serviced is first divided into g groups such that the streams within a group are serviced using SCAN algorithm, but the groups are served in a fixed order. The efficiency derived from GSS is that a stream's playout begins at the end of its group and not necessarily at the end of a cycle which results in reduced latencies and buffer space. In (Chen, Kandlur, and Yu 1993) and (Yu, Chen, and Kandlur 1993), GSS is optimized to allow for heterogeneous multimedia streams such that for each type of stream the playback time is greater than or equal to the cycle time. In (Anderson, Osawa, and Govindan 1992), a work-ahead scheduling scheme based on least slack time first is presented with worst-case service time assumptions. In (Kenchammana-Hosekote and Srivastava 1994), the round-robin algorithm is adopted as part of their cyclical scheduling policy to provide deterministic service guarantees. Data blocks fetched from a circular buffer for each stream in round k are consumed by the client during round $k + 1$. Kandlur et al. (Kandlur, Chen, and Shae 1994) use SCAN disk scheduling with a double buffer scheme for each stream being serviced in a round. They compare SCAN and fixed order scheduling using contiguous and constrained block allocation. In (Lee and Du 1997) a comparison of basic disk scheduling schemes such as SCAN, CSCAN, and EDF are presented. Jitter related behavior is analyzed under each scheduling scheme for MPEG video accesses.

More recently, Shenoy and Vin (Shenoy and Vin 1998) have proposed a disk scheduling framework called *cello* that is suited for servicing application classes with diverse requirements. They present a two-level scheduling architecture, consisting of a class-independent scheduler and a set of class-specific schedulers, each differing in the way they service requests. Similar to that work, Wijayaratne and Reddy (Wijayaratne and Reddy 1999) present an integrated scheme for providing different levels of performance guarantees to different classes of requests in a single I/O system. In (Rompogiannakis, Nerjes, Muth, Paterakis, Triantafillou, and Weikum 1998), disk scheduling algorithms for mixed workloads of periodic and aperiodic requests in a multimedia server are presented. Their focus is to reduce response time and starvation of aperiodic requests by considering a fixed workload from periodic requests. Vin et al. (Vin, Goyal, and Goyal 1995) propose a greedy disk scheduling algorithm that derives access to a sequence of media blocks by minimizing both seek and rotational latency. In contrast, most of the previous scheduling algorithms optimize seek times. They show an 80% improvement over SCAN scheduling through simulation by minimizing rotational latencies in their algorithms.

Data Placement and Retrieval

Video data placement needs to be considered on three levels: single disks, disk arrays, and server hierarchies. The focus of video data placement on a single disk is to minimize seek times and rotational latencies while retrieving that data. The placement of data on multiple disks or disk arrays introduce other issues such as load balancing and availability. Server hierarchies pose the problem of allocating video files to different levels of that hierarchy so that communication and storage costs may be minimized given a skewed demand access to videos. I discuss placement issues in the context of single disks or disk arrays in the following paragraphs and the topic of hierarchical server configurations will be deferred to Section 2.2.3.

A multimedia data stream is stored as blocks on a disk. The placement of media blocks required for playing back a multimedia stream is critical because of the continuity requirement, especially when multiple simultaneous streams need to be serviced. Random, contiguous, and constrained placement of blocks are discussed in (Rangan and Vin 1991). Random allocation of blocks is usually common in most storage server architectures. In such servers guaranteed support for continuous retrieval is difficult because separations between blocks of a stream may be too large to guarantee bounds on access and latency times of successive blocks. Unconstrained variation in latency may be nullified using buffering techniques. However, the cost of buffering may be too high in addition to introducing large initial latencies. Contiguous block allocation results in disk fragmentation though continuous access to blocks of a media stream is guaranteed.

Constrained block allocation results in desired average seek time while minimizing buffering requirements. The size of the blocks and the separation between them need to be determined to satisfy continuity requirement. In (Rangan and Vin 1991; Rangan and Vin 1993) an abstraction called a *media strand* is used to refer to a sequence of continuously recorded video frames or audio samples. To better utilize disk space in constrained block allocation, storage patterns of different media strands are merged in the gaps between media blocks of one strand with media blocks of other strands (Rangan and Vin 1993). Merging can be done only if the storage patterns for each strand can be maintained after merging. This scheme has the advantage of contiguous allocation for the same strand while utilizing the disk space better. Using the same concepts, Vin and Rangan (Vin and Rangan 1993) present a retrieval algorithm called *quality proportional multi-subscriber service* where a number of blocks proportional to the client's playback rate are retrieved with better performance than a fixed block retrieval scheme based on maximum playback rate.

A theoretical framework for the real-time retrieval requirements of multimedia data and placement strategies on secondary storage are discussed in (Gemmell and Christodoulakis 1992). They show that interleaved and contiguous placement strategies achieve higher performance than non-interleaved and scattered placement, though at the expense of flexibility and disk fragmentation. Srivastava et al. (Srivastava, Kumar, and Singru 1997) present an interleaved storage scheme with constrained allocation of video and audio blocks on a disk. The scheme interleaves different phases of the same movie.

In (Keeton and Katz 1995) the strand abstraction from (Rangan and Vin 1991) is called a *retrieval block* and corresponds to an integral multiple of logical playback units having a fixed playback duration in a given sequence. The data for a retrieval block is stored contiguously. However, the placement of individual retrieval blocks for a given stream is unrestricted. In the same paper, Keeton and Katz combine data layout strategies with image coding on RAID-style (Chen, Lee, Gibson, Katz, and Patterson 1996) disk arrays for storing multi-resolution video data and experiment with different retrieval block sizes. Chang and Zakhor (Chang and Zakhor 1994b) provide placement strategies for video data that is compressed using subband coding technique to accommodate users with different playback rates. Three placement strategies: consecutive placement of video data of a single rate, different disks for different rates, or interleaved frame groupings from different rates are examined. Chen et al. (Chen, Little, and Venkatesh 1995; Chen and Little 1996) provide storage and retrieval mechanism for MPEG-coded video from a single disk storage system. The disk scheduling scheme is based on round-robin algorithm with a fixed period. The time required for data retrieval in the next period is computed a priori and if there is a potential for starvation, the missed data is distributed equally among all sessions. Since the placement strategy places most important data, for example the I frames, at the beginning of a storage pattern sequence, only less important frames are unread.

Performance of RAID-style video servers has been examined in (Hsieh, Lin, Liu, and Du 1995; Barnett and Anido 1998). In (Hsieh, Lin, Liu, and Du 1995), different video file striping schemes at the application level or the device driver level are evaluated in a RAID-3 environment. In (Barnett and Anido 1998), performance of RAID-3 and RAID-5 disk arrays are compared and they conclude that for large-scale video servers, a coarse-grained striping in a RAID-style disk array is most cost-effective. An analytical performance model of a clustered video server is used to find the loss probability of requested data not available for playback by its deadline in (Tewari, Mukherjee, Dias, and Vin 1996).

Much of the work on video data placement mentioned so far is independent of the video characteristics such as their popularity. The actual distribution or assignment of videos to a particular disk or disk array must address the load balancing issue. Load imbalance arises as a result of the differing popularity of videos and therefore, different access patterns to the videos on individual disks in a disk array. Some of the solutions to the load balance problem are replication, striping, and proper video placement. Other runtime solutions discussed in the field are dynamic replication of partial segments or blocks of videos and request migration from overloaded disks to underloaded disks subject to availability of replicated copy of the videos on the underloaded disks.

In (Little and Venkatesh 1995), a linear predictor function is used to find the popularity of a movie using previous access history. The movie assignment to disks follows from making disk accesses equiprobable. This idea achieves both load balancing and minimum request blocking at the same time. Wolf et al. (Wolf, Yu, and Shachnai 1997) address load balancing in the context of disk groups called Disk Striping Groups (DSG). In the static component of their algorithm optimal number of copies of videos are found based on apportionment techniques. Multiple copies of a video are assigned to disk groups based on a graph theoretic concept of *cliques* and single copy videos are assigned to disk groups based on a “least loaded first” algorithm. Given this static allocation, an incremental algorithm is run periodically to account for changes in demand by minimizing the changes in assignment and number of copies. The dynamic component handles real-time scheduling of video streams to DSGs by first finding an optimal load balance and then maintaining that balance using an algorithm called *DASD Dancing*, if the balance shifts from the optimal goal by migrating requests to other DSGs. Wang et al. (Wang, Liu, Du, and Hsieh 1997) prove that the video file allocation over disk arrays is NP-hard. They present heuristic algorithms to replicate, stripe, and place videos over a set of RAID-3 disk arrays. The access profile of each video is found using normalized Geometric distribution unlike the more popular Zipf distribution.

In (Dan and Sitaram 1995), a dynamic placement policy based on the Bandwidth to Space ratio (BSR) is proposed. The idea behind the policy is to find an optimal mix of *hot* and *cold* videos with replicated copies of some videos to make the best use of bandwidth and space of a storage device. Their simulation results show that for a given storage and bandwidth capacity the policy performs better with fewer videos because of increased replication. A partial replication of video files at runtime for load balancing is proposed in (Dan, Kienzle, and Sitaram 1995). Flynn and

Tetzlaff present striping algorithms for individual movies which find different disk sequences for striping called permutations. They also study the effect of combining replication with striping by replicating striping units once. In (Venkatasubramanian and Ramanathan 1997), a load factor is computed with respect to each server and requests are assigned to servers based on the load factor. They present a heuristic method which iteratively maps a video to a storage device such that the total revenue resulting from such an allocation is maximum. Santos and Muntz (Santos and Muntz 1998) address the issue of load imbalances caused by asymmetries in the system introduced as a result of heterogeneous disk configurations with different Bandwidth to Space Ratio (BSR). They combine random allocation with replication of data blocks to balance load on disks with small and large BSR and unlike (Dan and Sitaram 1995), their approach is independent of the popularity of video because they are dealing with video data blocks. Unlike the dynamic replication scheme presented in (Dan and Sitaram 1995; Dan, Kienzle, and Sitaram 1995; Venkatasubramanian and Ramanathan 1997), Lie et al. (Lie, Lui, and Golubchik 1998) use threshold-based triggers to activate replication and dereplication in a parallel VoD architecture consisting of multiple storage nodes. In (Ghandeharizadeh and Ramos 1993), two alternative approaches based on disk multitasking and data replication are examined in a multi-processor environment. Video object fragments are assigned to different processors such that the workload on each processor is approximately equal. They conclude that replication approach outperforms disk multitasking as long as there is storage capacity to provide the number of required replicas.

Admission Control

Admission control algorithms at a video server determine whether a new client can be admitted without violating the continuous playback requirements of the clients already being serviced; that is, the service time needed to retrieve the blocks for the existing clients should not exceed the minimum playback duration of the blocks retrieved for an existing client.

In the literature, two classes of admission control strategies are discussed: deterministic and statistical. Deterministic admission control strategies are those where worst-case assumptions regarding service times are made so that the continuous playback requirements are never violated for the duration of the entire service (Anderson, Osawa, and Govindan 1992; Vin and Rangan 1993; Gemmell and Christodoulakis 1992; Gemmell 1993; Reddy and Wyllie 1993; Tobagi, Pang, Baird, and Gang 1993; Yu, Chen, and Kandlur 1992). Statistical admission control strategy is based on probabilistic analysis of finding when the rate of transfer to retrieve the total data requested by all users exceeds the disk throughput capacity (Chang and Zakhor 1994a; Vin, Goyal, Goyal, and Goyal 1994b). Deterministic admission control might result in underutilization of server resources by its worst-case assumptions. Further, given that human perception is tolerant to occasional loss of media information and that most continuous media contain redundant data, it is possible to use statistical averages in developing admission control strategies.

Chang and Zakhor (Chang and Zakhor 1994a) present both deterministic and statistical admission control strategies for two types of data placement in Variable Bit Rate (VBR) servers: Constant Data Length (CDL) retrieving equal amounts of data for each user and Constant Time Length (CTL) retrieving unequal amounts of data proportional to playback duration. In order to develop statistical admission control, convolution techniques are used to find the *pdf* of the sum of data rates for all users, including the user that seeks admission. The probability of overload is computed and compared to the threshold limit corresponding to the disk read bandwidth. The new user is denied access if the resulting overload probability is more than the acceptable level. Similar to Chang and Zakhor's CTL situation, Rangan et al. (Rangan, Vin, and Ramanathan 1992) propose a quality proportional multi-subscriber servicing (QPMS) algorithm for admission control, in which the number of blocks retrieved during each service round for each subscriber request is proportional to its playback rate.

Vin et al. (Vin, Goyal, Goyal, and Goyal 1994b) provide a statistical admission control algorithm by defining an overflow probability computed using service time distributions for different placement of data blocks on disks. During overflow rounds, minimum service guarantees are provided by discarding excess media blocks equally among the clients. In (Vin, Goyal, Goyal, and Goyal 1994a; Vin, Goyal, and Goyal 1995), an admission control strategy based on a predictive approach using historical observed values for the time to retrieve disk blocks. The new request is admitted for service only if the predicted extrapolation from current measurements of the storage server utilization indicate that the service requirements of all clients can be met satisfactorily. More recently, Lin (Lin 1998) examines two disparate goals of VoD admission control: maintaining Quality of Service (QOS) requirements for all clients and maximizing the system throughput. Two new types of admission control schemes called Enhanced Strict Admission Control (ESAC) and the Probabilistic Admission Control (PAC) are proposed. Under ESAC pre-computed statistical information is used instead of peak frame Sizes to still achieve strict guarantees. The PAC algorithms use similar statistics but allow for some lost or overdue data to achieve higher throughput.

2.2.2 Multimedia File Systems

Conventional file systems are optimized for the storage and retrieval of textual data. With the proliferation of the multimedia applications, storage and retrieval of different types of data, such as audio and video, in addition to text, are becoming more common. Anderson and others (Anderson, Osawa, and Govindan 1992) provided early insight into file systems that can handle both real and non-real time data with their CMFS file system. Shenoy et al. (Shenoy, Goyal, Rao, and Vin 1998; Shenoy, Goyal, and Vin 1999) examine the architectural considerations for next generation file systems which are expected to support heterogeneous data types using data type specific techniques. In particular, the file systems are expected to handle both QOS-driven real-time requests as well as non-real time requests. Support for multiple block sizes and intelligent and diverse placement policies will be required in such systems. Resource reservation or admission control is a requirement for real-time multimedia applications. Shenoy et al. present the design of their integrated multimedia file system called *symphony* in (Shenoy, Goyal, Rao, and Vin 1998) and evaluate two approaches, partitioned and integrated file servers. They conclude that integrated approach performs better than partitioned approach, especially for bursty loads. Usually in a resource sharing environment complete sharing between competing applications works better than dedicating a certain set of resources for each application. Two of the commercial implementations of integrated file systems are the Tiger Shark file system (Haskin and Schmuck 1996) from IBM and XFS from SGI (Holton and Das). Tiger Shark provides continuous time delivery, wide striping, and a large data block size in addition to supporting arbitrary video rates. The system also uses block-level replication strategy; when a disk fails the load on that disk is spread evenly on the remaining disks.

In addition to the file system, the operating system must provide real-time features for continuous time data access required for multimedia applications. These include: no interference from kernel and other programs with the continuous time data delivery, prevention of page faults by fixing server code and data buffers in memory, higher priority for multimedia application daemon to prevent other processes from interfering with the continuous time delivery, and real-time disk I/O to achieve uninterrupted dataflow to clients (Krikelis 1998). Large block sizes help multimedia file systems maximize throughput from the available storage.

2.2.3 Scalability

A large scale VoD system is expected to serve thousands of clients. While it is not cost effective to have a centralized server facility because of the tremendous networking requirements to reach all user populations, it is also impractical to have many replicated facilities to respond locally to individual user populations. To strike a balance between availability and cost-effectiveness, hierarchical configurations of multimedia servers and network switches are employed (Furht, Kalra, Rodriguez, and Wall 1995). Typically, a neighborhood server will handle requests from a particular user population and the server on the next level will handle requests from several user populations if they cannot be handled at their local servers. This hierarchical configuration provides scalability in addition to providing cost efficiency. Scalability is provided by using clusters of storage nodes with disk arrays. Concurrent access to many requests and parallel access to many disks for the same request in the disk array provide better throughput.

Brubeck and Rowe discuss Berkeley VOD system in (Brubeck and Rowe 1996) which consists of one or more video file servers, one or more archive servers, and an administrative server with a database. Multiple continuous media objects are cached from the archive servers to video file servers for real-time playback of a request. Cache management and load balancing among video file servers are addressed. A three level architecture with the lowest level storing a limited number of videos with high viewing probability and the top level serving as an archive is presented in (Lai, Lin, and Lai 1997). Parallel architectures are used in (Jadav and Choudhary 1995; Lee 1998). Hierarchical configurations using tertiary storage are presented in (Merchant, Ren, and Sengupta 1996; Chervenak, Patterson, and Katz 1995; Lau, Lui, and Wong 1995; Keeton, Drapeau, Patterson, and Katz 1994; Pang 1997; Jun and Lee 1998), though the necessity of having tertiary storage as a cost-effective means of storage and retrieval of multimedia data is diminishing because of the tremendous progress made in secondary storage devices; in other words, secondary storage has become cheap. In (Doganatha and Tantawi 1994) a normalized geometric distribution is used to define user access profile to movies and an allocation scheme based on cost models are used to distribute video files across tertiary and secondary storage devices and the RAM.

Video file allocation needs special consideration when dealing with multiple hierarchies of servers or even with clusters of servers with RAID-style disk arrays. Video availability is affected by the popularity of a video; the more popular is the video, more demand it will have and more resources are needed to respond to that demand. A highly skewed access to videos is not unusual. A judicious placement of popular videos across and within levels of server hierarchy will increase availability of those videos. A survey of popularity-based video access models in VoD systems is presented in (Griwodz, Bar, and Wolf 1997). They show through analysis that while Zipf distribution is appropriate for characterizing short-term popularity of videos, the long-term popularity of a video and the impact of the size of user populations on popularity distributions cannot be ignored. They present a model for request generation that takes into account the popularity of a movie through its life cycle.

In (Schaffa and Nussbaumer 1995; Nussbaumer, Patel, Schaffa, and Sterbenz 1995), an analytical study of the tradeoff between overall bandwidth and storage requirements is provided. They derive cost functions to reflect caching and replicating popular programs at certain levels of the server hierarchy. Bisdikian and Patel in (Bisdikian and Patel 1995) examine a replication strategy that puts as many of the most popular movies as possible at each server in a server hierarchy, constrained only by storage limitations. They also consider a placement scheme for movies on a r-ary tree-like network topology which strikes a balance between communication and replication costs. An early work (Ramarao and Ramamoothy 1991) examined the allocation problem as a set partitioning problem with minimizing some cost function as its objective. They consider a three-level hierarchy and partition the video database into three sets, each set being allocated to one of the three levels using an $O(N^2)$ algorithm.

Similar to the Berkeley VoD System (Brubeck and Rowe 1996), cache management schemes are examined in (Bianchi, Melen, and Rainoni 1995; Miyazaki and Nahrstedt 1997; Sonah and Ito 1998; Kim and Das 1998). In (Bianchi, Melen, and Rainoni 1995), a two level server hierarchy, cache and a video server library are used and cached objects are replaced using several replacement policies. Stationary request distribution based on rectangular distribution and non-stationary request distribution based on incremental token depletion scheme to account for slow decrease of popularity of videos are examined. In (Miyazaki and Nahrstedt 1997), a protocol to facilitate the dynamic coordination of movie locations on cache servers is described. (Sonah and Ito 1998) present an enhanced cache replacement algorithm for the Berkeley VoD system based on the additional characteristics of the media object. Kim and Das (Kim and Das 1998) provide an analytical model of interval caching. In interval caching, two consecutive streams are paired and the data for the time interval between the two streams is buffered, making new disk accesses unnecessary for the second stream. In (Barnett and Anido 1996), a cost analysis of a two level hierarchy consisting of video servers and front end servers similar to the cache servers from previous references is provided.

2.2.4 Fault Tolerance

Much work has been done on video server design such as scheduling schemes, disk layouts, and admission control. Many of these designs incorporate disk arrays to serve a large number of clients. For a single disk, the mean time to failure (MTTF) is about 300,000 hours. A server with 500 disks has an MTTF of 600 hours or about 25 days. Even a single disk failure can cause unacceptable interruptions. To provide continuity of service, data from the failed disk has to be quickly reconstructed. RAID-style servers employ data redundancy to recover lost data using parity information. However, these servers were originally designed for conventional workload which is very different from real-time, periodic, large data size workload characteristic of multimedia retrieval. Migrating requests served out of failed disks to other servers or even *piggybacking* (Golubchik, Lui, and Muntz 1996) them on to other streams from replicated copies may be a solution to maintain the continuity of presentation. Despite the efforts of previous and current research in this area, failure recovery in real-time is still an open problem, partly because of lack of techniques but more for lack of studies on the performance of implementing these techniques in real time during failure events.

Fault-tolerant design of multimedia servers has been discussed previously in (Ozden, Rastogi, Shenoy, and Silberschatz 1996; Berson, Golubchik, and Muntz 1995; Tobagi, Pang, Baird, and Gang 1993; Krishnan, Venkatesh, and Little 1997; Tewari, Dias, Mukherjee, and Vin 1996; Wang and Du 1998; Shyu and Shieh 1999). A variation on the conventional RAID arrays is the streaming RAID discussed in (Tobagi, Pang, Baird, and Gang 1993). In a streaming RAID, disks are grouped into fixed size clusters and in each cluster, one disk is dedicated to store parity information while the rest of the disks store data. For each active stream, a parity group consisting of all data blocks plus the parity block is retrieved in each cycle. In case of a disk failure, missing data can be reconstructed on the fly because the rest of the parity group is available. In (Ozden, Rastogi, Shenoy, and Silberschatz 1996), techniques based on pre-fetching data blocks in a parity group and reserving contingency bandwidth to retrieve additional blocks in the event of a disk failure are presented. Parity based fault tolerance schemes are presented in (Berson, Golubchik, and Muntz 1995) and (Tewari, Dias, Mukherjee, and Vin 1996) and each scheme is evaluated in terms of storage overhead, bandwidth wastage and memory requirement. In (Shyu and Shieh 1999), request migration schemes from failed server to one of the other working servers is presented. To prevent overload on the remaining servers, merging schemes with catch-up or replay windows are examined. Krishnan et al. (Krishnan, Venkatesh, and Little 1997) discuss merging streams by varying display rates similar to (Golubchik, Lui, and Muntz 1996) in case of server failure. In addition to manipulating display rates they examine varying content progression in a catch-up window between two streams to be merged. Content insertion is considered as a solution to fault recovery and overload situations in (Krishnan, Venkatesh, and Little 1997).

Recently, Hwang et al. (Hwang, Kilic, and Subrahmanian 1998) have addressed the issue of crash recovery which satisfies client requests with no deterioration in QoS with a prototype implementation and experimental results. They also address the important topic of updates so far ignored except for some dynamic replication and dereplication schemes in (Wolf, Yu, and Shachnai 1997; Dan, Kienzle, and Sitaram 1995; Venkatasubramanian and Ramanathan 1997). Most assume that the updates are handled during off-peak loads. In contrast, Hwang et al. (Hwang, Kilic, and Subrahmanian 1998) present a priority based model with two classes of events, user-initiated and system-initiated. System-initiated events handle updates, inserting new movies or deleting old movies even as a set of clients are being served.

Several data sharing techniques have been proposed previously. With all these techniques the on-demand nature of the VoD system and some QoS guarantees have to be sacrificed. The VoD systems employing such stream optimizations are therefore termed *Near-VoD*. A few of the references on fault tolerance indicate that these techniques may help in fault recovery such as the novel stream merging technique in (Golubchik, Lui, and Muntz 1996). Batching techniques are discussed in (Leung, Lui, and Golubchik 1997; Almeroth, Dan, Sitaram, and Tetzlaff 1997; Aggarwal, Wolf, and Yu 1996; Kim and Zhu 1998; Chan, Tobagi, and Ko 1998; Tsai and Lee 1998). Disk striping schemes optimized for batching or multicasting are presented in (Ozden, Rastogi, and Silberschatz 1997).

2.3 Network Subsystem

Traditionally, three separate industries have provided service for voice, video, and data networking; telephone companies for voice, cable TV networks for video, and computer communications for data networks. With the introduction of many distributed multimedia applications, the trend has been toward providing integrated services for all media types on a single network based on packet switching (Liebeherr 1995). Many of the promises made during the last 5-10 years with respect to multimedia applications such as interactive VoD have not been realized partly because of the difficulty in bringing together the three major industries: telephone, cable TV, and the computer industry (Chiariglione 1997). It is difficult to change the perception of cable TV companies from broadcasting to unicasting and from single user high interactivity to mass provision for the computer companies. Whatever the roadblocks are to commercial deployment of distributed multimedia applications, many of the research issues regarding the transition from a best-effort to a guaranteed and integrated services network paradigm still remain.

In this section, I present a survey of issues concerning guaranteed services over an IP based packet switching network. I focus only on this type of network because of the current technology trend toward integrating traffic of all media types over packet switching networks and the popularity of IP and IP over ATM networks. The continuity requirements of multimedia presentation demand that guarantees on timely delivery of multimedia data be provided. This is different from the best-effort policies of current IP network architectures where the service is as good as the lack of network congestion on it.

Because of their continuity requirements and real-time performance, distributed multimedia applications place a different type of burden on the existing network infrastructures and protocols. To support several audio/video streams simultaneously, the network must have the necessary bandwidth to handle all requests as well as guarantee each request continued bandwidth over a prolonged period of time. Multimedia traffic requirements placed on the network may be grouped into six categories: streaming requirements, high bandwidth, low latency, semi-reliability, multi-point communication, and integrated services (Tobagi and Dalgic 1996). Error checking and retransmission are necessary for total reliability of data networks. With compressed multimedia data however, error free transmission is a requirement but retransmission is not a solution because of the additional delays. Given these requirements, resource allocation and management to provide guarantees on bandwidth, delay, and error rates at the time of connection establishment becomes a very important issue for multimedia applications.

The stringent performance requirements of distributed multimedia applications are often stated in terms of *Quality of Service* (QoS) parameters such as throughput, network delays, delay jitter, and error rates (Liebeherr 1995; Nahrstedt and Steinmetz 1995; Chatterjee 1997). Guaranteed rates and network delay bounds are required for stored video delivery. However, for interactive live video transfers such as video conferencing applications both delay and delay jitter bounds are required. Bit error rates should be less than 10^{-6} and packet loss should be less than 10^{-11} for compressed video as compared to 10^{-2} for uncompressed video (Nahrstedt and Steinmetz 1995). Maximum end-to-end delay can be in the order of several 100 ms for stored video transfers.

Performance guarantees in terms of QoS parameters are provided using a combination of traffic reshaping using leaky bucket or token bucket controllers, scheduling schemes, and a signaling mechanism (Chatterjee 1997). This guaranteed delivery paradigm is described using the basic unit of a *flow*. A flow represents a stream of packets in response to a request from sender to receiver over the network for the duration of the request's lifetime. The definition of a flow with respect to each request makes it easy to allocate and monitor network resources (Zhang 1991). This is different from packets in a data network where each packet is its own entity. Similar to a flow definition, the term *real-time channel* is also used to signify a fixed routing path from a source to a destination with resources reserved along that path during connection establishment (Ferrari and Verma 1990). A flow or a channel with known QoS requirements is set up by using protocols such as Resource Reservation Protocol (RSVP) to reserve network resources along the request path on the Internet (Wroclawski 1997; White 1997; Zhang, Deering, Estrin, Shenker, and Zappala 1993; Braden, Clark, and Shenker 1994; Wolf, Griwodz, and Steinmetz 1997). Admission control is enforced prior to resource reservation to make sure that the QoS requirements can be met on the routing path of an arriving request.

Rate based service disciplines along with admission control policies are employed to provide guarantees on reserved rates, end-to-end delay, and delay jitters. While admission control algorithms assist in resource reservation, the service disciplines aid in resource allocation according to a service contract during data transfer (Zhang 1995). A service discipline implemented on a switch will schedule and manage the bandwidth and the buffer space and assign priority to different flows based on their characteristics (Nahrstedt and Steinmetz 1995). A minimum service rate is guaranteed to each flow regardless of the traffic pattern of other flows on the same routing path. Providing performance guarantees requires a careful selection of the service discipline because of the algorithmic differences and the guarantees provided by each discipline.

2.3.1 Network Service Options

Window-based flow control generally used in best-effort data networks are not appropriate for video traffic because loss recovery through acknowledgments and retransmissions introduce additional delays. Rate-based service disciplines have been the focus of research in the last ten years as alternate ways of traffic control in integrated services network (see (Zhang 1995) for a survey). Rate-based service disciplines provide delay bounds such as maximum end-to-end delay bound, bounds on minimum end-to-end delay leading to a delay jitter bound, and guaranteed throughput for multimedia traffic.

Two classes of service disciplines are discussed in the literature: *rate allocated* and *rate controlled* disciplines. The difference between them is that under rate controlled service disciplines traffic is reshaped at each switch through the use of a regulator and a scheduler. Another classification groups service disciplines as either work conserving or non-work conserving. Work conserving disciplines ensure that the server is never idle when there are packets to be transmitted. On the other hand, under non-work conserving disciplines the server may be idle even when there are packets to be transmitted. The reason for this is that the packet transmission times are explicitly computed for each packet based on some rationale and this feature works to control and reshape traffic along the routing path.

Rate allocated disciplines are work conserving and among them are: Delay Earliest Due-Date (Delay-EDD) (Ferrari and Verma 1990), VirtualClock (Zhang 1991), Fair queueing (FQ) (Demers, Keshav, and Shenker 1990), and variants of Fair Queueing as in Weighted Fair Queueing (WFQ) (Parekh and Gallager 1993; Parekh and Gallager 1994), Self-Clocked Fair Queueing (SCFQ) (Golestani 1994), and Worst-case Fair Weighted Fair Queueing (WF^2Q) (Bennett and Zhang 1996). The basic idea behind Delay-EDD (Ferrari and Verma 1990) is to combine deadline-based scheduling such as Earliest Due Date (EDD) with an explicit computation of deadlines to packets based on their overall delay bound. The scheme distributes the overall delay bound, D_i among the nodes on the path of a channel to be established. Therefore, at node n, a delay bound of $d_{i,n}$ is used to compute the deadline of an arriving packet, say at time t, as $t + d_{i,n}$. The admission control test will determine if every node on the routing path can satisfy the bound $d_{i,n}$ before establishing a channel or a connection. The VirtualClock algorithm (Zhang 1991) mimics Time Division Multiplexing (TDM) by assigning each packet a VirtualClock time based on resource reserved for that flow as well as the arrival pattern and ordering the transmission of packets based on their assigned times. A misbehaving flow can consume idle resources by sending data at a higher rate than specified at the time of connection set up. But service level guaranteed to other flows is not disrupted because of the service order established using virtual clock according to resource reservation.

A Fair Queueing algorithm called Generalized Processor Sharing (GPS) scheduling policy as originally formulated is an idealized scheduling discipline. This is because the discipline assumes that nonempty queues from different flows may be simultaneously serviced using infinitely small amount of data which is not practical. Each session or flow is served using a minimum reserved rate, but excess bandwidth is redistributed among all flows according to their reserved rates which results in a max-min fair allocation (Demers, Keshav, and Shenker 1990). Because GPS is not implementable, other algorithms which approximate GPS have been proposed. WFQ or packet GPS (PGPS) (Parekh and Gallager 1993; Parekh and Gallager 1994) is one such approximation which relaxes the assumption of infinitesimal data of GPS by using packets as data units and serving them in the order that would have been used in an idealized GPS. Implementation complexities arise because WFQ simulates GPS in parallel to derive the service order using only the finish times of packets in the corresponding GPS system. WF^2Q (Bennett and Zhang 1996) discipline differs from WFQ in that it uses both start times and finish times of packets in the reference GPS which achieves a more accurate emulation of GPS. In both cases, computation of the start and finish times for packets is quite expensive for a scheduler serving a large number of connections ($O(N)$ complexity where N is the number of active sessions). Despite these problems (Keshav 1997) indicates that many leading manufacturers of routers and ATM switches are implementing some form of WFQ scheduling as early as 1996.

Self-clocked Fair Queueing (SCFQ) (Golestani 1994) simplifies the computation of start and finish times for packets by using the finish times of the packet currently in service. Simplification results from not having a reference GPS to compute the finish times. However, fairness may be sacrificed in certain instances as a result of this simplification and the delay bounds are larger than WFQ. More recently, a number of algorithms which have the same or comparable delay bounds and similar fairness properties of a WFQ but without the computational complexity have been proposed (Chiussi and Francini 1998; Bennett and Zhang 1997; Suri, Varghese, and Chandranmenon 1997). See (Stiliadis and Verma 1998a; Stiliadis and Verma 1998b) for an analysis of FQ algorithms.

Rate controlled disciplines such as Jitter-Earliest-Due-Date (Jitter-EDD) (Verma, Zhang, and Ferrari 1991), Stop-and-Go (Golestani 1991), Hierarchical Round Robin

(HRR) (Kalmanek, Kanakia, and Keshav 1990), and Rate-Controlled Static Priority (RCSP) (Zhang and Ferrari 1993) are all non-work conserving disciplines. In a non-work conserving service discipline the server may be idle even when there are packets to be transmitted. This works to reshape traffic along intermediate nodes and therefore the rate controlled disciplines mentioned above provide delay jitter bounds in addition to the maximum end-to-end delay bound and bandwidth guarantees. Delay jitter is defined as the maximum difference between end-to-end delays experienced by any two packets. Rate controlled disciplines reshape traffic at intermediate nodes in the following way. A rate controlled server consists of a regulator and a scheduler. The regulator holds the packet until it is eligible for transmission at which point it places the packet in a scheduler which then transmits the packet. The eligibility time is computed using one of two regulators, a delay-jitter regulator or a rate-jitter regulator. A delay jitter regulator computes eligibility time as the sum of the eligibility time at the previous *switch*, the delay bound at the previous switch and the largest possible link delay from the previous to the current switch. Therefore, the output stream from a regulator is a time shifted traffic fully reconstructed at every regulator. A rate jitter regulator on the other hand computes the eligibility times as a sum of the eligibility time of the previous *packet* on that connection and the peak rate service time of a fixed size packet (Keshav 1997).

Jitter-EDD (Verma, Zhang, and Ferrari 1991) is a combination of delay jitter controlling regulator and an earliest-due-date scheduler. The eligibility time per packet for jitter-EDD is found as follows (Zhang 1995):

$$e_{i,j}^{k} = a_{i,j}^{k} + Ahead_{i-1,j}^{k} \tag{2.1}$$

where $e_{i,j}^{k}$ is the eligibility time and $a_{i,j}^{k}$ is the arrival time for k^{th} packet from j^{th} flow on i^{th} switch and $Ahead_{i-1,j}^{k}$ is the amount of time the packet is ahead of schedule at the $(i-1)^{th}$ switch. All packets receive the same delay on all but the last link. The delay jitter for the flow is the delay jitter on the last link.

Stop-and-go (Golestani 1991) uses a framing strategy; the timeline is divided into constant length periods called frames. All packets arriving in one frame are transmitted in the next frame. The arriving frame has a corresponding departing frame separated by a constant delay. Packets of the same frame at the source stay the same throughout the network. The delay of any packet is bounded by two frame times. HRR (Kalmanek, Kanakia, and Keshav 1990) is an extension of round robin scheduling and uses a similar frame-based strategy as stop-and-go.

Disciplines based on earliest due date algorithm such as Delay-EDD or Jitter-EDD do not have the tight coupling between bandwidth and delay present in FQ algorithms. Under FQ algorithms, a higher bandwidth has to be reserved to receive smaller delays. While the flexibility of EDD algorithms is desirable, there is still the implementation complexity of sorted priority with those algorithms. To get around this undesirable effect, rate-controlled static priority (RCSP) (Zhang and Ferrari 1993) was proposed. As the name indicates the scheduling scheme is based on a static priority scheduler. The RCSP server consists of a regulator for each connection and a scheduler. The regulator may be delay-jitter based or rate-jitter based. The scheduler works on static priority in multiplexing eligible packets for transmission by choosing packets from the highest priority level first. A priority level is assigned to each connection at the connection establishment phase. The delay bound depends on the priority level specified for the connection.

In (Georgiadis, Guerin, Peris, and Rajan 1996), it is shown that using appropriate shapers on the intermediate network nodes will not affect the end-to-end delay bound but will increase the average delay. However, end-to-end delay jitter will be significantly reduced leading to reduced buffer requirements at each network element through reshaping. As a general rule, reducing delay jitter has beneficial effects, but the question is at what cost, memory versus network complexities (see (Partridge 1994) for a brief discussion). For their analysis, they make use of the concept of an *envelope* function for the source traffic and the bounds on delay and buffer requirements as first introduced by Cruz in (Cruz 1991; Cruz 1995).

2.4 Client Subsystem

Clients of a VoD system use a device generally known as the set-top box (STB) to interact with the video servers over an integrated services network. Most of the proposed architecture for STBs assume that cable TV networks will coexist with some kind of packet switched integrated services network. In (Kohiyama, Shirai, Ogawa, Manakata, Koga, and Ishizaki 1996), an architecture of a 64QAM MPEG-2 digital set-top box for VoD applications is described. The architecture integrates high resolution (720X480), color (256 colors our of 17.6 million) graphics, MPEG-2 video and analog NTSC television signals. If current television technology is kept as an output medium for VoD services, a digital video signal has to be converted to an analog NTSC signal before it is displayed on a conventional TV (Furht, Kalra, Rodriguez, and Wall 1995). Furht (Furht, Kalra, Rodriguez, and Wall 1995) proposes a hardware architecture of an STB as consisting of a processing subsystem which runs a real-time operating system, a video subsystem which decompresses encoded video streams such as MPEG-2 streams, possibly using MPEG-2 decoding chipsets, and an audio subsystem which decodes the audio corresponding to the video stream.

In this dissertation, buffer management is the issue we are most concerned with for the client subsystem. Gollapudi and Zhang (Gollapudi and Zhang 1998) examine buffering requirements necessary to maintain continuity and synchrony of presentation of multimedia objects. This is done by taking into account the network and server delays while matching pre-loading and consumption times of media objects. Biersack and Geyer in (Biersack and Geyer 1999) derive buffer requirements for synchronization schemes to playback stored multimedia streams from a single server or striped streams from multiple server nodes; their synchronization schemes do not use global clocks but instead rely on time stamps for synchronization. Chang and Garcia-Molina (Chang and Garcia-Molina 1999) provide an algorithm called MEDIC for managing memory and disk caches on a client PC or Set-top box. The video data may be downloaded, but instead of waiting for the complete download, the data will be played back as soon as some of the initial load arrives. The Medic algorithm will direct the data either to the decoder if there is enough memory or to the disk. Downloading an entire video and providing interactive playout at a client station is possible (Chen and Kandlur 1995). Copyright protection however, will have to be enforced for commercial multimedia content. Griwodz et al. (Griwodz, Merkel, Dittman, and Steinmetz 1998) present schemes which may be used in content protection while delivering major portions of multimedia content through caching and pre-fetching. In this dissertation, we only consider periodic delivery of segments of multimedia content to a client. Therefore, downloading an entire video and copyright issues do not arise.

2.5 Compression

Compression is probably the most important enabling technology for the VoD systems, next to the storage technology of recent years. Without compression, it is not possible to store and deliver videos to thousands of subscribers with reasonable resources. For instance, the bit rate of a CCIR 601 digital television signal is 216 Mbps (calculated using 30 frames per second, 858 X 525 luminance samples, 429 X 525 X 2 chrominance samples and 8 bits per sample) (Riley and Richardson 1997). Distributing such a bit stream even over future gigabit networks is still very expensive. Fortunately, several video coding techniques have been developed that can be used in compressing the video information prior to storage and delivery and decompressed before displaying it. Video compression ratios of 50-100:1 make it possible to store and deliver the videos using current storage and network technology.

The basic idea behind compression techniques is to remove redundant information from digital video data prior to storage and delivery using a coder and reinserting it before playback using a decoder. The redundant information is found in primarily two forms: statistical and psychovisual (Lodge 1992). Statistical redundancy occurs because the sample values of a picture or a video are not independent, but rather correlated with the neighboring values so the signal is predictable from its past.

Statistical redundancy could manifest within the frames called spatial redundancy or across frames as temporal redundancy. The psychovisual redundancy refers to the fact that human perception is tolerant to some distortion that the original picture sample values need not be reproduced with the same degree of accuracy. In addition to getting rid of redundancy, the compression techniques also provide ways to efficiently code the remaining information for storage and delivery. Most compression techniques consist of the following steps: representation of the picture content information, quantization, and encoding. Representation of the picture data consists of dividing an image into subimages or subblocks. Each subblock is represented as a combination of luminance and chrominance components. Luminance component is given higher weight because the human eye is more sensitive to brightness than to color or chrominance. The YUV component is a variant on the usual RGB component of a picture, giving luminance higher importance than chrominance components. MPEG coding schemes use the YUV components in the ratio 4:1:1 or 4:2:2. The individual values of blocks of pixels are scaled using quantization reducing the amount of information used to represent the picture information. The resulting data is encoded to achieve compression.

Different compression techniques include predictive coding, transform coding, vector quantization and subband coding (Waltrich 1993). Differential pulse code modulation (DPCM) is a predictive coding technique which estimates the value of a pixel based on the values of preceding pixels. A linear predictor function of weighted values of preceding pixels is used. The prediction error between predicted value and the actual value is quantized and transmitted resulting in reduced bit rates. The error signal is further encoded using a variable length coding technique such as Huffman coding. Huffman coding works by assigning codewords to data values, shorter codewords to more frequent values and longer codewords for infrequent values. Transform coding is used to convert spatial domain information of an image to other domain. The transform coefficients are encoded and inverse transforms are employed at the receiver to reconstruct the original image. Most widely used transform technique is the Discrete Cosine Transform (DCT). The contents of an 8 X 8 or 16 X 16 pixel blocks are transformed into a series of coefficients. Only a few of these coefficients contribute to the major portion of image content. These coefficients are further selected after discarding coefficients which are not so useful using methods such as zonal coding or threshold coding. In vector quantization, an image is divided into a number of blocks, each represented as an N-dimensional vector where N is the number of pixels in a block. These image vectors are compared with standardized codevectors and the codevector which matches closely is transmitted to the receiver. In subband coding, the image is represented by a set of frequency subbands and the subbands are coded using other compression techniques such as DPCM. Motion compensation in conjunction with these compression schemes is used to take advantage of frame to frame redundancy. A motion vector derived by finding the displacement of a block in question in one frame to the next using search windows are transmitted. For instance, the Motion Picture Experts Group (MPEG) compression standard uses DCT compression for within the frame compression and temporal DPCM technique for motion compensation across frames.

Commercial interests in video communications with applications such as video-on-demand prompted the need for standards in coding digital audio and video signals for storage and transmission. The results of that standardization process are MPEG-1, MPEG-2, and the recent MPEG-4 video coding standards (LeGall 1991; Morris 1995; Sikora 1997; Chiariglione 1997). By standardizing only the bit stream formats and decoder structures, a lot of flexibility in encoding efficiency is still made available to manufacturers. MPEG-1 standard was established in 1992 with a mandate for providing VHS quality video at about 1.5 Mbps. MPEG-2 was developed with a view to providing NTSC/PAL quality and up to CCIR 601 quality video with target bit rates of 2-10 Mbps. The MPEG compression algorithm uses a combination of DCT transform coding using 8 X 8 blocks and interframe coding using DPCM for motion compensated prediction between frames. MPEG encoding results in three types of pictures based on the combination of intraframe and interframe coding. The intrapictures, I-frames are coded without reference to other frames. Further coding efficiency is brought about by using motion compensation to derive interframe predicted pictures, P-frames using the nearest I-frame as a reference. Bi-directional predicted/interpolated pictures called the B-frames are coded using past and future frames as references. B-frames by themselves do not provide the necessary information for reconstructing the video. A particular combination of I, P, and B frames in a video sequence is up to the user depending on tradeoffs between compression efficiency, coding complexity, degree of random access and FF(fast Forward)/FR(Fast Rewind) functionality.

The size of a video frame depends on the level of compression achieved as well as the underlying scene complexity within a video sequence. The bit rate of the compressed video is inherently variable unless some rate control measures are adopted. It is possible to tailor an MPEG bit stream into some target bit rate by varying the quantization level. For instance, coarse quantization results in higher compression rate but may degrade quality. MPEG provides the flexibility of specifying quantization levels at the encoder which makes it possible to produce constant bit rate (CBR) streams for storage and transmission. Rate control algorithms vary the quantization level with feedback control or discard high frequency DCT coefficients or use buffers to generate a constant bit rate (Lam, Chow, and Yau 1994; Lam, Chow, and Yau 1996; LeGall 1991; Sikora 1997). Rate control algorithms are not part of MPEG standard and it is left up to the manufacturers to come up with efficient rate control algorithms. Without rate control or quantization feedback or smoothing buffers, the generated video stream is usually variable bit rate (VBR) because of the variations in scene complexity within a given video sequence.

The important issue is whether to generate CBR or VBR compressed video stream. Because the frame sizes vary with the compression level and scene complexity, VBR compression produces a constant quality bitstream whereas generating a CBR bitstream involves introducing image quality fluctuations due to discarding of information to produce the constant bit rate. On the other hand, transmission of CBR bitstream makes for simple resource reservation and admission control. However, if choosing CBR, the bit rate has to be kept high at may be peak rate to produce the same level of quality as from VBR which results in low utilization of resources. The issue is to maintain the image quality using VBR compression and transform the variable bitstream to CBR transport. Several bandwidth smoothing algorithms have been proposed (Liew and Tse 1996; Liew and Tse 1998; Salehi, Zhang, Kurose, and Towsley 1998; Feng and Rexford 1997; Feng 1997) which employ true VBR streams as input and smooth out the bursts using different techniques. The result of bandwidth smoothing is a significant reduction in the stream's peak bandwidth rate compared to the peak rate requirement of an unsmoothed VBR bitstream.

In (Salehi, Zhang, Kurose, and Towsley 1998), a piecewise-CBR transmission schedule at the server is derived off-line for each video stream by considering the client buffer size and the consumption rate. The idea is to find a feasible segmented line between the cumulative buffer occupancy curve and the consumption curve and optimal algorithms to achieve this balance are provided. Similar to (Salehi, Zhang, Kurose, and Towsley 1998), off-line smoothing algorithms are discussed in (Feng and Rexford 1997; Feng 1997) for minimizing rate increases and peak transmission rates. Instead of sustaining a piecewise-CBR transmission from the server, an alternative approach is to adopt a constant time length (CTL) retrieval and transfer method as discussed in (Chang and Zakhor 1996). Under CTL, the video data is stored and retrieved in unequal amounts to conform to the real-time playback duration. This will prevent overflow at the client buffer. Liew and Tse (Liew and Tse 1998) provide a control-theoretic approach for adapting VBR compressed bitstreams to CBR transport by using an adaptation controller which uses the change in buffer occupancy at the smoothing buffer to change encoding levels at the encoder to produce a CBR output. The design of the controller must be such that the image quality and buffer occupancy fluctuations are minimized and unnecessary control of the encoder is avoided. In (Liew and Tse 1996), a video aggregation method is provided which multiplexes several VBR streams and the aggregated video bunch will have CBR requirements over a long term period assuming that it is highly unlikely that the streams will peak together.

2.6 VoD Performance Modeling and Simulation

Much of the research in VoD systems has centered around video server design. Analytical performance models of the overall VoD systems have not received much attention. Closely related to our own research are the following two papers. Won and Srivastava in (Won and Srivastava 1996) have proposed an analytical model that computes blocking probability in a hierarchical configuration for a VoD system. A two-tiered scheme of tape drives and staging disks is assumed. Video files are loaded on to staging disks from the tape drives if they are not already loaded. All video files are assumed to be of the same length and size resulting in the same service time at the staging disks and the tape drive. A probabilistic model based on Poisson distribution is used for computing the probability of staging disks being full or tape drives being busy, either one of which will result in blocking of the request. Computing the probability of staging disks is omitted for reasons of analytical complexity.

Li et al. in (Li and Liao 1996) present a performance model for an interactive VoD system. Their architecture includes local video servers at tier 0 with magnetic disks as their primary storage devices and metropolitan servers at tier 1 consisting of both magnetic and tape drives. Servers at each level of the hierarchy are modeled as an M/M/C/K queueing node (Kleinrock 1975; Gross and Harris 1998) with local servers responding to popular movie requests only and metropolitan servers serving unpopular movies from the archives as well as blocked requests from local servers. Additional loading time from archives to on-line storage is ignored. Since each request is either routed to the local server or if blocked, routed to the metropolitan server, it is not difficult to find the overall arrival rate of requests at each node and use that rate to compute blocking probability. Only a combination of a single local server connected to a metropolitan server is modeled citing the similarity between all local servers and metropolitan servers.

Almeroth and Ammar compare unicast (single stream per customer) versus multicast (single stream per many customers) delivery of video to VoD customers through simulation (Almeroth and Ammar 1996). Multicast delivery is made possible by waiting for a small time interval called a slot length to accumulate requests for the same movie and then multicasting it to those viewers. It is assumed that the underlying network is capable of multicasting and any other network details are not considered. Length of the videos are the same across video selection. Blocking probability is used as the performance metric. In (Kalva and Furht 1996) a simulation study of VoD system using CATV network is presented. They compare multicasting and unicasting using segmentation techniques. A movie is divided into segments and the segments are transferred at a faster rate than the playback rate so that the channels may be used for other purposes. Simulation studies previously mentioned sacrifice the on-demand nature of the VoD system by using multicasting; however, in (Hua, Cai, and Sheu 1998) a third alternative called *patching* to unicast or multicast is considered using simulation. They show that this technique does not delay service in anticipation of batching requests and therefore preserves the true on-demand nature of VoD. The patching technique involves additional buffering between content transferred over multiple channels so that requests can download data from one channel for future consumption while playing back data from another channel. Freedman and DeWitt in (Freedman and DeWitt 1995) present a simulation study of a video server based on SPIFFI, a scalable parallel file system implemented on an Intel Paragon. They do not address details of the network and assume that it is not a bottleneck.

Because of their stringent real-time requirements, VoD systems have to limit the number of requests sharing the overall resources of the system. Analyzing their performance is similar to a large class of capacity allocation problems in shared resource environments. There is a long history of research in this area from the late 70s and through the 80s in the context of communication networks (Aein 1978; Kaufman 1981; Roberts 1981; Schwartz and Kraimeche 1983; Jordan and Varaiya 1991). Most of this research was the result of the progress made in queueing network theory evident in (Baskett, Chandy, Muntz, and Palacios 1975). The classic Erlang Loss model has been extended to consider multiple classes of customers each with its own service demands and arrival patterns. Efficient methods based on recursion similar to that presented in (Buzen 1973) have been developed to compute performance measures. Efficient methods to compute the normalization constant presented in (Buzen 1973; Kaufman 1981; Roberts 1981) refer to a complete sharing scenario, where all classes of customers have the same access to the capacity of a resource.

Aein (Aein 1978) introduced a simple birth-death model that analyzed access constraints imposed on user classes as a strategy to sharing transmission capacity. A product form solution was derived. The analysis was limited to a blocked-calls-cleared mode of operation which otherwise can easily become intractable. An exhaustive enumeration of the state space and the associated state probabilities to compute blocking probability is computationally expensive and might even be impossible due to hardware limitations. Kaufman and Roberts (Kaufman 1981; Roberts 1981) independently provide a one-dimensional recursive scheme to compute blocking probability that is computationally more efficient. In (Schwartz and Kraimeche 1983) the product form state equations are used in the context of a wideband TDM channel providing capacity to a mix of both circuit switched and packet switched traffic. Similar one-dimensional recursion is used to compute blocking probability. In all of the papers, the recursion is limited to the complete sharing policy; that is, there are no class-based restrictions on how the resource capacity should be shared among classes.

Tsang and Ross in (Tsang and Ross 1990) present algorithms to compute class-based exact blocking probabilities in the context of circuit-switched multi-rate tree networks. A multi-rate tree network consists of multiple access links each supporting a class of request connected to a common link. The call establishment algorithm accepts a new request only if sufficient capacity is available on the access link *and* the common link. The blocking probability is therefore calculated as the probability of the access link *or* the common link being full.

2.7 Summary of Related Work

In this section, I cite the topics that have an immediate relation to the research presented in this dissertation. The list of topics are:

- Admission control
- Analytical model of distributed VoD systems
- Simulation of distributed VoD systems
- Integration of real-time disk and network scheduling schemes in VoD modeling
- Impact of server and network performance on client subsystems

Admission control is an important concept for enforcing a certain request handling mode within a VoD system. However, the topic of admission control is usually addressed in the realm of either disk access level or in the framework of a network reservation protocol. The concept of end-to-end admission control that includes server and network in tandem has not received much attention. In fact, in most previous work the network subsystem is not included or simple assumptions are made about the network. In addition, research efforts in admission control (Anderson, Osawa, and Govindan 1992; Vin and Rangan 1993; Gemmell and Christodoulakis 1992; Gemmell 1993; Reddy and Wyllie 1993; Tobagi, Pang, Baird, and Gang 1993; Yu, Chen, and Kandlur 1992) do not differentiate between request classes. In (Chen and Chen 1996;

Chen and Hsi 1998), high and low priority classes are defined but the resource capacity is partitioned and admission control enforced on the dedicated partition for each class. The loss rate is found by modeling each dedicated partition of server capacity as a separate M/M/C/C queue. In this dissertation, the admission control scheme discussed in (Chen and Chen 1996; Chen and Hsi 1998) is called complete partitioning admission control or dedicated access (Aein 1978). Blocking in dedicated access is easily computed using Erlang loss model whereas computing blocking for complete sharing or partial sharing based on thresholds requires computational algorithms.

Distinguishing between request classes for providing access based on their merit is advantageous. For example, restricting access to low priority requests so that more capacity is made available to high priority requests may result in higher revenue. Some of the research in resource allocation algorithms ((Aein 1978; Ross and Tsang 1989)) mention this advantage in the context of a single resource based on thresholds which can be extended to a complex VoD architecture. Computation of blocking probabilities for threshold-based resource usage has not been done before.

Analytical models of distributed VoD system from (Li and Liao 1996; Won and Srivastava 1996) are limited to video servers and do not include the network subsystem in their analysis. The simulation studies from (Almeroth and Ammar 1996; Kalva and Furht 1996; Freedman and DeWitt 1995) assume that the network is not a bottleneck resource. No other details about the network are included. The analysis of the Berkeley VoD system in (Brubeck and Rowe 1996) does include the network component in their load balancing scheme. But the network is characterized only by its bandwidth capacity. The same is true of the movie placement algorithm presented in (Bisdikian and Patel 1995) where they represent the network as an r-ary tree with each link characterized by its bandwidth.

Research on network disciplines typically do not present an explicit model of server retrieval mechanism. They usually assume a leaky bucket type controller at the server as the packet source. The potential benefit of controlling network delay jitter and reshaping traffic has been previously studied in a general networking framework (Georgiadis, Guerin, Peris, and Rajan 1996; Liebherr, Wrege, and Ferrari 1996) but not in the context of a periodic server transmission schedule which is expected to sustain a continued display on a client subsystem. Client buffer requirements with a focus on synchronization is the topic of research in (Gollapudi and Zhang 1998; Biersack and Geyer 1999) without an explicit model of disk and network scheduling and its impact on client display requirements.

2.8 Evaluation of Previous Research

Much of the research in VoD systems has focused on analyzing the individual subsystems such as video servers. Multimedia data transfer over network subsystems has received much attention but not necessarily in the context of a specific application. I mentioned several issues addressed by the research community in the areas of server and network subsystems. What is lacking in all of these analyses is a comprehensive modeling of VoD system as an end-to-end system. Previous research on video server design has shown several ways of maximizing retrieval effectiveness. But its implication on the network or the client subsystem is missing. On the other hand, several network service disciplines have been proposed. But the impact of a server transmission pattern coupled with a known network service discipline on the display requirements of a client has not received much attention.

While there is merit in discussing issues related to individual subsystems, I argue that a complex VoD architecture needs to be evaluated as an end-to-end system for the following reason. Because of the limited resources and guaranteed service requirement for each request, the number of simultaneous requests being served at a resource needs to be limited. Therefore, some of the arriving requests will have to be turned away when there are no adequate resources to provide guaranteed services to those requests. Based on this mode of request handling, we have to maximize the resources so as to minimize rejection of service to incoming requests. This requires a careful analysis of all the subsystems involved in the VoD architecture. Contrast this logic with the other mode of request handling similar to best effort paradigm, namely, accept all incoming requests. In this mode of operation, if the system gets overloaded the service will get degraded. Therefore, the analysis can be limited to isolated subsystems to verify the level of performance obtained in each subsystem at various workloads.

The basic premise I build this dissertation around and also argue as missing from previous research is the following:

- a comprehensive modeling of VoD infrastructure that includes an integrated model of the server and the network subsystems with analysis of its impact on client operations.

I summarize below the limitations of previous research as addressed by this dissertation in the ensuing chapters:

- an integrated model of the VoD system that unifies the video server retrieval and network transfer mechanism.

- threshold-based admission control for the VoD system.

- comprehensive simulation of a distributed VoD architecture consisting of server hierarchies over local and wide-area networks.

- designing request handling policies and global performance metrics for evaluating the *overall* performance of a distributed VoD system.

- impact of traffic reshaping in the network on client display requirements under a given server transmission schedule.

CHAPTER 3

PROPOSED INTEGRATED FRAMEWORK FOR VOD SYSTEMS

In the previous chapter, I showed that the VoD system is a complex system consisting of distinct subsystems. Each of these subsystems is complex in its own right involving critical design issues. As pointed out in the previous chapter, several of these issues have been addressed by other researches.

In this chapter, I focus on an end-to-end approach to modeling the VoD system and developing admission control conditions. I develop a unified model of a hierarchical VoD system by integrating the storage and the network subsystems. Rather than restricting the analysis to an isolated subsystem, the VoD system is modeled as an end-to-end system. The significance of this integrated analysis is the development of a method for designing and measuring the effectiveness of end-to-end admission control and request handling policies. This methodology is used later in the simulation.

This chapter is organized as follows. In Section 3.1, the VoD architecture is presented. Ideas behind the server and network models are discussed in Sections 3.2, 3.3 and 3.4. In Section 3.5, I develop an integrated admission control model. The model is extended to multi-rate service in Section 3.6 and a chapter summary is provided in Section 3.7.

3.1 Hierarchical VoD Architecture

A typical VoD architecture consists of three critical subsystems: single or clusters of video servers, high speed wide-area and local distribution networks, and many user populations. The proposed hierarchical VoD system architecture consists of local and remote sites. Each site is characterized by a cluster of video servers. The video servers deliver high quality digitized multimedia data to clients over local distribution networks from local sites or over high speed networks from remote sites. The set-top boxes at the client site provide the decoding and display functionality, in addition to providing buffers for periodically delivered video segments from the video servers.

Each local cluster is dedicated to a single user population, referred to as the *reference* user population. The local cluster may store a complete or a partial set of videos from the video collection. Each video is stored on a server in the cluster and video segments are periodically delivered in response to a client's request. The most popular videos are replicated and stored on different servers in the local cluster. A video replication algorithm that distributes the video collection between local and remote clusters based on video popularity is proposed in Chapter 5. Also discussed in that chapter is a placement algorithm which assigns videos to servers within a cluster.

The service from the local cluster is provided over a local distribution network, such as an ATM LAN or HFC or xDSL. It is assumed that there are sufficient network resources at the local distribution network to deliver videos to the clients and that there is no resource contention on the local distribution network. This is a reasonable assumption, because the local cluster acts as a neighborhood cluster of servers and as the overall VoD user population grows, more new local clusters are added. When a new neighborhood local cluster is added to the distributed VoD system, the network capacities can be sized to match the reference user population for that neighborhood. The requests originating from a reference user population are best served by its own local site because of the absence of network contention. However, clusters of servers have limited bandwidth and therefore reject requests when there are not enough resources to serve them. The local cluster of servers administers admission control tests before accepting new requests.

The remote site may be archival in nature, providing a permanent repository for

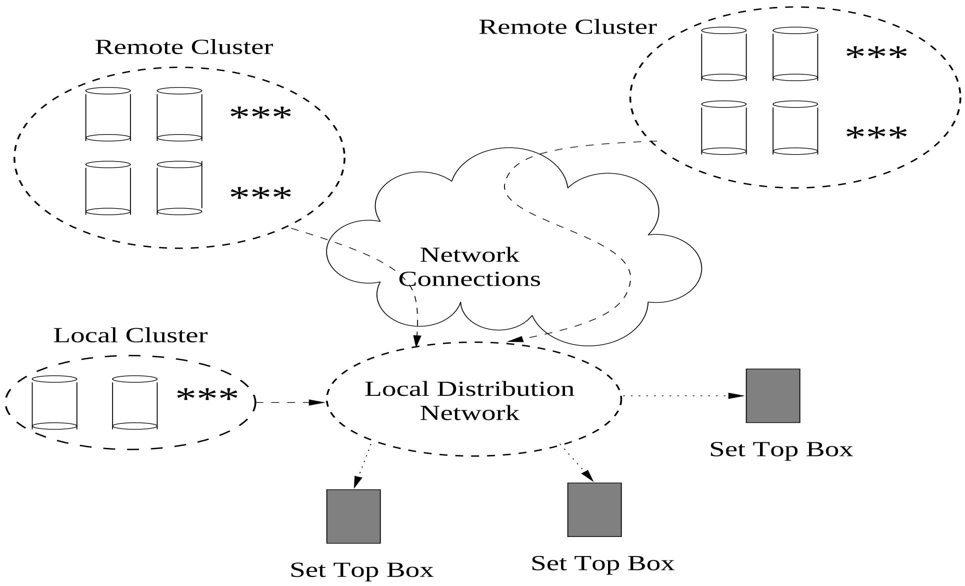

Figure 3.1: Hierarchical VoD architecture

all videos, or they may act as replicated servers such as mirrored sites. Remote servers also provide video delivery service over high speed networks. There is resource contention on the high speed networks connecting to remote sites. The remote sites may provide service to many user populations. A portion of the remote cluster capacity may be dedicated to each user population or they may be dynamically shared by the user populations.

A typical organization of local and remote clusters in reference to a single user population is shown in Figure 3.1. The hierarchy of clusters of servers and networks results in a scalable VoD system by providing multiple resources to the user population; if a request cannot be served from the local site, it may be directed to other remote sites. The requests are generated by the reference user population. Each request is assigned its own video data stream from either the local or the remote server, so that there is no batching among request streams in the servers. Each video request is served for the duration of the video playback. Though I do not model interactivity, there is nothing in the architecture or in the ensuing analysis that will prevent extending the analysis easily to include VCR-like functions.

3.2 End-to-End Admission Control

Real-time requirements for multimedia data transfer dictate a tight coordination among server and network subsystems for remote transfers. Current data management practices both at the server and the network levels must be modified to enable this tight coordination. The modifications at the server take the form of buffer management and round length restrictions while at the network, rate-based scheduling schemes provide the necessary bounded delays in addition to the bandwidth guarantees. Resources must be available at each subsystem before a new request is admitted into the VoD system. At the server, an admission control algorithm checks if the required disk bandwidth is available. At the network, a routing *path* must be set up and resources reserved along that path. The required bandwidth must be guaranteed for a bounded delay requirement. The unified admission control algorithm must reject the arriving request if any of these requirements cannot be met.

3.3 Server Retrieval Model

Previous research presented in Chapter 2 on video server design showed us that servers based on RAID-style disk arrays are appropriate because of their retrieval efficiency as well as cost effectiveness (Barnett and Anido 1998; Keeton and Katz 1995; Tobagi, Pang, Baird, and Gang 1993). The proposed server model consists of a storage architecture based on high capacity and high bandwidth RAID-5 storage. Using such a video server, the data is transferred to the client periodically; that is, the requests are processed in *rounds* (Rangan and Vin 1991). Within each round, a disk scheduling algorithm such as Scan EDF (Reddy and Wyllie 1993) or Grouped Sweeping Scheme (Yu, Chen, and Kandlur 1992) must be employed to provide guaranteed real-time retrieval. One of the key features of the unified model is that the buffer size at the client's set-top box determines the amount of data retrieved at the server and sent over the network to the client in each round using a *double buffer scheme*. The round

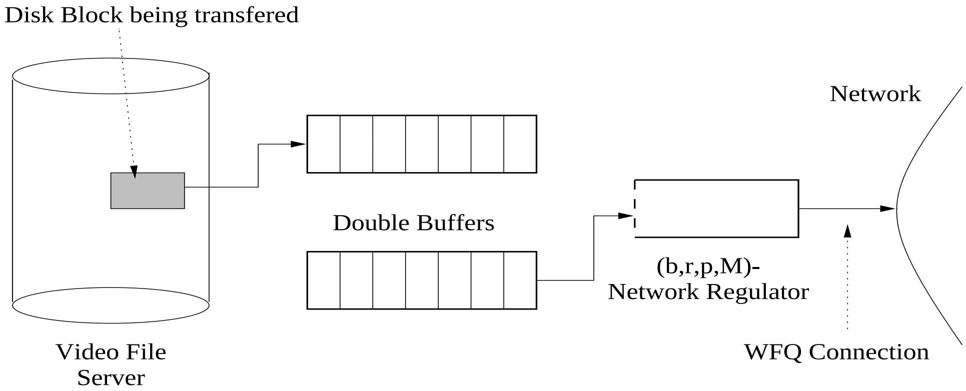

Figure 3.2: Double buffer at the server with fair queueing network

length itself, is determined by the duration of the data in the client's buffer played back at a known rate called the playback rate. Starvation at the client buffer is prevented by delivering the new data before the end of the round length. Use of double buffer in a video server has been mentioned before (Lee and Du 1997; Chang and Zakhor 1994b). In the double buffer (see Figure 3.2), as the first buffer is being filled from the disk, the second buffer is being emptied into the network and buffer roles are switched at the end of each round. The advantage of using this scheme is that the order in which requests are served from the disk is not important in the first buffer. However, the request data is injected into the network from the second buffer in a constant order.

The video data stream is MPEG compressed. In addition it is assumed that an efficient smoothing algorithm has been employed to smooth the true VBR stream into a CBR stream (Lam, Chow, and Yau 1994; Salehi, Zhang, Kurose, and Towsley 1998) with minimum image quality fluctuations. A notion similar to a retrieval block in (Keeton and Katz 1995) is also employed. In the proposed model however, the amount of data, B_r, retrieved per round per request, also referred to as a *retrieval block*, is defined by a storage pattern of I, P, and B frames of the MPEG stream. For instance, IBBPBBPIBBPBBIBBPBB can be a pattern, the size of which is almost equal to B_r but never more than B_r. The I frame is much larger than either the

P or the B frame and contributes to most of the variability in the bit rate. The size of the I frame may be made to conform to a particular value by using different quantization levels and the whole pattern may be made to conform to a size close to B_r using smoothing algorithms with minimal image quality fluctuations. Since we are not dealing at the frame level for retrieval and transfer, the variability in bit rate can be controlled with appropriate parameters during the encoding process. Given this scenario, the resulting compressed stream stored on the server is a CBR stream with minimum fluctuations in image quality. The rate of the CBR stream is the smoothed rate and not the peak or the high rate that is usually associated with rate-controlled CBR streams where the rate may have been controlled by arbitrarily dropping higher DCT coefficients or by using coarse grained quantization levels.

3.4 Network Transfer Model

Serving requests from remote clusters involves data transfer over the network. Figure 3.2 illustrates the stages involved in the remote transfer. In the proposed network model, the double-buffer at the server acts as the packet source for the *traffic regulator*. It is assumed that a (b r p M) regulator (Georgiadis, Guerin, Peris, and Rajan 1996) exists at the server. Each request is modeled as a flow passing through this regulator into the network. The data retrieved at the server for each request is packetized instantaneously and fed into the regulator. The regulator monitors the flow of traffic into the network. The regulator guarantees that in any interval $x \geq 0$, the number of bytes appearing at its output is given by

$$min(M + px, b + rx) \tag{3.1}$$

where b is the token bucket size, and r is the token accumulation rate. The parameters b and r control burstiness of the flow into the network. p and M are hardware parameters based on the network interface to the server; p is the peak rate of the network interface card, and M is the maximum packet size for the flow. Parameter

p defaults to infinity if unknown but is usually assumed to be greater than r. The regulator's effect is to define an *envelope function*, $A(x)$, for each connection (Cruz 1991). $A(x)$ is defined as follows: Let $A[t, t+x]$ be the actual amount of traffic that arrives over the connection. $A(x)$ is an envelope function for the connection if

$$A[t, t+x] \leq A(x), \;\; t, x \geq 0 \tag{3.2}$$

The (b r p M) network regulator defines an envelope function such that

$$A(x) = min(M + px, b + rx) \tag{3.3}$$

The network context for the proposed analysis is a high speed, integrated services, packet switching network. The network model may be based on the type of guaranteed service discipline implemented on it. Guaranteed service disciplines ensure that all packets associated with a particular connection never experience an end-to-end delay greater than the connection delay that is *established* at the time the connection is set up. Guaranteed network service disciplines differ, however, in how they provide support for other quality-of-service parameters such as delay jitter bound.

In the network model, *Weighted Fair Queueing* (WFQ) scheduling scheme is assumed to be implemented in the network. In (Parekh and Gallager 1993; Parekh and Gallager 1994), it is shown that WFQ provides a firm per-packet end-to-end delay bound on a per-link and per-routing path and ensures that all transmitted packets will be able to meet this bound. The WFQ delay bound is used in the proposed analysis and inverted to find reserved rates at the network in the next section. The choice of WFQ as the scheduling scheme in the proposed model is appropriate because WFQ provides the tightest end-to-end delay bound compared to other disciplines. It is also a well understood discipline and has already found its commercial implementation in routers (Keshav 1997).

Table 3.1: Model parameters for integrated admission control

B_r	Buffer size at the receiver or the retrieval block size
R_p	Playback rate per request (at the client)
R_d	Overall disk bandwidth
R_r	Reserved rate at the network per request
T_{round}	Round length
n	Maximum number of requests at the server
n_c	Number of requests currently being served
D_{max}	Maximum bounded end-to-end delay

3.5 Analytical Model

In this section, I develop an integrated analysis by combining the retrieval mechanism at the server with the rate-based transfer mechanism in the network. This integrated framework is critical to the proposed end-to-end analysis of the VoD system which is the main goal for this dissertation. A unified model is developed using the buffer size at the client to derive the round length at the server as well as the reserved rate at the network. I derive admission control conditions at the server and the network, both of which have to be satisfied for a remote request to get admitted into the system. This end-to-end admission control model is later implemented in the simulation. The model parameters are shown in Table 3.1.

At the server, a retrieval block per request is retrieved into the double buffer in each round. The size of the retrieval block per round is also the buffer size at the receiver on the client site. The retrieval block, B_r is played back at a rate R_p at the client site. The round length is determined as the playback duration of the retrieval block at the client site. The round length, T_{round}, for the server is determined by:

$$T_{round} = \frac{B_r}{R_p} \tag{3.4}$$

As illustrated in Figure 3.2, the size of the buffer in the double buffer scheme corresponds to the total data retrieved in a round length for a maximum number of

n requests. The maximum number of requests, n, serviceable at the server is derived as follows. The time to transfer the amount of data retrieved per round is bounded by the round length, that is, $\frac{nB_r}{R_d} \leq T_{round} = \frac{B_r}{R_p}$. The maximum number of requests that can be served at the server, n, is: $n \leq \left\lfloor \frac{R_d}{R_p} \right\rfloor$.

An admission control test at the server determines if the arriving request can be admitted into the system. The server is able to serve a maximum of n requests in a round. The request will be admitted into the system if the following condition holds:

$$n_c + 1 \leq n \tag{3.5}$$

For video requests which must use a network connection, the minimum reserved rate, R_r, at the network required to prevent starvation at the client site is given by:

$$R_r \geq \frac{B_r}{T_{round}} = R_p \tag{3.6}$$

The reserved bandwidth cannot be greater than the overall bandwidth of each link or hop on a routing path with J links, $R_r \leq A_j,$ for $j \in [1, J]$ where A_j is the overall bandwidth on j^{th} link. Therefore, the bounds on reserved rate at the network, R_r, are: $R_p \leq R_r \leq A_j$ for $j \in [1, J]$.

By using a (b r p M) regulator and a minimum service level of R_r on a routing path with J links, the maximum end-to-end delay bound D_{max} is given by (Cruz 1995; Georgiadis, Guerin, Peris, and Rajan 1996):

$$D_{max} = \frac{(b-M)}{R_r}\frac{(p-R_r)}{(p-r)} + \frac{(M+C_{tot})}{R_r} + D_{tot} \quad \text{if } R_r < p \tag{3.7}$$

$$D_{max} = \frac{(M+C_{tot})}{p} + D_{tot} \quad \text{otherwise} \tag{3.8}$$

where $C_{tot} = \sum_{j=1}^{J} C_j$ and $D_{tot} = \sum_{j=1}^{J} D_j$. C_j and D_j are error terms originating from the router's approximation of the perfect fluid model (Cruz 1995; Georgiadis, Guerin, Peris, and Rajan 1996). The errors refer to the delay introduced at the router: for instance, a packet arrives but just misses its turn to be scheduled at a router or when a packet has to wait for the transfer of the packet in progress at all routers along the path. The terms C_{tot} and D_{tot} represent the summation of the C_j and D_j error terms at each link, respectively.

Since R_r is almost always less than p, the peak rate of the network interface card, only the case where $R_r < p$ in Equation 3.7 needs to be considered. In the proposed model, at the (b r p M) regulator the depth of the token bucket, b, is the same as the retrieval block per request per round, B_r. The rate at the token bucket is equal to the reserved rate at the network, $r = R_r$. After substituting for b and r in Equation 3.7 and some simplification, the maximum end-to-end delay is:

$$D_{max} = \frac{B_r}{R_r} + \frac{C_{tot}}{R_r} + D_{tot} \tag{3.9}$$

The values for C_j and D_j depend on the scheduling mechanism implemented at the network. Using WFQ, the maximum end-to-end delay bound per packet for a flow with a guaranteed rate of R_r is given by (Parekh and Gallager 1993; Parekh and Gallager 1994):

$$D_{max} = \frac{B_r}{R_r} + \frac{(J-1)M}{R_r} + \sum_{j=1}^{J} \frac{M_{max}}{A_j} \tag{3.10}$$

where M_{max} is the maximum size of a packet permitted in the network (MTU).

Using (3.10), R_r is derived corresponding to an input delay bound D_{max}. This required minimum rate, $R_r \geq R_p$, is used in the admission control test before admitting a new request on the network.

$$R_r = \frac{B_r + (J-1)M}{D_{max} - \sum_{j=1}^{J}(M_{max}/A_j)} \tag{3.11}$$

Admission control tests for new flow requests are run on a link by link basis over the routing path. Each router along the routing path will administer these tests to ensure that there are enough available resources. If enough resources are available, they are reserved for the new flow using a signaling protocol, such as RSVP (Zhang, Deering, Estrin, Shenker, and Zappala 1993).

Admission control condition for setting up a new flow is as follows. Suppose that link j has currently accepted T flows $1, 2, \ldots, T$. The i^{th} flow has a reserved rate of R_{r_i} computed using (3.11). If the *new* flow requests a rate of $R_{r_{new}}$, it will be admitted only if the following relation holds:

$$\sum_{i=1}^{T} R_{r_i} + R_{r_{new}} \leq A_j \tag{3.12}$$

3.6 Extension to Multi-rate Service Model

In this section, I present an extension to the model from Section 3.5 to include heterogeneous client classes. All requests in that model required the same playback rate, R_p and the client buffer size was limited to B_r. Relaxing that assumption here, a set of client classes is assumed, each with its own playback rate and buffer size. A request originating from a client of class k, $k \in [1, K]$ will have a playback rate of R_p^k and buffer size B_r^k. It is reasonable to assume that a client with a large buffer can also support a higher playback rate. Therefore, without loss of generality we can assume that $B_r^1 \leq B_r^2 \leq B_r^3 \ldots \leq B_r^k$ and $R_p^1 \leq R_p^2 \leq R_p^3 \ldots \leq R_p^k$.

Equation 3.4 is now modified based on the minimum buffer to playback rate ratio. The round length is equal to T_{round}^{min} and defined as:

$$T_{round}^{k} = \frac{B_r^k}{R_p^k} \tag{3.13}$$

$$T_{round}^{min} = \min_{k \in [1,K]} T_{round}^{k} \tag{3.14}$$

Retrieval block sizes, $\widehat{B_r^k}$, for each client class based on the minimum round length, T_{round}^{min}, are:

$$\widehat{B_r^k} = T_{round}^{min} R_p^k \quad \text{for } k \in [1, K] \tag{3.15}$$

Since the size of the retrieval block per client class is computed using the minimum round length, overflow of the client buffer is prevented for all classes. In the following Lemma, it is shown that the retrieval block for each class k is less than the buffer size of that class.

Lemma 1 *The size of the retrieval block, $\widehat{B_r^k}$ for a k^{th} class client is always less than or equal to their buffer size, B_r^k.*

$$\widehat{B_r^k} \leq B_r^k \quad \textit{for } k \in [1, K] \tag{3.16}$$

Proof of Lemma 1:

Since T_{round}^{min} is the minimum of all T_{round}^k, we have

$$T_{round}^{min} \leq T_{round}^k \quad \text{for } k \in [1, K]$$

Multiplying with R_p^k on both sides, we have

$$R_p^k T_{round}^{min} \leq R_p^k T_{round}^k \quad \text{for } k \in [1, K]$$

From Equations 3.15 and 3.13, the LHS is equal to the retrieval block for k^{th} class whereas the RHS is equal to the buffer size of that class. Therefore we have,

$$\widehat{B_r^k} \leq B_r^k \quad \text{for } k \in [1, K]$$

QED

The size of the retrieval block in a round for i^{th} request of class k, is $\widehat{B^k_{r_i}}$. The admission control condition at the server in (3.5) is modified as:

$$\frac{\sum_{i=1}^{n_c+1} \widehat{B^k_{r_i}}}{R_d} \leq T^{min}_{round} \quad \text{for } k \in [1, K] \tag{3.17}$$

The bounds on minimum reserved rate at the network for a request from class k, R^k_r, are: $R^k_p \leq R^k_r \leq A_j$ for $j \in [1, J]$. This required minimum rate, $R^k_r \geq R^k_p$, is now computed as:

$$R^k_r = \frac{\widehat{B^k_r} + (J-1)M}{D_{max} - \sum_{j=1}^{J} M_{max}/A_j} \quad \text{for } k \in [1, K] \tag{3.18}$$

The admission control condition at the network in (3.12) is now modified as follows:

$$\sum_{i=1}^{T} R^k_{r_i} + R^k_{r_{new}} \leq A_j \quad \text{for } k \in [1, K] \tag{3.19}$$

where $R^k_{r_i}$ refers to the minimum rate for i^{th} request flow of class k.

3.7 Summary

In this chapter, I developed a model for analyzing a VoD system as an end-to-end system. The end-to-end admission control concept in a distributed VoD system is presented by deriving admission control conditions at the server and the network. The significance of the analysis presented in this chapter:

- Shows how we can accomplish an end-to-end analysis of a distributed VoD system by developing an integrated framework of storage and network subsystems with client buffer size as the input.

- Develops a general purpose model that may include any disk scheduling scheme on the server side and any network scheduling discipline on the network side.

- Shows how WFQ delay bounds and minimum reserved rate are incorporated into an integrated admission control model for the distributed VoD system.

- Shows that by deriving a lower bound for reserved rate at the network, $R_r \geq R_p$, and by enforcing admission control, we are able to maintain strict time constraints for periodic retrieval and transfer of data and prevent starvation at the client.

- Proves that client buffer overflow is prevented by defining the round length as the minimum buffer to playback rate ratio among heterogeneous client classes and by determining retrieval blocks based on that minimum round length.

The analytical model presented in this chapter for single rate and multi-rate service forms the basis for a simulation of the overall VoD system presented in Chapter 5.

CHAPTER 4

THRESHOLD-BASED ADMISSION CONTROL IN VOD SYSTEMS

In the previous chapter, I developed a model for integrated admission control on multiple resources along a request path in a distributed VoD system. The admission control tests depended on capacity constraints alone and all video requests were admitted into the system as long as there were enough resources to serve them. Consider inducing a preference order among requests. For instance, if we have two classes of video requests, popular and less popular, requests for popular videos may be preferred over less popular. In that case, we may want to conserve resources by rejecting requests for less popular requests in anticipation of popular requests. We can accomplish that by limiting access to video requests for less popular videos by enforcing thresholds in the VoD admission control process. In this chapter, I define threshold-based admission control policies for VoD systems and develop an analytical model for evaluating the performance of a single VoD resource such as a video server using threshold-based analysis.

This chapter is organized as follows. In Section 4.1, threshold-based admission control concept with examples is discussed. In Section 4.2, I develop an analytical model for computing blocking probabilities efficiently under threshold restrictions using a recursive method. In Sections 4.3 and 4.4, I present numerical results and summarize the significance of threshold-based model.

4.1 Threshold-based Admission Control at a Video Server

An admission control *policy* determines whether an arriving request for a certain class of video should be admitted given the current state of the VoD server. Without loss of generality, suppose there are two classes of requests and the vector (n_1, n_2) describes the state of the system in terms of the number of class 1 requests, n_1 and the number of class 2 requests, n_2. The state of the VoD server is describable as a state space constrained by capacity. A capacity constraint is defined by $\sum_{k=1}^{2} n_k \leq C$ where C is the maximum number of simultaneous requests that a video server can handle. The state space can further be constrained by other restrictions, such as when the admission control policy favors one class more than the other by enforcing a threshold on one of the classes.

Three types of class-based admission control are defined: *complete partitioning*, *complete sharing*, and *threshold type*. A complete partitioning policy means that the server capacity is completely partitioned and every class of request has access only to a dedicated set of resources. A complete sharing policy is one where requests of all classes are always accepted if there is available capacity. A threshold type policy allows requests of classes up to the thresholds assigned to those classes. A threshold *class-2* policy limits class 2 requests up to a specified threshold whereas always accepting class 1 requests if there is available capacity. Threshold type policies are useful because it may not be desirable to share all of the server capacity equally among classes. For instance, requests for certain classes of videos might bring in more revenue.

Given a state space of the system based on the admission control policy, we can identify those states where an arrival will be blocked either because of the capacity constraint or the threshold. Blocking probability is computed using this information. For example, Figure 4.1 illustrates the situation for two classes of requests and maximum capacity of five requests at the video server. The policy illustrated in the figure is to allocate no more than 60% of the resources at the video server to class 2

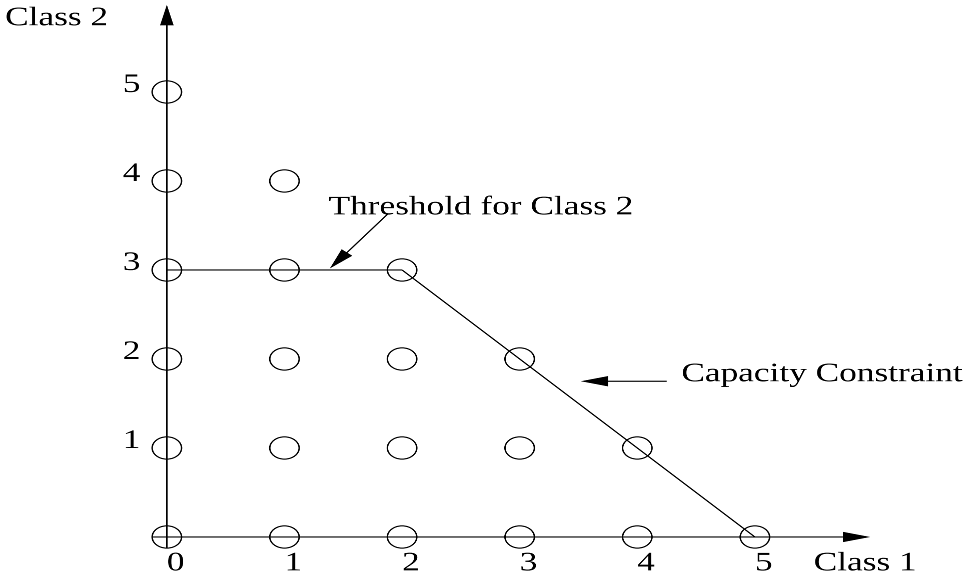

Figure 4.1: State space for threshold class-2 policy

requests. Only a maximum of three class 2 requests are allowed in the system. There are no restrictions on class 1 requests other than the capacity of the system. Because of capacity and policy restrictions, arrivals of the pertinent classes are rejected when the VoD system is in one of the blocking states.

The state space, Ω, is defined by

$\Omega = \{(n_1, n_2) : 0 \leq n_1 \leq 5 \text{ and } 0 \leq n_2 \leq 3 \text{ and } \sum_{k=1}^{2} n_k \leq 5\}$

Blocking states for class 1 requests, $\Omega_1 = \{(2,3)(3,2)(4,1)(5,0)\}$ and blocking states for class 2 requests, $\Omega_2 = \{(0,3)(1,3)(2,3)(3,2)(4,1)(5,0)\}$.

4.2 Analytical Model

In this section, I develop an analytical model for evaluating the performance of a single video server under threshold-based admission control. Two classes of videos, *popular* and *unpopular* are assumed to be stored on the video server. Each class will consist of many videos of different duration. The duration of the video playback defines the service time for a video request. The service time is assumed to be exponentially distributed. This assumption is reasonable because it allows us to have a large array

Table 4.1: Model parameters for threshold analysis

λ_1	Arrival rate of request for popular movies on video server (class 1)
λ_2	Arrival rate of request for unpopular movies on video server (class 2)
$1/\mu_1$	Service time for popular movies
$1/\mu_2$	Service time for unpopular movies
C	Maximum number of requests that the video server can handle simultaneously
l_k	Threshold for class k, $k = 1, 2$
P_{b_1}	Blocking probability for class 1 requests
P_{b_2}	Blocking probability for class 2 requests
$P(n_1, n_2)$	Probability of n_1 class-1 requests and n_2 class-2 requests, $\sum_{k=1}^{2} n_k \leq C$
Ω	Set of admissible states under a given policy for a system with threshold on both classes, $\Omega = \{(n_1, n_2) : 0 \leq n_1 \leq l_1 \text{ and } 0 \leq n_2 \leq l_2 \text{ and } \sum_{k=1}^{2} n_k \leq C\}$ for a threshold class-2 system, $\Omega = \{(n_1, n_2) : 0 \leq n_1 \leq C \text{ and } 0 \leq n_2 \leq l_2 \text{ and } \sum_{k=1}^{2} n_k \leq C\}$
Ω_1	Set of blocking states for class 1 requests
Ω_2	Set of blocking states for class 2 requests

of different length videos within a class. The steady state distribution provided in this section, however, holds for any arbitrary service time distributions (Kaufman 1981). The arrival rates of requests per class are based on Poisson process. The assumption of a Poisson arrival process is necessitated by the lack of actual workload characteristics for a VoD system. All incoming requests are blocked if the video server capacity is not available or if there are requests up to the threshold for that class. The admission control policy used is threshold type. Blocking probability for class 1 (popular) and class 2 (unpopular) requests may now be computed using the approach below. Model parameters and definitions are shown in Table 4.1.

4.2.1 Blocking Probability using State Space Enumeration

As shown in (Aein 1978) and (Kaufman 1981), the equilibrium probability, $P(n_1, n_2)$, of finding the server in the state (n_1, n_2) has the following product form:

$$P(n_1, n_2) = \frac{1}{G} \prod_{k=1}^{2} \rho_k^{n_k}/n_k!, \quad \text{all } (n_1, n_2) \in \Omega \tag{4.1}$$

where

$$G = \sum_{(n_1,n_2)\in\Omega} \prod_{k=1}^{2} \rho_k^{n_k}/n_k!$$

and

$$\rho_k = \lambda_k/\mu_k$$

The blocking probability, P_{b_k}, is given by

$$P_{b_k} = \sum_{(n_1,n_2)\in\Omega_k} P(n_1, n_2) = G(\Omega_k)/G \tag{4.2}$$

where
$\Omega_k = \{(n_1, n_2)|(n_1, n_2) \in \Omega,\ (n_1, n_2) + \overline{e_k} \notin \Omega\}$ where $\overline{e_k}$ is a vector equal to either $(0, 1)$ or $(1, 0)$, where the 1 is in the k^{th} position. Ω_1 and Ω_2 are the set of blocking states for class 1 and class 2 requests respectively.

The state enumeration technique for computing blocking probability presented in this section is computationally very expensive. The need for computational efficiency arises because the cardinality of Ω grows roughly as C^k which makes exhaustive enumeration techniques computationally too expensive. Hardware limitations rule out computing G for even moderate sized problems. Some previous studies in finding computationally tractable solutions for computing G were done in the context of multi-rate circuit switched networks (Kaufman 1981; Roberts 1981) and multiprogramming (Buzen 1973). For instance, Kaufman (Kaufman 1981) and Roberts(Roberts 1981) independently developed recursive relations to compute the G factor. All these recursive solutions including Buzen's refer to the complete sharing policies where requests of all classes are always accepted if there is available capacity (see Chapter 2). In contrast, under threshold-based admission control, requests of certain classes are not admitted into the system beyond a threshold even if there is available capacity. In the next section, I provide a recursive method for computing blocking probabilities under threshold-based admission control.

4.2.2 Blocking Probability using Recursion

In this section, I develop an efficient way of computing blocking probabilities for a single resource, multi-server, 2 class threshold type system based on recursion.

Kaufman/Roberts recursion (Kaufman 1981; Roberts 1981) was generalized to compute blocking probability for a multi-rate tree network in (Tsang and Ross 1990). In a multi-rate tree network, both access link and common link capacities have to be available for call establishment. The proposed threshold-based admission control situation can be analyzed based on the same logic. For instance, a video request with threshold restriction can only be admitted if there is capacity and if the number of requests currently being served is less than the threshold. Therefore, threshold type policies introduce additional blocking states that should be included in the computation of blocking probabilities. Also, the state space Ω is now truncated according to the threshold values. An additional function has to be included in the Kaufman/Roberts recursion to reflect additional blocking from the threshold blocking states.

The basic idea behind the recursion is summing $P(n_1, n_2)$ over Ω recursively along disjoint parallel planes on the diagonal, as shown in Figure 4.2. From this summing technique, we get a distribution in terms of the total number of servers being used. Denote the total number of servers used as $I = \sum_{k=1}^{2} n_k$, and define a state space,

$$\Theta(i) = \{(n_1, n_2) \in \Omega : \sum_{k=1}^{2} n_k = i\}$$

The distribution q_i, indicates the number of servers being used.

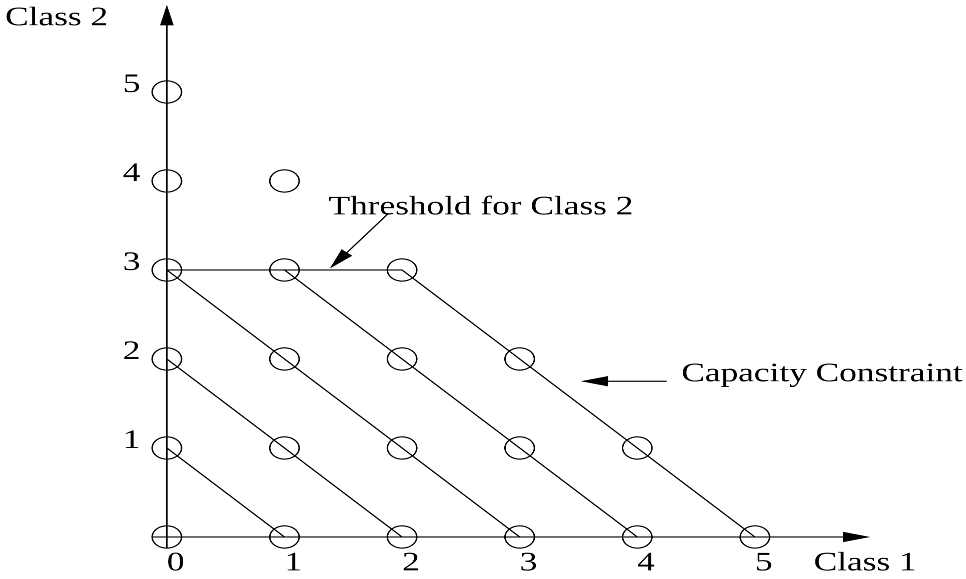

Figure 4.2: Planar states for distribution q_i

$$q_i = P(I = i) = \sum_{(n_1,n_2)\in\Theta(i)} P(n_1, n_2), \quad i = 0, 1, \ldots, C \tag{4.3}$$

Define,

$$G(i) = \begin{cases} \sum_{(n_1,n_2)\in\Theta(i)} \prod_{k=1}^{2} \rho_k^{n_k}/n_k! & i = 0, 1, \ldots, C \\ 0 & \text{otherwise} \end{cases}$$

Multiplying (4.3) with G and substituting for $P(n_1, n_2)$ from (4.1) we get the expression above for $G(i)$,

$$G(i) = q_i G \tag{4.4}$$

To handle threshold type policies, we define a threshold value which is less than C to limit the number of requests for those classes. Define $l_k < C$ as the threshold for class k requests.

Define for $k = 1, 2$,

$$B_k(i) = \begin{cases} P(I = i, n_k = l_k)G & \text{if } l_k \le i \le C-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

$B_k(i)$ is the unnormalized probability of being in one of the additional blocking states due to the threshold. If no threshold is defined for a particular class, $B_k(i)$ for that class is 0.

Blocking probability,

$$P_{b_k} = P(I = C) + P(I \le C-1, n_k = l_k)$$

$$P_{b_k} = \frac{G(C) + \sum_{i=l_k}^{C-1} B_k(i)}{\sum_{i=0}^{C} G(i)} \tag{4.6}$$

To compute blocking probability, we need only calculate the G factor and $B_k(i)$, $i = l_k, \ldots, C-1$, for $k = 1, 2$. We use the following recursion to accomplish that.

Theorem 1

$$G(i) = \frac{1}{i}\sum_{k=1}^{2} \rho_k[G(i-1) - B_k(i-1)], \quad i = 1, \ldots, C \text{ and } G(0) = 1 \tag{4.7}$$

The proof of Theorem 1 is contained in the Appendix.

Computing $B_k(i)$ for a threshold class-2 system is presented next. A threshold class-2 system allows class 2 requests up to a threshold and always accepts class 1 requests if capacity is available. In the proposed model, it is reasonable to limit class 2 (unpopular) requests and allow more of class 1 (popular) requests to be accepted into the system. Therefore, a threshold restriction is not assumed for class 1.

A function, $r(i)$ is defined using the same recursion from Theorem 1 but on class 1 dimension, to compute $B_2(i)$. The function $r(i)$ represents the resource usage from class 1 in excess of the class 2 threshold. Notice that the resource usage for class 2 is at its threshold, l_2.

$$r(i) = \frac{1}{i}\rho_1[r(i-1) - B_1(i-1)], \quad i = 0, \dots, C-1-l_2 \text{ and } r(0) = 1 \tag{4.8}$$

Since we do not have a threshold restriction for class 1, by definition we have, $B_1(\cdot) = 0$. Equation 4.8 becomes

$$r(i) = \frac{1}{i}\rho_1[r(i-1)], \quad i = 0, \dots, C-1-l_2 \text{ and } r(0) = 1$$

We use the function $r(i)$ to compute $B_2(i)$ as follows:

$$B_2(i) = (\rho_2^{l_2}/l_2!)\, r(i-l_2), \quad i = l_2, \dots, C-1 \tag{4.9}$$

Stripping off the recursion in (4.9) we have

$$B_2(i) = (\rho_2^{l_2}/l_2!)(\rho_1^{(i-l_2)}/(i-l_2)!)$$

which can be proved from equilibrium conditions, the product form expression in (4.1), and the definition of $B_2(i)$.

Remarks

In the previous section, the analysis was limited to a two class model. However, the model can easily be extended to K classes. The recursion in Theorem 1 is straight forward enough in that the summation will include K classes. The analysis gets complicated in the computation of $B_k(i)$s. The presence of interference between threshold planes requires further recursion in $r(i)$ when computing a certain $B_j(i), j \neq$

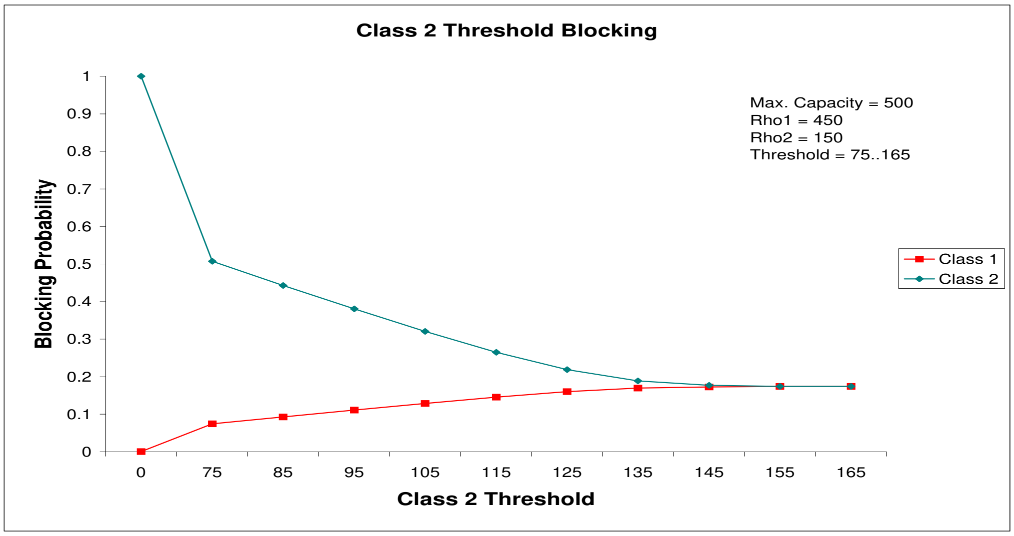

Figure 4.3: Class 2 threshold blocking

k. If there is no interference, the computation of $r(i)$ is as shown below because all $B_k(\cdot)$s will be zero.

$$r(i) = \frac{1}{i} \sum_{k=1,\ k \neq j}^{K} \rho_k [r(i-1)], \quad i = 1, \ldots, C - 1 - l_j \tag{4.10}$$

4.3 Numerical Results

By using the results of the previous section a number of different types of admission policies may be analyzed. In this section, I present numerical results for a VoD server with two classes of requests, class 1 (popular) and class 2 (unpopular), and an overall capacity of $C = 500$. No threshold restrictions are assumed on class 1 except the capacity of the server; but there is a threshold restriction on class 2. The recursion $r(i) = \frac{1}{i}\rho_1[r(i-1)]$ is used to compute all $r(i)$s and (4.9) is used to compute $B_2(i)$. Once all $B_2(i)$ are determined, Theorem(1) is used to compute $G(i)$.

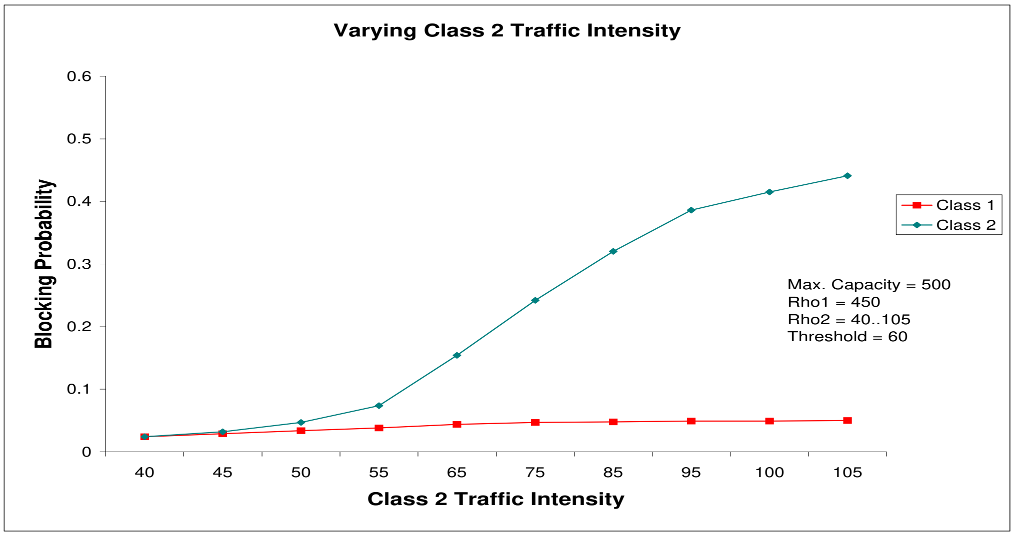

Figure 4.4: Varying Class 2 traffic intensity

Blocking probability for both classes is calculated as follows:

$$P_{b_1} = \frac{G(C)}{\sum_{i=0}^{C} G(i)} \tag{4.11}$$

$$P_{b_2} = \frac{G(C) + \sum_{i=l_2}^{C-1} B_2(i)}{\sum_{i=0}^{C} G(i)} \tag{4.12}$$

A computer program was written to compute blocking probabilities using (4.11) and (4.12). A standard 15 digit double precision using GNU C compiler on a Sparc workstation was used to compute the results.

The results of the first numerical analysis shown in Figure 4.3, shows the effect of varying the admission threshold for class 2 while ρ_1 and ρ_2 are held constant. In this analysis the capacity C is set to 500 (simultaneous requests), the threshold for class 2 ranges from 0 to 165, ρ_1 is set to 450 and ρ_2 is set to 150. As the threshold for class 2 is increased allowing more class 2 requests into the system, the blocking probability

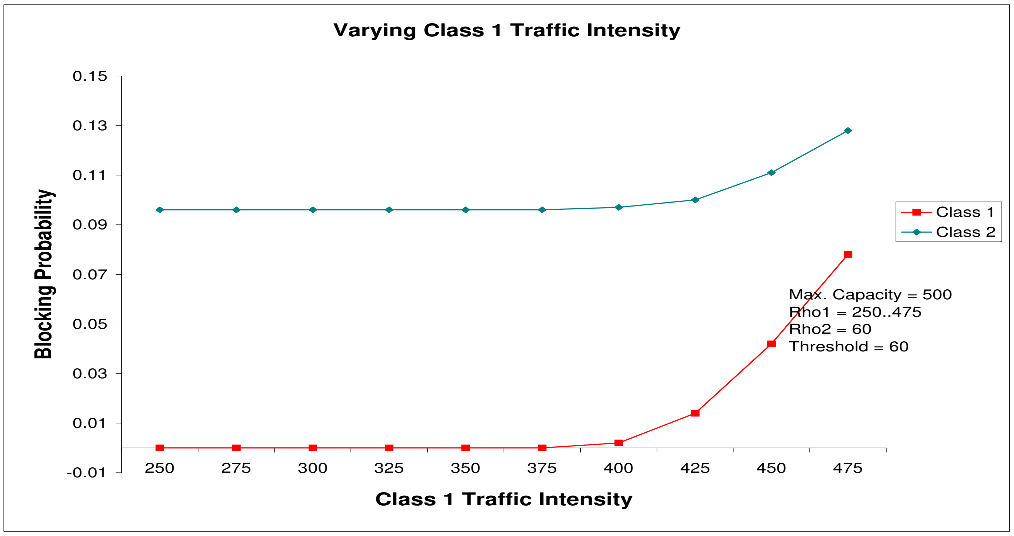

Figure 4.5: Varying Class 1 traffic intensity

for class 1 increases as a result of increased sharing of resources. Eventually, we have a situation similar to complete sharing policy, where each class has the same blocking probability.

Figure 4.4 shows the effect of increasing the arrival rate of class 2 while maintaining a fixed threshold. The blocking probability rapidly increases for class 2, while the blocking performance for class 1 barely varies.

Figure 4.5 shows the effect of increasing the arrival rate for class 1, while maintaining a fixed threshold and arrival rate for class 2. Blocking for both classes remains constant until the value of ρ_1 approaches capacity, at which point, blocking for both classes increases rapidly, as expected.

4.4 Summary

The significance of threshold-based analysis for VoD systems is enormous. In all of the previous research on admission control, there was no way of differentiating between the merits of classes of arriving requests. All classes of requests would have been admitted into the system if there were enough resources to provide guaranteed services. This type of admission control is called the complete sharing admission control policy. With the threshold-based admission control analyzed in this chapter, we now have a way of denying service to a video request which we deem to be of less value while conserving resources in anticipation of a more lucrative request. This mode of operation gives us better control for realizing a desirable level of performance from the VoD system.

I also showed that it is possible to analytically compute the blocking probabilities under threshold-based admission control. In place of the state enumeration method which is computationally expensive for large scale problems, I developed an efficient recursive method for computing blocking probabilities. Numerical results illustrate the behavior of a VoD resource under class-based threshold restriction.

CHAPTER 5

SIMULATION AND RESULTS

In this chapter, details of the simulation of a distributed VoD System are presented. I implement the integrated admission control model from Chapter 3 in the simulation. Global performance metrics are developed and used to evaluate the performance of the VoD system. Request handling policies and threshold-based admission control techniques are designed and evaluated to check how efficiently they use the multiple resources in the VoD system. I analyze the performance of the VoD system under two service models: single rate and multi-rate.

The rest of the chapter is organized as follows. In Section 5.1, I discuss the difficulties in developing an analytical model for a distributed VoD architecture and state why simulation is a better tool for analyzing the overall performance of the VoD system. In Section 5.2, I present VoD architectural details implemented in the simulation. In Section 5.3, metrics, policies, and design considerations are discussed. In Sections 5.4 and 5.5, I present the methodology and the simulation data. Results and summary of the simulation for single rate service are presented in Sections 5.6 and 5.7. In Sections 5.8 and 5.9, I present results and summary for the multi-rate service.

5.1 Difficulties in Analytical Modeling

We saw from Chapter 4 that analyzing the performance of a single resource can pose certain limitations such as the scale of a resource in VoD system. Analyzing admission control techniques such as threshold-based control on several resources for multiple classes of requests within the VoD system using the analytical model from Chapter 4 poses some challenge as well. The following major factors complicate the development of an analytical model of the distributed VoD architecture:

- Multiple resources on multiple hierarchies and a large video collection.
- Large number of videos and request classes, each differing in their relative popularity and significance.
- Request blocking and not queueing, with redirection of blocked requests to other resources permitted.
- Multiple classes of requests with one class being transformed into another for service after being blocked.

Reflecting all these features in a blocked queueing system of the Erlang Loss type (Kleinrock 1975) is difficult and may even be impossible. On the other hand, developing a discrete event simulation to reflect all aspects of the complex VoD architecture is quite possible as demonstrated in this chapter.

5.2 VoD Architecture Revisited

In the simulation, the hierarchical VoD architecture from Chapter 3 consisting of one local cluster and several remote clusters is used. Each cluster consists of many video servers. The local cluster is dedicated to one user population. This user population is labeled as the *reference* user population so that we can refer to it in our future discussions. The stream handling capacity on the local cluster is limited and requests will be blocked if there are no resources available to serve them. The remote clusters may be shared by other user populations. The remote clusters act as blocking resources if the available stream handling capacity on them with respect to the reference user population is less than the network path leading up to them.

The local cluster serves users over a local distribution network which is not a blocking resource. The remote clusters serve requests over high speed packet switching networks where there is resource contention. The routing paths leading to the remote clusters have limited bandwidth and block requests when there are not sufficient resources to serve them. Assume that the bandwidth on the routing paths is dedicated to the reference user population. Also assume that the routing paths are fixed for all requests directed to the remote clusters. These are not unreasonable assumptions considering that resources have to be reserved prior to connection establishment and fixing the path simplifies the resource reservation process. The network scheduling discipline along the routing path is based on WFQ.

The reference user population consists of a client base, each client capable of receiving compressed video information and decompressing it for display. Clients are connected to either the local site or the remote sites for service based on some request handling policy. In the simulation, requests are generated from the reference user population.

A known playback rate based on MPEG compression scheme characterizes each request stream. Servers in the cluster and the network are assumed to be capable of providing simultaneous service to a set number of requests at that playback rate. Request arrival rate from the reference user population is based on Poisson process with exponential interarrival times. The service time for a video is the duration of the video playback and is a constant. The video collection as a whole contains videos of different duration, short and long, the mean of which can be represented by an exponential random variable as shown in Chapter 4. In the simulation, each individual video in the video collection has a service time equal to its playback duration. Video popularity is used as the basis for determining the demand for a video and its arrival rate of requests. Multiple request classes are considered, each class with its own relative significance based on resource usage or the popularity of the video.

The videos in the video collection are replicated and distributed in the local cluster according to the replication and placement algorithm discussed in Section 5.2.1. The individual servers in the local cluster contain a subset of the video collection. Depending on storage limitations, partial or complete video collection is available on the local cluster as a whole. One important issue arising out of having clusters of servers is mapping individual videos to servers within a cluster referred to as the video *placement or allocation* problem. In (Wang, Liu, Du, and Hsieh 1997), the allocation problem is shown equivalent to a 2D vector packing problem and therefore, is found to be NP-hard. The objective of the allocation problem is to minimize the load imbalance. The videos must be assigned to the individual servers such that the storage constraints are met and the demand for the videos assigned to a server closely matches the stream handling capacity of the server. In the next section, I present two greedy algorithms for replication and placement which provide solutions to the load balancing problem.

5.2.1 Video Distribution and Load Balancing

In this section, I describe techniques for video distribution and balancing load at the local cluster in the hierarchical VoD architecture. Remote clusters are archival in nature and will store all videos in the video collection. The load balancing issue does not arise at the remote cluster because of its archival nature and absence of any storage constraints.

A local cluster consists of several video servers. The number of videos to be placed on the servers is assumed to be large. Typically, no single video server has either the storage or the request stream handling capacity to store and deliver all videos. The varying popularity of the videos results in different access patterns at the servers. Concentration of only popular or unpopular videos on a single server must be avoided to prevent load imbalance. The imbalance problem is remedied using replication and proper video placement schemes. I propose two greedy algorithms for replication and placement next.

Replication Algorithm

The replication algorithm computes the number of copies for videos by distributing the overall storage capacity of the cluster in proportion to their demand. Therefore, high demand videos will get a larger share of the storage capacity which translates into a large number of replicated copies. The VoD system offers a video collection of V videos. Let λ_i be the arrival rate and a_i be the service time for the i^{th} video. Let d_i be the size of the i^{th} video in gigabytes (GBytes). Let total storage available at the cluster be $S = \sum_{j=1}^{N} S_j$ where S_j is the storage capacity of the j^{th} server in GBytes and N is the number of servers in the cluster. The total traffic intensity from all videos is $\rho = \sum_{i=1}^{V} \rho_i$ where $\rho_i = \lambda_i a_i$. The number of copies of i^{th} video, K_i, is equal to :

$$K_i = \text{round}((S/d_i)(\rho_i/\rho))$$

The *round* function rounds K_i off to the nearest integer. Neither the ceiling nor the floor function gives the appropriate number of copies. Using a ceiling function, we will be overestimating the popularity of some videos. With a floor function, we will be underestimating the popularity of some other videos. The lower bound on K_i is 0 and a video with $K_i = 0$ is stored only on the remote cluster. The upper bound on K_i is N, the number of servers in the local cluster. The reason for this is that a single server can store only one copy of the video. Storing multiple copies of a video on the same server will not improve retrieval efficiency because of the double buffer scheme at the server and therefore, I do not consider that option. On the other hand, storing multiple copies of a video on separate servers will increase the throughput capacity available to that video.

Video Placement Algorithm

The idea behind the placement algorithm is to match high traffic intensity videos with high stream handling capacity servers. We achieve this by working with two ordered lists; one for the videos, from high to low traffic intensity and the other for the servers, from high to low stream handling capacity. The videos are then matched to servers in a round-robin fashion, subject to storage constraint. Videos that have $K_i \geq 1$ are considered for placement on the local cluster. The videos with $K_i = 0$ are not stored on the local cluster and will be served directly from the remote clusters. In the placement algorithm each replicated copy is treated separately and therefore, we have $\widehat{V}$ videos, $\widehat{V} \geq V$, accounting for each replicated copy as a separate video. For example, suppose we have two distinct videos, X, Y and $V = 2$. Video X is replicated three times and video Y two times. Treating each replicated copy as a separate video, we now have five videos, X1, X2, X3, Y1, Y2 and $\widehat{V} = 5$. Each replicated copy will share the traffic intensity of the original video equally. Therefore, the traffic intensity ϕ_k for video k, $k \in \{1 \ldots \widehat{V}\}$, is given by $\phi_k = \rho_i / K_i$ for $K_i \geq 1$ and $i \in \{1 \ldots V\}$. Each video from an ordered list of videos, $1 \ldots \widehat{V}$, is mapped to a server from an ordered list of servers, $1 \ldots N$, subject to storage constraints.

The placement algorithm is intended to work well with servers of different stream handling capacities. But with the proposed replication strategy, the high demand for the more popular videos is now more or less flattened because each replicated copy shares the high demand equally. Therefore, we expect the placement algorithm to work well even when all servers are identical in their stream handling capacities.

5.3 Simulation Modeling

The primary objective of this simulation is to analyze the end-to-end performance of the distributed VoD system. Developing appropriate metrics and request handling policies is critical to an accurate analysis. I address the following issues before proceeding with the simulation.

1. What are the evaluation metrics?

2. What are the request handling and admission control policies?

3. Are there special design considerations?

The main objectives for this simulation are:

- to quantify the advantages and limitations of various policies under varying load conditions using the proposed evaluation metrics.
- to study the utilization of the underlying resource infrastructure under each of these policies shedding light on their performance.
- to analyze the effect of policies and design considerations on the performance of request classes.

5.3.1 Global Performance Metrics

Request handling at the VoD system and metrics used in the performance analysis are described in this section. The new request is routed to either the local cluster or the remote clusters. The decision to serve from local or remote clusters is based on policy restrictions and availability constraints. A request can be served from the local cluster if the video is available at the cluster and if enough capacity is available at the server where the video resides. For a request to be served from the remote cluster, three requirements must be met: (i) capacity at the remote server (ii) guaranteed rate at the network and (iii) bounded delay service at the network. Since the local cluster is limited in storage space, the number of videos to be stored at the local cluster is limited. As a result, a large number of videos that are not frequently requested may not be stored on the local cluster. The demand for such videos is met from the remote clusters. Requests for videos not available at the local cluster are sent to remote clusters directly.

An end-to-end admission control test will determine if the new request can be admitted into the system based on resource availability on all subsystems involved. A request that gets rejected by the admission control test is said to be *blocked*. The number of blocked requests forms the basis for developing global performance metrics: *overall blocking probability* and *overall blocking rate*.

The overall request arrival rate at the VoD system is λ. Suppose that the probability of a request for i^{th} video is p_i. The arrival rate for i^{th} video is $\lambda_i = \lambda p_i$. The blocking probability, b_i for the i^{th} video is defined as the ratio of the total number of blocked requests to the total number of arrivals at the VoD system. The total number of blocked requests is the difference between the total number of arrivals and the total number of requests served at all resources in the VoD system. Overall blocking probability, P_{vod} is the weighted average over all videos of the blocking probability per i^{th} video,

$$P_{vod} = \sum_{i}^{V} p_i b_i \tag{5.1}$$

The overall blocking rate, BR_{vod} is used as a measure of the performance of the VoD system as a whole. It indicates the number of arriving requests at the VoD system that will get blocked per time unit. The overall blocking rate is defined by:

$$BR_{vod} = \sum_{i}^{V} \lambda_i b_i \tag{5.2}$$

In addition to the overall blocking metrics, class-based blocking results are useful when analyzing class-based threshold effects. The definition of a class may be based on the characteristic of a video such as the popularity or the characteristic of a request such as its playback rate. A video-based definition includes dividing the video collection into classes based on their popularity index, each class containing a subset of the video collection. Request classes may also be defined based on their playback rates, each playback rate signifying a class. Class-based blocking for various threshold-based admission control policies are measured.

Blocking statistics at individual resources may be collected to evaluate the performance of a certain policy. Analyzing blocking activity at resources will provide insight into how each request handling and admission control policy affects performance of the overall VoD system. Resource-based blocking results are especially useful in analyzing the performance of policies based on load sharing.

5.3.2 Proposed Request Handling and Admission Control Policies

Request handling policies are defined to determine how we try to establish connection for the new request in the hierarchical VoD architecture. The policies are designed based on three criteria:

- whether remote service is available;
- whether redirection to other clusters for blocked requests is permitted;
- whether the incoming traffic is first split between local and remote clusters.

The basic idea behind the split policies is load sharing and it works well in a dedicated resource environment. Implementing a split policy with no redirection for blocked requests is the simplest, but efficient splits may be difficult to determine especially in shared resource environments. A policy based on complete or some redirection has higher implementation overhead because of the multiple checks performed per blocked request.

During admission control, in addition to capacity restriction, threshold-based restrictions may be imposed on some videos. Threshold-based admission control policies are known to result in better performance than *complete sharing* policies (see Chapter 4). Threshold policies restrict admission at the server so that a request for a video over its threshold limit will not be admitted to the system even if the capacity is available. Usually, threshold restrictions based on capacity are placed on video classes with differing popularity or request classes with different playback rates or both. A threshold of 0.6 means that only 60% of the stream handling capacity of a resource is available to that class and hence, to all video requests in that class. Threshold-based admission control may be enforced in combination with other request handling policies.

5.3.3 Design Considerations

The way we use the individual resources in the VoD system is dependent on the size of the video collection, the storage limitations on the local cluster, or the available stream handling capacity at the remote cluster in a shared environment. The considerations are whether the complete video collection be made available at the local cluster in which case enough storage has to be provided; whether only a partial list of videos are stored on the local cluster due to storage limitations; whether the remote cluster is a blocking or non-blocking resource with respect to the reference user population. Based on these design considerations, four conditions are identified and scenarios defined which will be analyzed later in the experiments. The goal is to find policies that work best under each of these scenarios (see Figure 5.1). The conditions are:

1. Replicated Video Collection: All videos from the video collection are available on the local cluster.

2. Distributed Video Collection: Only a partial set of videos from the video collection is available on the local cluster. The requests for the set of videos not available on the local cluster are served by the remote clusters directly.

3. Non-Blocking Remote Clusters: A dedicated capacity at least as large as the network capacity leading to the remote cluster from the reference user population is provided at the remote cluster. Therefore, the remote cluster acts as a non-blocking resource for the reference user population.

4. Shared Remote Clusters: Remote clusters are shared by other user populations. A dedicated portion of the remote cluster capacity may be reserved for each user population. In a dynamically shared environment, the remote cluster may be blocking or non-blocking depending on the available capacity with respect to an arriving request from a user population.

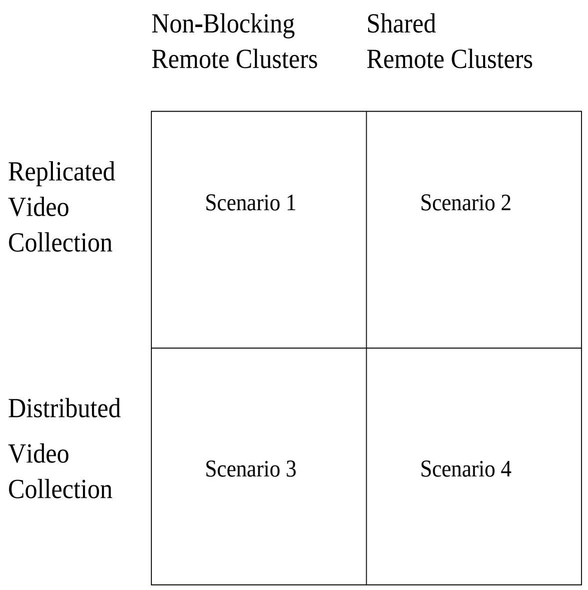

Figure 5.1: Design considerations: what policies work best for these scenarios?

5.4 Simulation Methodology

The proposed methodology consists of collecting blocking statistics at the VoD system using discrete-event simulation. Blocking based statistics give us a good insight into the performance of the VoD system under different operating conditions. Global performance metrics based on blocking performance of the VoD system were presented in Section 5.3.1. Each simulation run is conducted for a large period of time. The number of events collected over that length of time is sufficiently large. This ensures that the system has stabilized and the results obtained are valid. In all experiments presented later in the chapter, the simulation period is kept at 10,000 hours. In fact, the results of tests with periods higher than 10,000 hours match down to the fourth decimal of the test results with 10,000 hours. A confidence interval analysis at 95% level is used to verify the accuracy of simulation results.

Since we are not simulating a VoD system that is already in operation, we must validate the proposed VoD simulation model by carefully reviewing the assumptions made. I have justified all assumptions regarding VoD architecture and its operation including the assumption of Poisson arrival process. The Poisson process is a good approximation when actual workload for a VoD system is not available. In addition, constructing a high level discrete-event simulation allows us to extend the analysis by generating the incoming request traffic differently as and when the actual workload becomes available.

5.4.1 Analytical and Simulation Results for Single Video Server

Even though it is difficult to develop an analytical model for the distributed architecture of the VoD system, it is possible to represent a single resource in that architecture in an analytical framework and compute blocking performance (see Chapter 4). To compare and verify simulation results of a single video server performance, we use the analytical model from Chapter 4 without using threshold restrictions. The results of that comparison are shown in Table 5.1.

In the simulation, the blocking performance of the video server is found by collecting blocking statistics. Analytical results are found as explained in Chapter 4 by using both enumeration and recursive methods. The video server has a capacity of 100 requests. The first experiment used an arrival rate of 117 requests per hour; 17 requests per hour from class 1 and 100 requests per hour for class 2. The second experiment used an arrival rate of 125 requests per hour; 100 requests per hour for class and 25 requests per hour for class 2. The service times are kept constant at one hour. Notice that the analytical model from Chapter 4 applies to any general service distribution. There are no threshold restrictions on either class. In the simulation, the number of arrivals, the number of requests served, and the number of blocked requests are collected. Blocking probability is defined as the ratio of the number of blocked requests to the number of arrivals. As shown in Table 5.1, the blocking probability matches that from the analytical model verifying the simulation methodology.

Table 5.1: Comparing analytical and simulation results

Experiment	Enumeration	Recursion	Simulation
1	0.179	0.179	0.177
2	0.225	0.225	0.223

5.5 Simulation Setup

In this section, I provide details of the simulation data. The simulation was conducted using the CSIM simulation package (Schwetman 1998).

The simulated VoD architecture consists of three levels: *local cluster*, *remote cluster1* via *network1*, and *remote cluster2* via *network2* over a set number of hops. The routing paths, network1 and network2, leading to remote cluster1 and remote cluster2 respectively are fixed. Requests are generated by a single user population associated with the local cluster which is labeled as the reference user population. Requests are served by one of three resources: local cluster, remote cluster1, or remote cluster2. The network capacity on the routing paths is dedicated to the reference user population and not shared by other user populations. The remote clusters are however, shared by other user populations.

Each local cluster server is modeled after a 500 GBytes RAID-style server that can simultaneously serve 150 request streams of 8 megabits per second (Mbps). The remote clusters are archival in nature and store all videos. They also provide direct real-time video delivery service to arriving requests. The network characteristics are based on high speed, integrated services, packet switching network and network related data are shown in Table 5.2 along with other details. The reserved rate at the network corresponding to the input delay is calculated using Equation 3.11. The reserved rate is at least as large as the playback rate.

The video collection consists of 150 videos and the demand for each video depends on the popularity of the video. Zipf distribution (Zipf 1949) is used to find the probability of choosing k^{th} video from a collection of V videos. The probability of

Table 5.2: Simulation data

Servers in local cluster	5
Storage capacity per local server in GBytes	500
Disk transfer rate at local server in Gbps	1.2
Hops to remote cluster1	3
Hops to remote cluster2	6
MTU in bytes	1500
Maximum packet size in bytes	1500
Network bandwidth in Mbps	2488
End-to-end delay in ms	300
Size of video collection	150
Size of videos in GBytes	2.46 to 4.8
Service time in hours	0.68 to 2.03

choosing the k^{th} most popular video from a collection of V videos, is found by Z/k where $Z = 1/(1 + 1/2 + 1/3 + \ldots + 1/V)$. The popularity of the videos in the video collection according to Zipf distribution is shown in Figure 5.2. The order in which they appear is an indication of their popularity, the first video being the most popular and the 150^{th} video the least popular. Using the Zipf distribution, if the overall arrival rate of requests to the VoD system is λ, the arrival rate of k^{th} video is determined by $\lambda_k = \lambda p_k$ where $p_k = Z/k$. Arrival rate based on popularity is used to determine the traffic intensity for each video. Traffic intensity defines the number of requests arriving for a video during the service time of that video request and is determined by $\rho_k = \lambda_k/\mu_k$ (see Chapter 4). Traffic intensity is used as the basis for replication and placement of videos.

Based on the criteria presented in Section 5.3.2, the following request handling policies are defined for evaluating the performance in this simulation:

- Direct — No remote service available.
- Redirect — Blocked requests at the local cluster are redirected to remote cluster1 and blocked requests from remote cluster1 are redirected to remote cluster2.

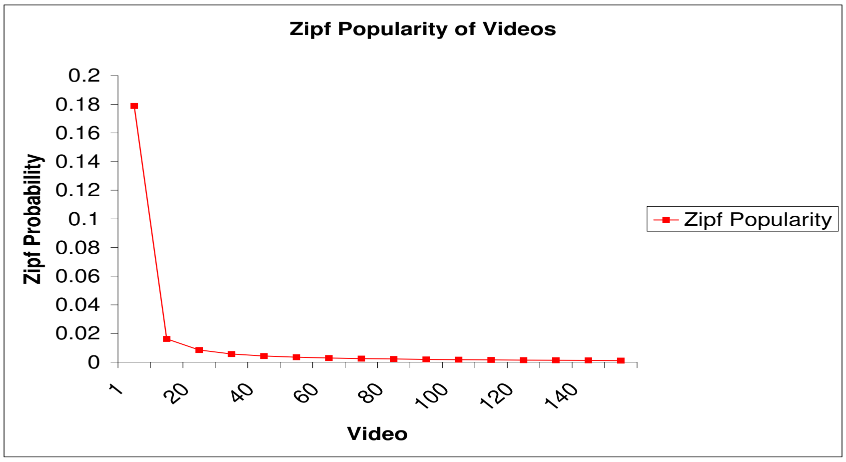

Figure 5.2: Popularity of videos using Zipf distribution

- Split-redirect — The incoming request traffic for videos common to local and remote clusters is first split between local and remote clusters. The remote cluster traffic is again split between remote cluster1 and remote cluster2. For example, a Split-redirect50-60 policy will direct 50% of the incoming request traffic for videos common to both local and remote clusters to local cluster. Of the remote traffic, remote cluster1 will serve 60% and remote cluster2 will serve the remaining 40%. In addition, all blocked requests from local cluster are redirected to remote cluster1. Blocked traffic from remote cluster1 is not redirected to remote cluster2. Several different split values are evaluated.

- Split — Similar to Split-redirect, but blocked requests at local cluster are not redirected to remote clusters. Several different split values are evaluated.

5.6 Performance Evaluation for Single Rate Service

In this section, I present results of performance evaluation for a single rate playback service. The request playback rate is limited to 8 Mbps and the set-top box buffer is

250 Kbytes. Each server in the local cluster has a disk transfer rate of 1.2 Gbps which translates to 150 simultaneous request streams at 8 Mbps. The request arrival process is assumed to be a Poisson process. The range of request arrival rate, λ, is varied between 500 and 1000 requests per hour. For analyzing threshold-based admission control, the workload is fixed and threshold values are varied.

I evaluate the performance of the VoD system under several scenarios as enumerated below. The list of experiments under each of the scenarios are also shown. Results of each experiment are presented in the following sections.

- Replicated Video Collection and Non-blocking Remote Clusters
  1. System Performance — the goal is to find which policies perform better under varying load conditions and why.
  2. Reverse Popularity — the goal is to evaluate VoD performance for a different video collection under varying load conditions.
  3. Threshold Blocking Performance — the goal is to analyze VoD performance under threshold-based admission control and validate analytical results.
  4. Scalability — the goal is to analyze if end-to-end results are scalable and to study the effect of video collection size on a given VoD system size.
- Replicated Video Collection and Shared remote Clusters
  1. System Performance — the goal is to evaluate VoD performance when remote clusters are blocking resources with respect to the reference user population.
- Distributed Video Collection and Non-blocking Remote Clusters
  1. System Performance — the goal is to evaluate VoD performance when only a partial set of videos from the video collection are available at the local cluster.

2. Class-based Performance — the goal is to evaluate blocking performance for local cluster videos and remote only videos.

5.6.1 Confidence Interval Analysis

In this section, the results of estimating confidence intervals for the average overall blocking probability under the three basic policies — redirect, split-redirect, and split — are presented. Detailed results for all scenarios described above will follow in the next several sections.

The method used for confidence interval estimation is based on replication (MacDougall 1987) where we make 6 simulation runs for loads between 500 to 1000 requests per hour. A different random number stream using a new seed is generated in each run. A single simulation run collects events over a period of 10,000 hours. At the lowest load of 500 requests per hour arrival rate, a total of 5 million arrivals per run (observations) are generated during one simulation run.

The overall blocking probability for the VoD system under each policy from the six simulation runs are shown in Tables 5.3 through 5.5. In Table 5.3, overall blocking probability from six separate simulation runs for the redirect policy under loads varying from 700 to 1000 is shown. For loads 500 and 600, the blocking probability is zero. Also shown are the mean and the standard deviation of the blocking probability. Similar results are shown for split-redirect and split policies in Tables 5.4 and 5.5, respectively.

The values of the overall blocking probability at 95% confidence interval are shown in Tables 5.6 through 5.8 for all three policies. The confidence interval half-width is computed using the standard $\boldsymbol{t}$ distribution (MacDougall 1987).

5.6.2 Replicated Video Collection and Non-blocking Remote Clusters

For this scenario, the local cluster consists of five servers and the storage capacity on each server is 500 GBytes with an overall storage availability on the local cluster at

Table 5.3: Overall blocking probability from simulation runs: Redirect

Runs	Load=1000	Load=900	Load=800	Load=700
1	0.186590	0.102666	0.021646	0.000003
2	0.185934	0.102143	0.021246	0.000012
3	0.186331	0.102415	0.021380	0.000002
4	0.186148	0.101986	0.021118	0.000002
5	0.186773	0.102677	0.021728	0.000000
6	0.186190	0.102022	0.020985	0.000001
Mean	0.186328	0.102318	0.021351	0.000000
Std. Dev.	0.000308	0.000312	0.000293	0.000000

Table 5.4: Overall blocking probability from simulation runs: Split-redirect

Runs	Load=1000	Load=900	Load=800	Load=700
1	0.192857	0.113851	0.039381	0.000528
2	0.192283	0.113384	0.038934	0.000557
3	0.192572	0.113566	0.039194	0.000486
4	0.192415	0.113269	0.038815	0.000450
5	0.193012	0.113860	0.039527	0.000523
6	0.192419	0.113200	0.038769	0.000507
Mean	0.192593	0.113522	0.039103	0.000509
Std. Dev.	0.000284	0.00287	0.000313	0.00004

Table 5.5: Overall blocking probability from simulation runs: Split

Runs	Load=1000	Load=900	Load=800	Load=700	Load=600
1	0.197773	0.122248	0.052925	0.009189	0.000273
2	0.197159	0.121801	0.052756	0.009206	0.000242
3	0.197535	0.121976	0.052840	0.009320	0.000276
4	0.197380	0.121733	0.052547	0.009132	0.000283
5	0.197915	0.122144	0.053189	0.009375	0.000258
6	0.197273	0.121863	0.052710	0.009160	0.000291
Mean	0.197506	0.121961	0.052828	0.009230	0.000271
Std. Dev.	0.000293	0.000202	0.000218	0.000095	0.000017

Table 5.6: 95% Confidence interval for blocking probability: Redirect

Load	Mean + half-width	Mean - half-width
1000	0.186650	0.186005
900	0.102646	0.101990
800	0.021658	0.021043
700	0.000000	0.000000
600	0.000000	0.000000
500	0.000000	0.000000

Table 5.7: 95% Confidence interval for blocking probability: Split-redirect65-40

Load	Mean + half-width	Mean - half-width
1000	0.192891	0.192295
900	0.113823	0.113221
800	0.039431	0.038775
700	0.000547	0.000470
600	0.000000	0.000000
500	0.000000	0.000000

Table 5.8: 95% Confidence interval for blocking probability: Split55-50

Load	Mean + half-width	Mean - half-width
1000	0.197814	0.197198
900	0.122172	0.121749
800	0.053057	0.052599
700	0.009331	0.009130
600	0.000289	0.000252
500	0.000000	0.000000

2.5 Terabytes (TBytes). This storage capacity is enough to store the complete video collection on the local cluster. The total storage requirement of the video collection is around 800 GBytes. But additional storage is required to replicate copies of videos as dictated by the replication algorithm in Section 5.2.1. The complete video collection is available on the remote clusters as well.

The dedicated capacity at the remote cluster with respect to the reference user population is at least as large as the network capacity leading up to them. Therefore, the remote clusters are non-blocking resources. The network bandwidth is assumed to be 2488 Mbps on network1 and network2 and therefore, can serve 311 streams of 8 Mbps.

Request handling policies evaluated include redirect, split-redirect, and split. Split-based policies are especially relevant in this scenario because it is easy to find an efficient split by simply dividing the incoming traffic in proportion to the stream handling capacity of the local cluster and the network paths. This is possible because the local and remote clusters are acting as replicated clusters with all videos available on them so that a video request can be serviced at any of the three resources. With the redirect policy, we do not need *a priori* information about either the video distribution or the blocking or non-blocking nature of remote clusters.

System Performance

The purpose of this experiment is to test the performance of the VoD system using different request handling policies. The objectives are:

- to evaluate the general performance of the three main request handling policies, redirect, split-redirect, and split policies under varying load conditions by keeping the VoD system size fixed.
- to evaluate interactions between various split policies.
- to compare split-based policies which divide the incoming traffic load in the same proportion as the resource capacities on the VoD system.

In Figures 5.3 and 5.4, the performance of redirect policy is compared with various split-redirect and split policies. Blocking rates in number of requests blocked per hour are plotted as a function of varying load in requests per hour. Redirect policy has the minimum overall blocking rate in both cases. The redirect policy gives maximum utilization at each resource because remote service is required only when the local cluster cannot handle the load. Through these results, I show that even with only limited redirection it is possible to improve the performance of the VoD system.

Among split-based policies, the 55-50 split for the split policy and 65-40 split for the split-redirect policy have the lowest overall blocking rate. At low loads, the difference between split and split-redirect is minimal in which case an adaptive request handling policy could be devised by eliminating redirection. Depending on the load, split or split-redirect policies may be enforced.

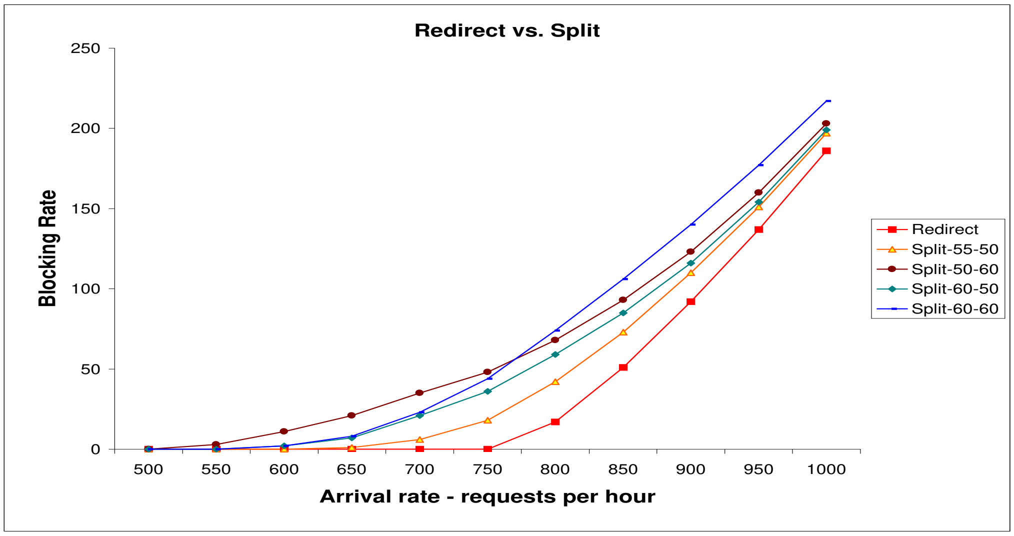

Figure 5.3: Blocking rate: Redirect and Split

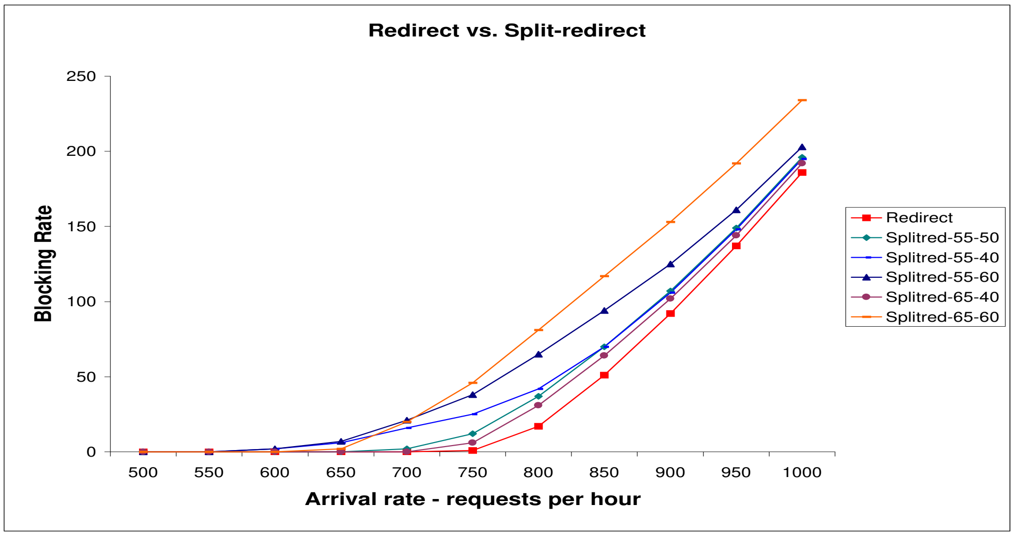

Figure 5.4: Blocking rate: Redirect and Split-redirect

Discussion

The interaction between various splits in the split-based policies are characterized by *crossover* and *divergence*. For instance, in Figure 5.3, we see a crossover between split-50-60 and split-60-60 policies. At low loads of 500 to about 800 requests per hour, split-60-60 policy performs better than split-50-60 policy. To explain the reason for crossovers and divergences, we have to look at the actual percentage of traffic directed at each resource under a split-based policy (see Figure 5.5). The actual traffic directed toward a resource will determine how efficiently that resource is being utilized. Under split-50-60 policy, the proportion of traffic going to local, remote cluster1 and remote cluster2 is 50, 30, and 20 percent respectively and under split-60-60 policy, it is 60, 24, and 16. The reason for the crossover is that at low loads, the local cluster is better utilized and networks are not congested yet with the split-60-60 policy. But at high loads, both local and network1 are overutilized and network2 is under-utilized resulting in worse performance than split-50-60 policy.

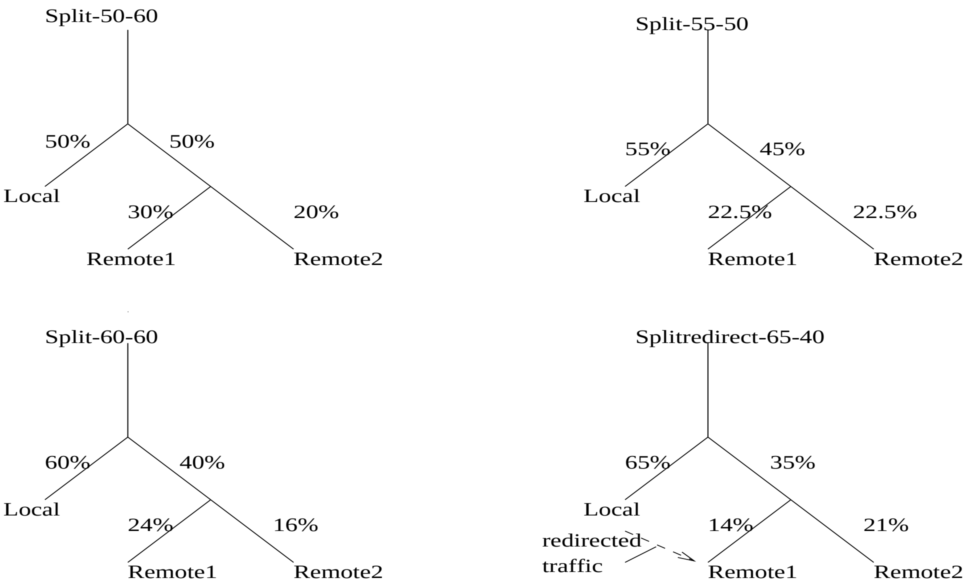

Figure 5.5: Traffic to local and remote clusters for different splits

In Figure 5.3, we notice a divergence behavior with the split-60-50 and split-60-60 policies. At low loads of up to 650 requests per hour, both policies have the same performance. The divergence occurs at around 650 requests per hour with split-60-60 policy having worse performance. The reason for this is that at higher loads, network1 gets congested faster than the other resources for the split-60-60 policy.

In Figure 5.6, several split-redirect and split policies are plotted again characterized by crossovers and divergences. We see a crossover between split-redirect-65-60 and split-redirect-55-60 policies. A divergence occurs between split-redirect-55-60 and split-redirect-55-40.

In general, crossovers are driven by the proportion of traffic directed toward the local cluster and divergence by the remote traffic. Also note that a certain split that performs well under low load conditions will deteriorate quickly as load is increased as shown for policy split-redirect-65-60 in Figure 5.6.

Analyzing blocking at individual resources helps explain the crossover and divergence behavior; for example, the crossover between the split-redirect-65-60 policy and the split-50-60 policy or the divergence between split-redirect-65-60 and split-redirect-

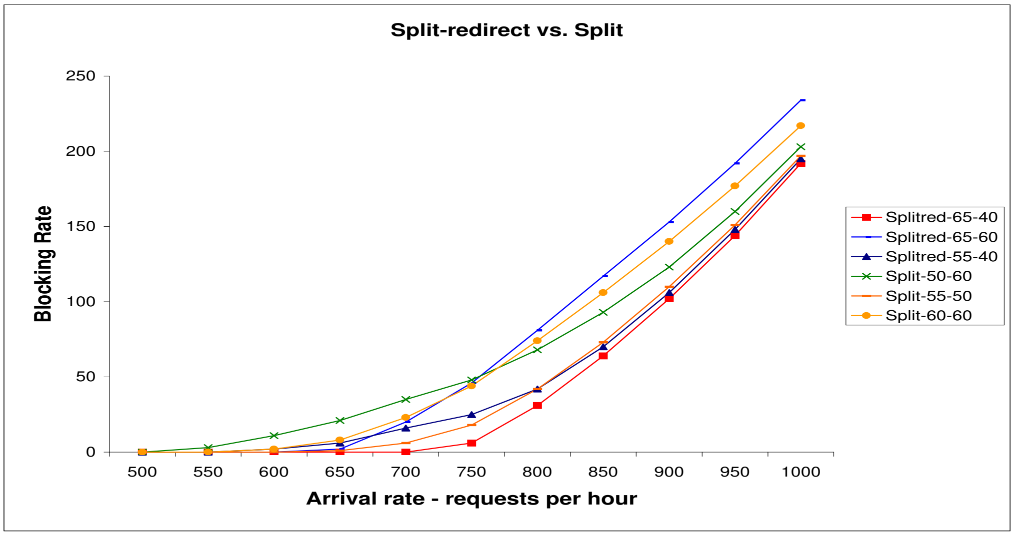

Figure 5.6: Comparing Split-redirect and Split Policies

65-40 in Figure 5.6. At low loads there is already a significant blocking activity at network1 for the split-50-60 policy (see Figure 5.8), but blocking is more gradual. However, the crossover occurs because the performance of the split-redirect-65-60 policy deteriorates at a faster rate as a result of overloading network1 with redirected traffic from the local cluster and underutilization of network2 (see Figure 5.7).

A divergence between split-redirect-65-60 and split-redirect-65-40 is explained by the overloading of network1 at loads higher than 650 requests per hour as shown in Figure 5.7. The overloading of network1 is more gradual under the split-redirect-65-40 policy as shown in Figure 5.9.

In Figure 5.10, blocking at resources for the split-55-50 policy is plotted. Because of the even split between network1 and network2 traffic, the blocking at these resources are the same. Up to a load of 650 requests per hour, both split-55-50 and split-redirect-65-40 policies have the same performance, after which the split-55-50 policy deteriorates at a more rapid pace than the split-redirect policy. The reason is that all three resources, the local cluster, network1, and network2 experience significant blocking (see Figure 5.10).

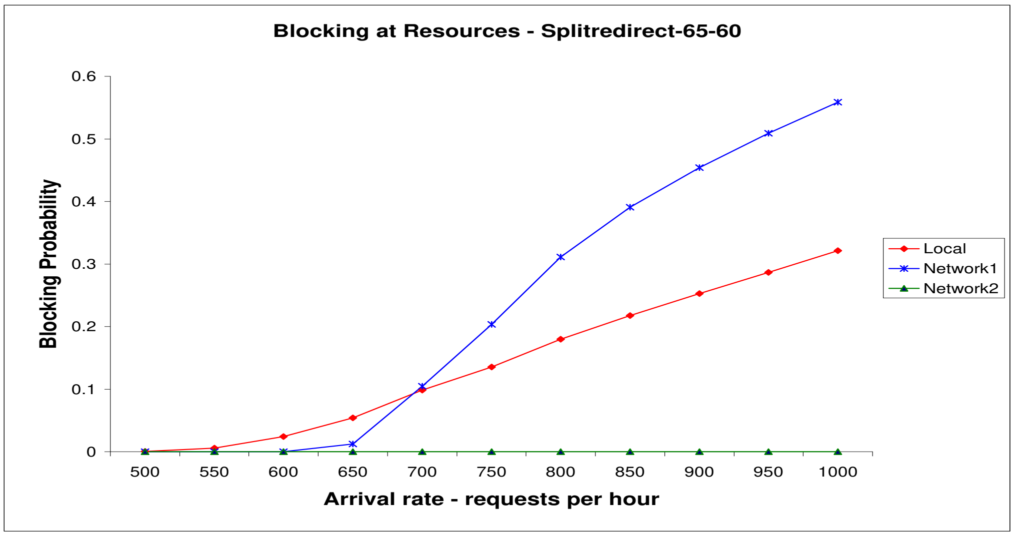

Figure 5.7: Blocking at resources: Split-redirect-65-60

In the VoD architecture, the stream handling capacity at the local cluster is 750 requests and the network stream handling capacities are 311 requests each for the two routing paths to the remote clusters. Therefore, 55% of the overall VoD system capacity is at the local cluster with the networks providing 22.5% each. We can find efficient split policies by choosing splits that are close to this proportion. This is possible only if the portion of remote cluster capacity dedicated to each user population is known ahead of time. However, when remote clusters are dynamically shared with other user populations, it will be difficult to find efficient split policies and redirect policy will be the only alternative in that situation. In Figure 5.11, a comparison of policies with splits proportional to resource capacity available on local and remote clusters is shown. Split-55-50 policy has the worst performance and redirect policy

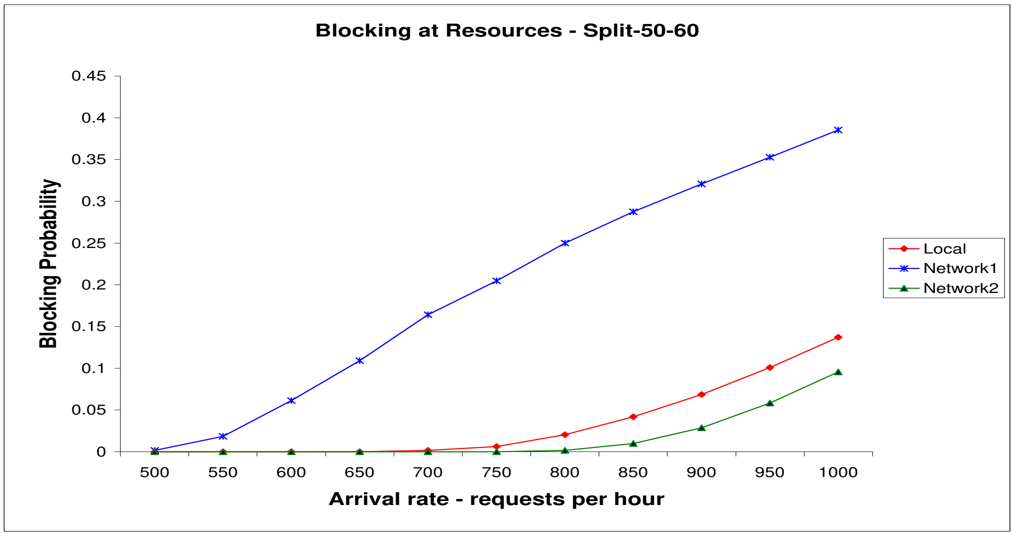

Figure 5.8: Blocking at resources: Split-50-60

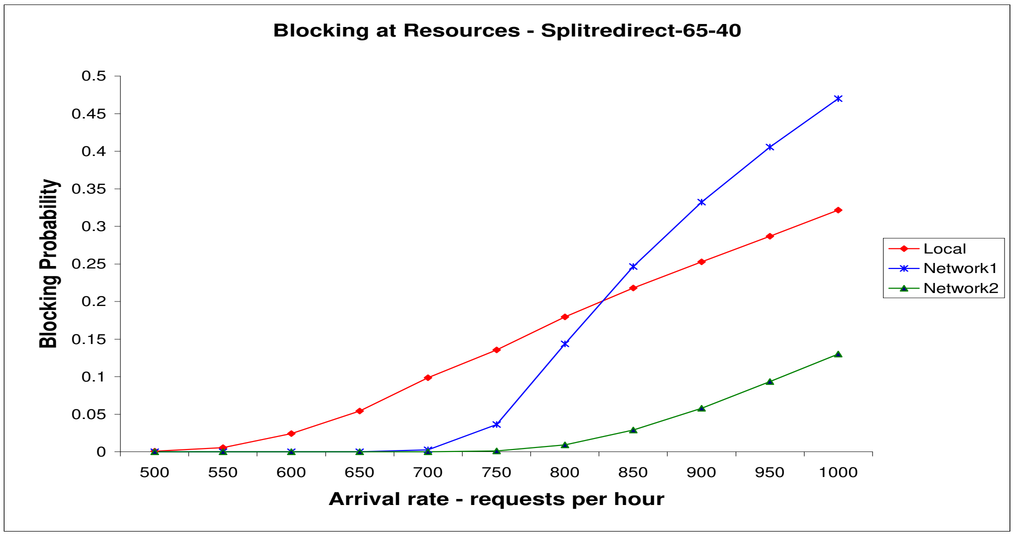

Figure 5.9: Blocking at resources: Split-redirect-65-40

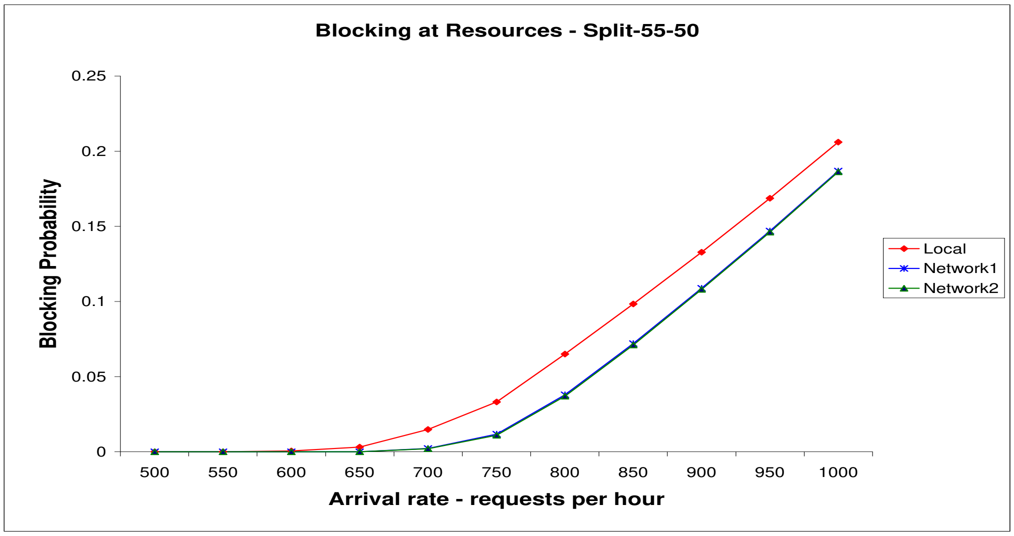

Figure 5.10: Blocking at resources: Split-55-50

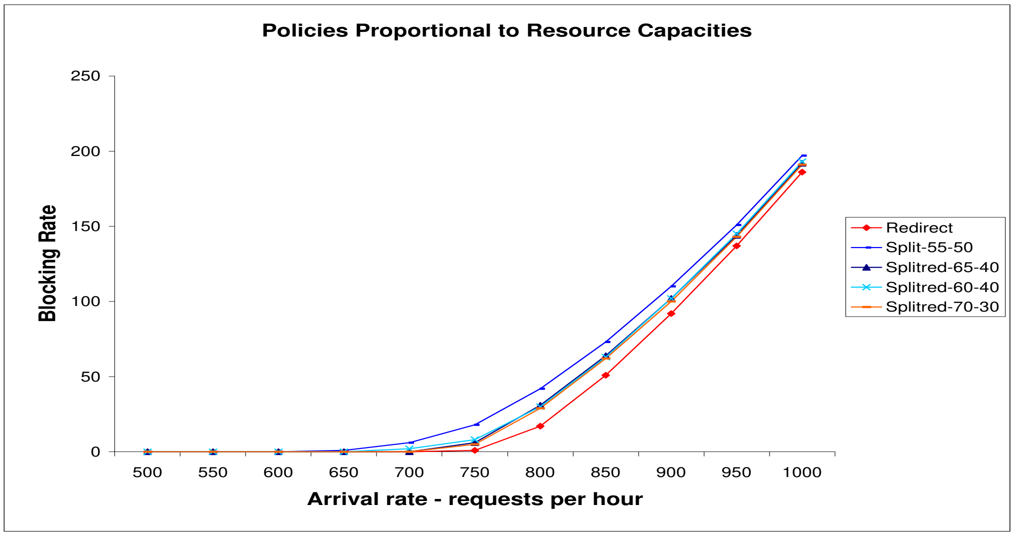

Figure 5.11: Blocking rate: splits proportional to resource capacities

the best. Among split-redirect policies, 65-40, 60-40, and 70-30 splits have similar performance. Even a one-level redirection as provided by the split-redirect policies is enough to result in better performance than the most efficient split policy.

Reverse Popularity

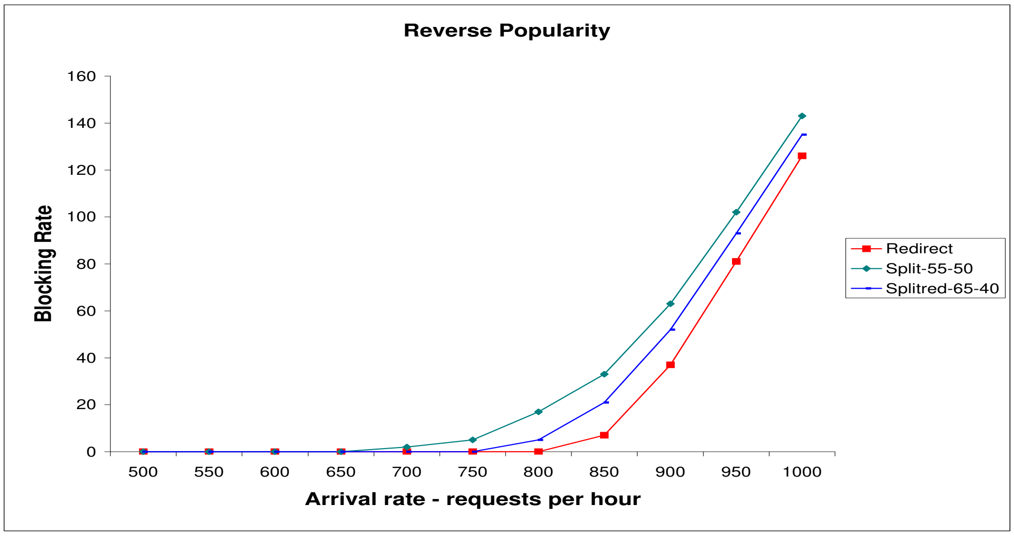

Figure 5.12: Reverse popularity

The purpose of this experiment is to evaluate the VoD system performance for a different video collection. To obtain a different video mix, the popularity of the videos in the collection is reversed with the least popular video having the most demand. In the video collection, the 150^{th} video is the least popular and happens to be the shortest video. When we reverse the popularity, the shortest video becomes the most popular. Since Zipf distribution is a highly skewed distribution, the service time of the most popular video defines how well each of the policies perform under a given load. Because of this reason, all policies have better performance for the reverse video mix compared to the results from the previous section.

In Figure 5.12, I show that the redirect policy still performs well when the popularity of the videos is reversed. This result indicates that the performance *order* exhibited by the three basic policies is insensitive to service times or different video mix. The relative order of performance of the three basic policies is the same for any video collection of a given size and the service times of the videos.

Threshold Blocking

In this section, I show the effects of class-based threshold restrictions on the performance of the request classes for the VoD system. The goal of this experiment is to show that the performance results from the simulation are similar to the analytical results from Chapter 4. Since we have an ordered list of videos based on popularity, class 1 is defined as the top 20% of the videos and class 2 as the middle 40% of the videos with class 3 having the bottom 40%. The threshold values for class 2 and class 3 are in the range of 0% to 5%.

In Figures 5.13 to 5.18, the effect of threshold blocking under each of the three policies is shown. Class-based blocking is evaluated as a function of threshold restrictions on either class 2 or class 3 requests. The behavior shown in these figures matches that of the analytical model from Chapter 4. The threshold experiments are done at a request load of 850 requests per hour. Requests for class 2 and class 3 consisting of the less popular videos are subjected to thresholds. In Figure 5.13, class 2 threshold restriction results in 0% to 5% blocking for class 1 and class 3 videos. Blocking for class 2 is at 100% at 0 threshold, meaning that no requests of class 2 are accepted into the system. Blocking levels off around 5% for all three classes at about 3% threshold when the threshold has no effect since the arrival rate is below the threshold anyway. In Figure 5.14, the effect of class 3 threshold on class 1 and class 2 blocking are shown. Similar analysis is applied to split-redirect and split policies and results shown in Figures 5.15 to 5.18. The behavior shown in all these figures conforms to the analytical model from Chapter 4.

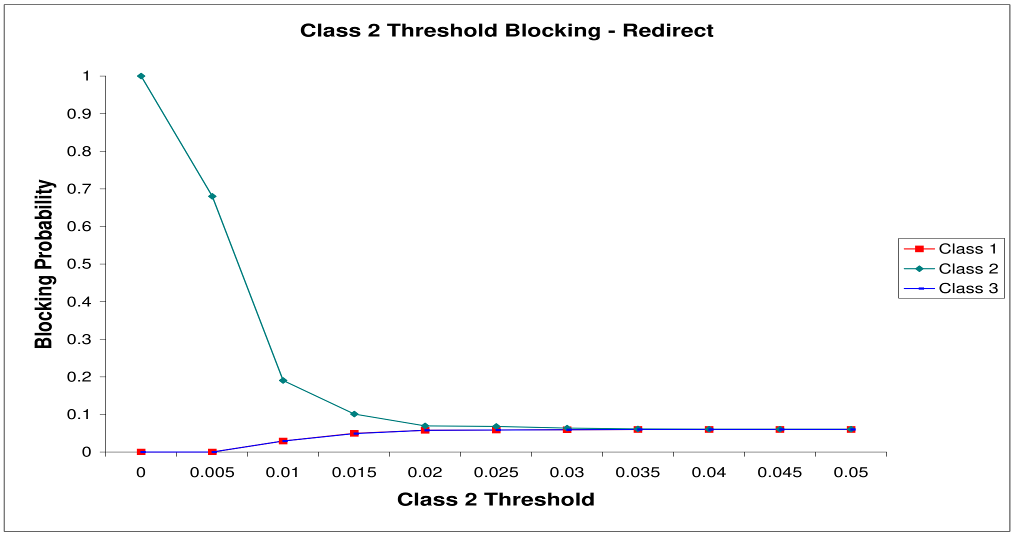

Figure 5.13: Class 2 threshold blocking: Redirect

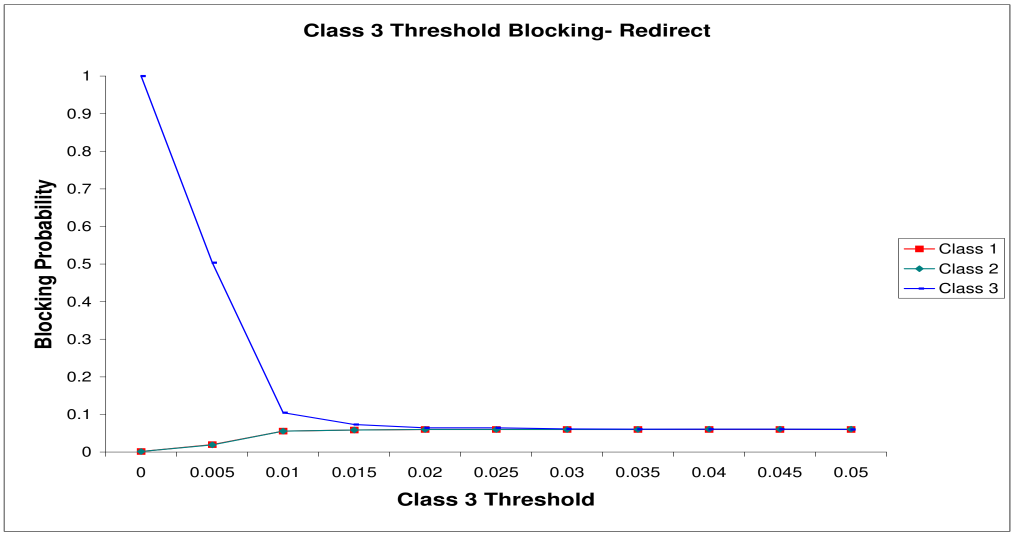

Figure 5.14: Class 3 threshold blocking: Redirect

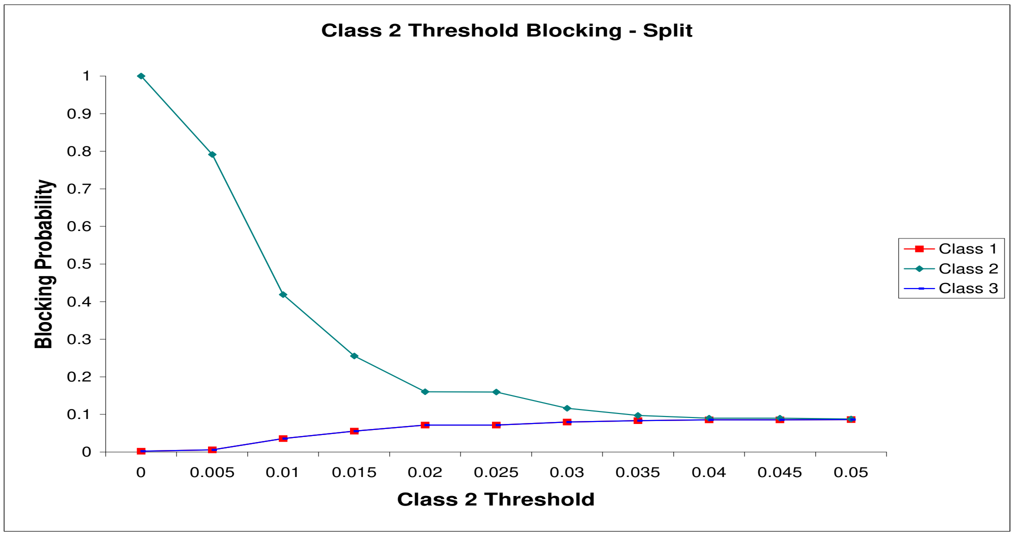

Figure 5.15: Class 2 threshold blocking: Split

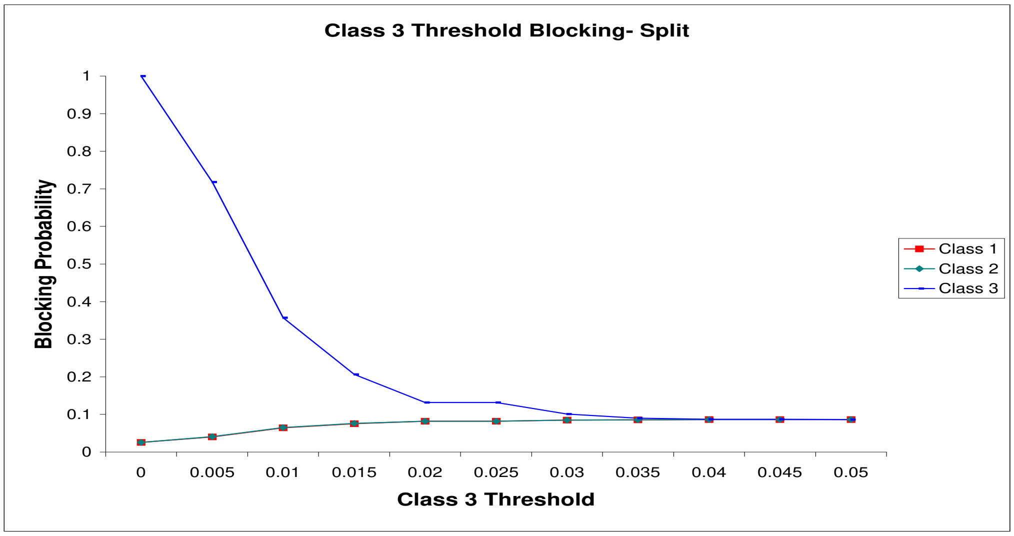

Figure 5.16: Class 3 threshold blocking: Split

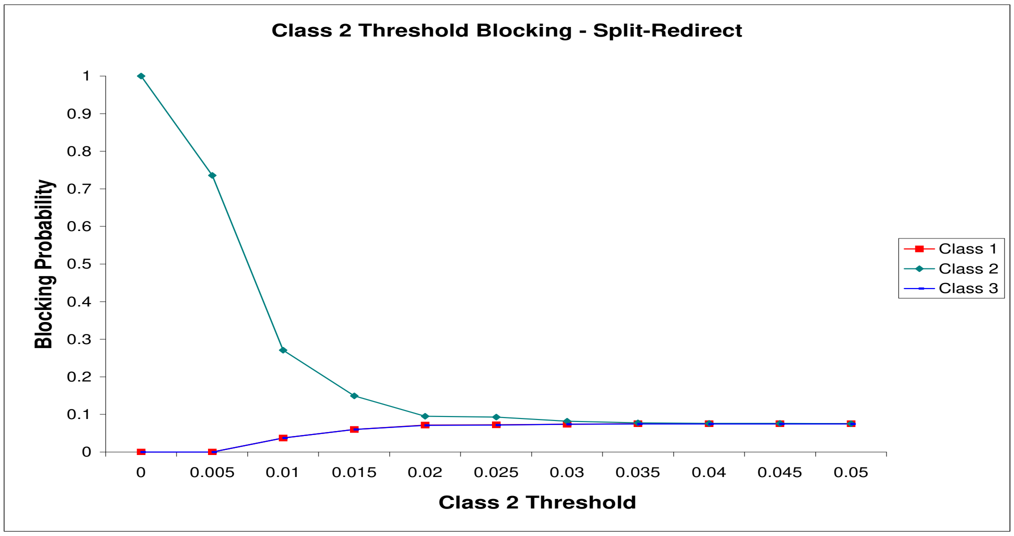

Figure 5.17: Class 2 threshold blocking: Split-redirect

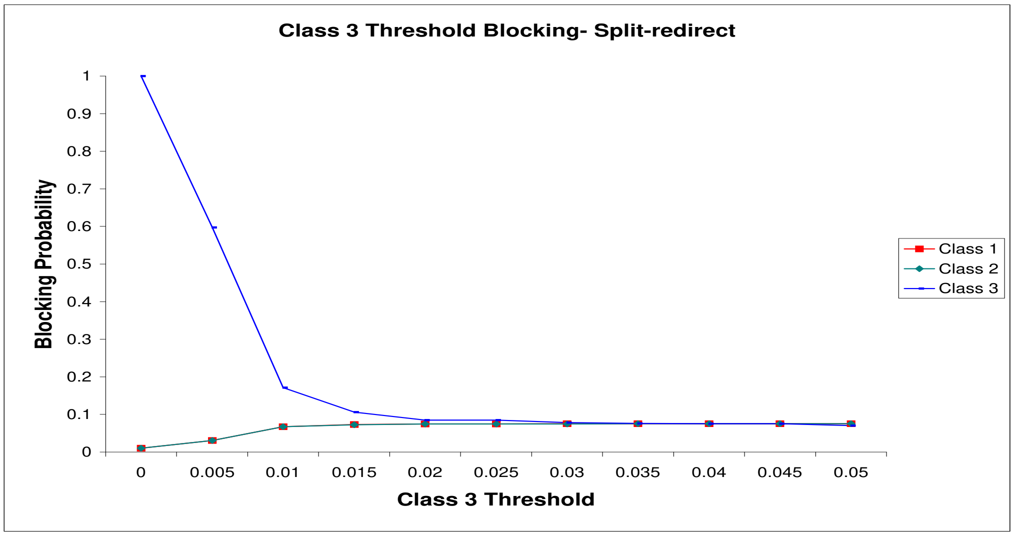

Figure 5.18: Class 3 threshold blocking: Split-redirect

Table 5.9: Scalability data

Workload	Num. of Servers	Network1	Network2
170	1	498	498
340	2	996	996
510	3	1494	1494
680	4	1992	1992
850	5	2488	2488
1020	6	2988	2988
1190	7	3486	3486
1360	8	3984	3984
1530	9	4482	4482
1700	10	4980	4980

Scalability

In this set of experiments, the resource capacities in the VoD system are varied in proportion to the request workload to demonstrate that the overall blocking probability remains the same. However, the video collection size is kept constant for all experiments which will enable us to study the effect of video collection size over VoD system size. The primary purpose of this analysis is to show that the results of the proposed end-to-end evaluation of the VoD system performance are scalable. The experiment is constructed as follows - we fix the load per server and the network bandwidth on the two network paths and increase all four in lock step. For instance, at 170 requests per hour, the number of servers at the local cluster is equal to one and the network bandwidth is 498 Mbps. As we double the workload to 340 requests per hour, we double the number of servers and the network bandwidth. Table 5.9 shows the values used in this experiment.

In Figure 5.19, I show that the results of the proposed end-to-end analysis are scalable for the three request handling policies. The overall blocking probability is plotted as a function of VoD system size. The VoD system size is varied in terms of the number of servers in the local cluster and the network capacities along the routing paths to the remote cluster. The load is proportionately varied. I show that the blocking performance levels off at about 5% for the redirect policy. The blocking performance for the split-redirect-65-40 policy is shown leveling off at 6% and for the split-55-50 policy it is shown to level off at 8%.

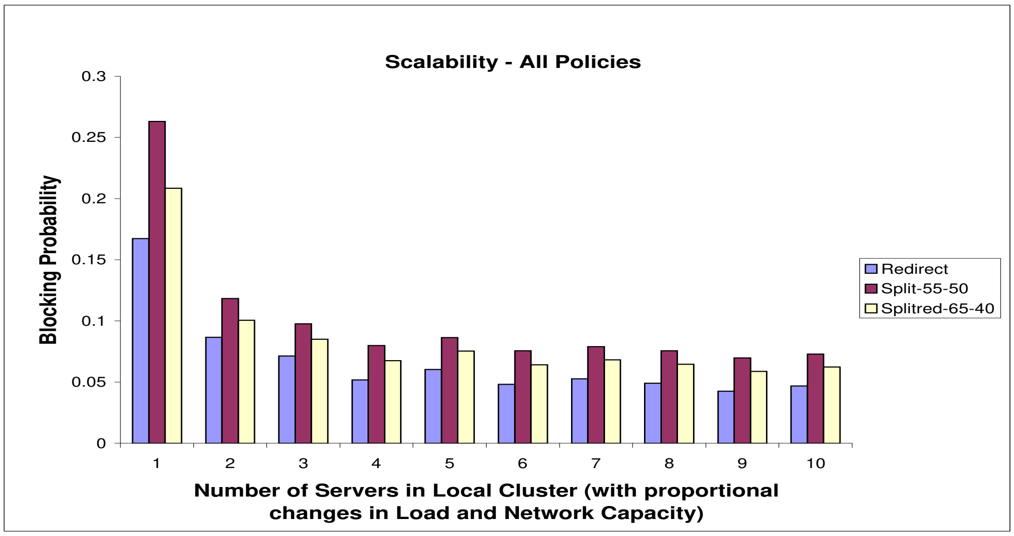

Figure 5.19: Scalability: all policies

The significance of this result is that as the VoD system size is scaled up from one server to 10 servers at the local cluster with appropriate network bandwidth increases, the overall blocking stabilizes gradually and remains constant. One exception to this behavior occurs at one server capacity. The reason for high blocking for the one server capacity is that nearly 70% of the video collection has to be serviced from the remote clusters. Only the top 30 videos are stored in the local cluster because of limited storage capacity. Serving requests from the remote clusters overloads the network resulting in very high blocking. The other important factor contributing to the degraded performance is the lack of replication because we only have one server in the local cluster. The number of copies per video is limited to one. Therefore, the advantage of replication of top videos is lost leading to worse performance. Even at two-server capacity, blocking is still high for the same reason. As we increase the number of servers, the replication advantage becomes marginal. More on the importance of replication is presented in Section 5.6.5.

5.6.3 Replicated Video Collection and Shared Remote Clusters

System Performance

The purpose of this experiment is to evaluate VoD performance when remote clusters are shared by other user populations and block requests from the reference user population. The available capacity on remote clusters is varied to mimic the situation where the clusters are shared by other user populations. This situation is similar to a dedicated scenario where each user population gets a share of the remote cluster capacity. Conditions of having no remote service to non-blocking remote clusters are experimented. Remote clusters do not block when they provide as much capacity as the network paths leading up to them from a user population.

In Figure 5.20, the overall blocking probability of the VoD system for varying request load between 500 and 1000 requests per hour is plotted. The available capacities on the remote clusters are varied between 0 and 350. Since the bandwidth on the network path to remote cluster is 2488 Mbps, the network can simultaneously service 311 streams of 8 Mbps. Therefore, remote cluster becomes a blocking resource for stream handling capacities less than 311. The performance of the VoD system under redirect policy is shown in the Figure 5.20.

The performance deteriorates as available capacity on the remote cluster decreases. The blocking is the highest when there is no remote service available. This situation is similar to adopting the direct policy for request handling where no remote service is allowed. At the other extreme, the remote clusters at a stream capacity of 350 become non-blocking with only the network providing resource contention.

In a dedicated scenario where the available capacity with respect to the reference user population is known, the split policies may be found using the proportion of resource levels. However, split-based policies cease to have any relevance in a *dynamically* shared remote cluster environment because it is difficult to find an efficient split without the knowledge of available capacity on remote clusters.

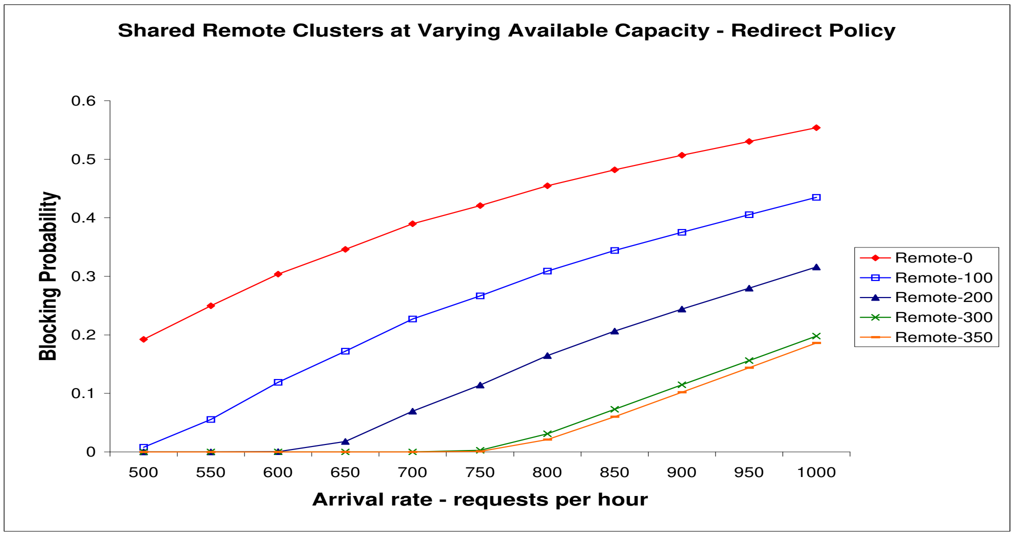

Figure 5.20: Shared remote clusters

5.6.4 Distributed Video Collection and Non-blocking Remote Clusters

System Performance

The purpose of this experiment is to evaluate blocking performance of the VoD system when only a partial list of top videos are stored on the local cluster. Requests for the remaining videos will be served from the remote clusters. This partial allocation makes more stream capacity available to the top fraction of the videos at the local cluster. This will reduce blocking and therefore, redirected traffic from the local

Table 5.10: Video distribution for different storage capacities

Storage used (in GB per server)	Local	Remote
100	30	120
200	76	74
300	135	15
400	147	3
500	150	-

cluster, but at the same time we expect increased blocking for videos served from the remote clusters only. The remote clusters are non-blocking in this experiment.

The basic idea behind the replication algorithm from Section 5.2.1 is to distribute the available storage capacity at the local cluster to videos in proportion to their popularity. Therefore, some low demand videos will not be assigned to local cluster. In this experiment, by varying the storage capacity used at the local cluster, we are able to allocate a partial to complete video collection to the local cluster. The storage capacity used is varied from 100 to 500 GBytes per server in the local cluster. The local cluster size is still maintained at 5 servers. The video distribution between local and remote clusters at each storage capacity used is given in Table 5.10. For instance, if a storage capacity of 200 GBytes per server is used on the local cluster, we have top 76 videos placed on the local cluster. The demand for the rest of the videos has to be exclusively met by the remote clusters.

In Figure 5.21, the effect of video distribution is shown at a varying load of 500 to 1000 requests per hour for the redirect policy. The video distribution at storage capacity 100 GBytes per server results in the lowest blocking performance. At 500 GBytes per server storage capacity, we have a replicated video collection both at the local and the remote clusters. This condition has the highest blocking for varying load conditions. The reason for low blocking at low storage capacities is that more stream capacity is available to a top fraction of the video collection at the local cluster. Providing more videos at the local cluster increases resource contention resulting in higher overall blocking.

Low overall blocking may be desirable, but partial allocation of the video collection to the local cluster results in different blocking performance for individual videos. At 100 GBytes per server used storage capacity, class 2 and class 3 videos which account for the bottom 80% of the video collection experience more than 25% blocking as shown in Figure 5.22 at a load of 1000 requests per hour. At 200 GBytes per server used storage capacity, the bottom 40% of the videos still experience above 25% blocking. The decision to partially or completely allocate videos to local cluster must consider the fairness issue toward each video request. A replicated video collection at all resources provides the same blocking probability for all videos whereas a distributed video collection will induce additional blocking for videos found on fewer resources because of reduced stream handling capacity available to those videos.

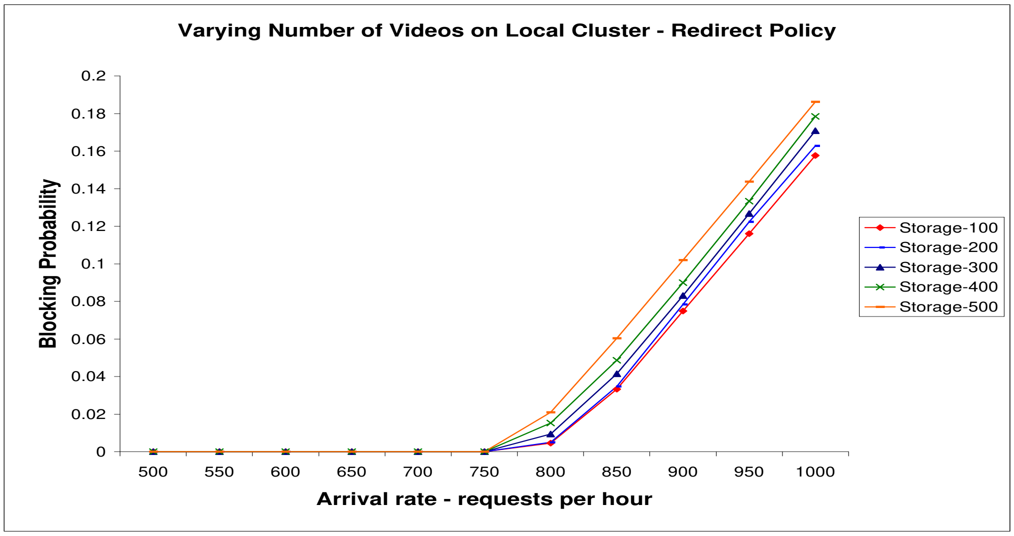

Figure 5.21: Different video distribution: average blocking for Redirect policy

The class-based performance of the VoD system under distributed video collec-

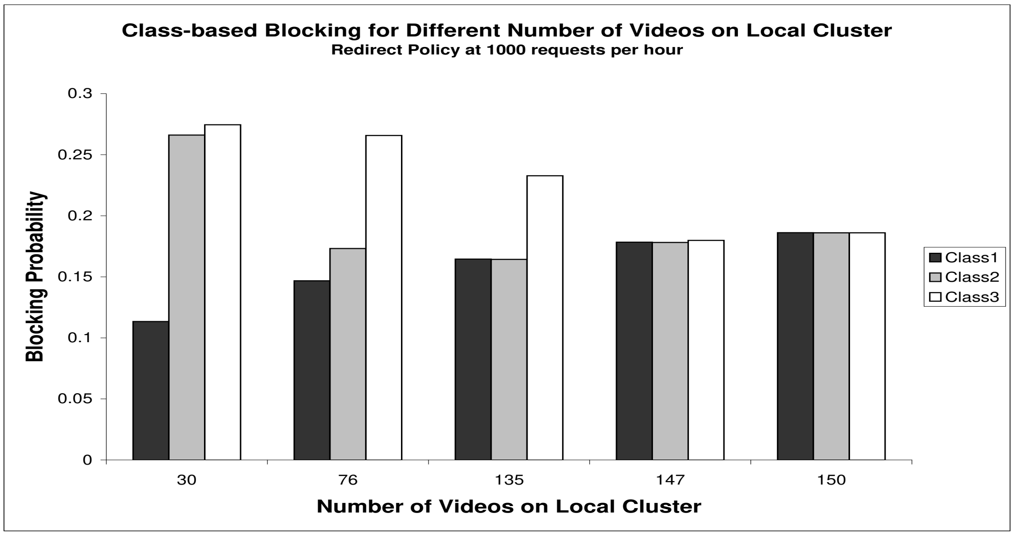

Figure 5.22: Different video distribution: class-based blocking at load = 1000

tion is similar to the threshold performance. By forcing some of the traffic to remote clusters, more stream handling capacity at the local cluster is made available to the popular videos. This will reduce blocking for those videos as seen in Figure 5.22 at the cost of increased blocking for videos not found at the local cluster. Threshold performance however, may be easily predicted by setting appropriate threshold parameters based on the demand. Finding the exact video distribution to realize a certain class-based blocking performance will not be that trivial. It will require *a priori* computation about the video distribution according to the replication algorithm and storage capacities available.

In Figures 5.21 and 5.22, the results are shown with respect to redirect policy. Split-based policies are not appropriate in a distributed video collection environment because of the difficulty in estimating the traffic split without *a priori* computation. The appeal of split-based policies is their simplicity which will be lost if any computation is required ahead of time for finding an efficient split.

The performance of a distributed video collection in a shared remote cluster environment can only at best equal what is presented in this section. Again, redirect policy is more appropriate because of the difficulty in finding an efficient split-based policy in a shared environment.

5.6.5 Significance of Replication

In this section, I discuss the significance of replication of videos at the local cluster by cross referencing results from different experiments. I show through these results that it is better to have many smaller servers in the local cluster than one large server to take advantage of the proposed replication scheme.

In Figure 5.23, blocking probability results from three different experiments for the redirect policy are shown. The first experiment is part of the scalability results from Section 5.6.2 where we have a single server in the local cluster at 500 GBytes storage, 150 request stream capacity, and the load is 170 requests per hour. The blocking performance is at 16%. The second experiment is part of the video distribution results from Section 5.6.4 where we vary the storage capacities used per server in the local cluster. At a storage capacity of 100 GBytes used per server and 5 servers in the local cluster with an overall stream handling capacity of 750 requests and a load of 850 requests per hour, the blocking probability is about 3%. Comparing the two experiments, we have the same local cluster storage capacity of 500 GBytes, and the load and the stream capacities are proportional. In both experiments, since the storage capacity is the same, we have the same video distribution between local and remote clusters. The top 30 videos are placed on the local cluster while the remaining 120 videos are directly serviced from the remote clusters. Yet, the blocking performance is vastly different.

The reason for the poor performance in the first experiment is the lack of replication because we have a single server in the local cluster. Though the server has a 500 GBytes storage capacity, the number of copies of videos is limited to one (see replication algorithm in Section 5.2.1). In the second experiment, the top videos are replicated up to 5 times because we have 5 separate servers in the local cluster. Because the load is distributed on the five replicated copies of videos, the redirected load on the networks is reduced resulting in low blocking probabilities. Using this result, I conclude that it is better to have many smaller servers in a local cluster than one large server to take advantage of the replication process.

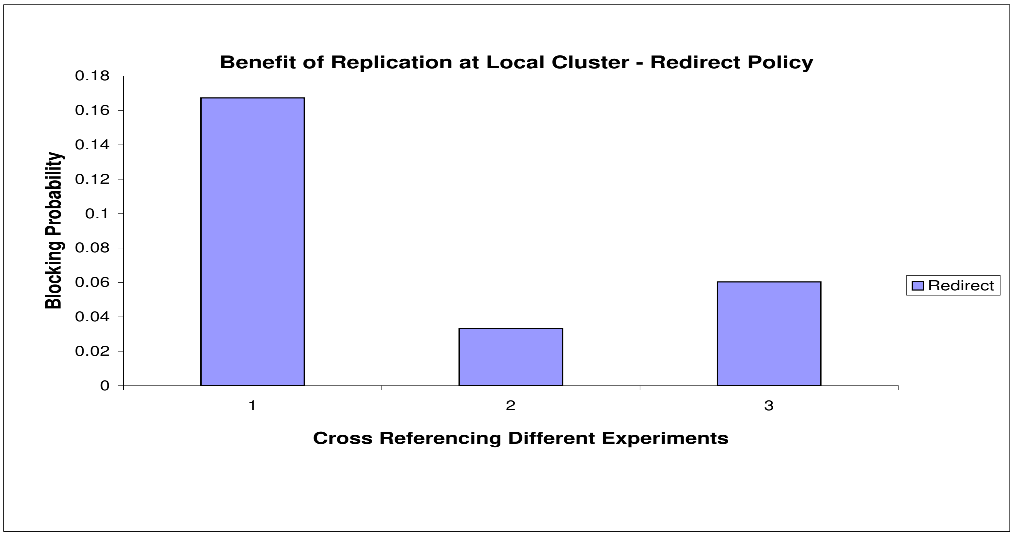

Figure 5.23: Benefit of replication at the local cluster

Comparing experiments 2 and 3 in Figure 5.23, the same performance as documented in Section 5.6.4 is observed. That performance being that the distributed video collection results in lower overall blocking. The VoD system capacity and load conditions in both experiments are the same except for storage. In the second experiment, we have five servers each with 100 GBytes storage in the local cluster whereas in the third experiment, we have five servers each with 500 GBytes storage. In the second experiment, we only have a partial list of top videos from the video collection available at the local cluster. The third experiment is part of the scalability results from Section 5.6.2 where we have the complete video collection available at the local cluster. As in Section 5.6.4, the partial allocation results in lower overall blocking probability because there is less contention of resources for the top videos at the local cluster. However, it also introduces higher blocking for videos serviced only from the remote clusters.

There are other advantages to the proposed replication scheme. The simulation results show that the per video blocking probability in a local cluster or on the network is almost the same for each video in the video collection. Even though the Zipf distribution is highly skewed, we notice that because of the aggressive replication scheme, the demand for the top videos is distributed among the replicated copies resulting in more or less the same load on a video or video copy. Even the simple round-robin type placement scheme implemented in the video placement algorithm results in a balanced load on individual servers in the local cluster.

5.7 Single Rate Performance Summary

In the previous section, I presented performance results for a single rate playback VoD system. Four scenarios shown in Figure 5.24 were considered and the applicability of different request handling policies under each of them was analyzed. The results show that the redirect policy even with only limited redirection has better performance than split-based policies. Redirect policy also provides simpler connection establishment semantics though with a higher implementation overhead.

Under replicated video collection and non-blocking remote clusters, I showed that:

- The performance of redirect policy is better than other policies despite having the highest path establishment overhead.

- The split-based policies are most efficient when the incoming traffic is split in proportion to the available resource capacities.

	Non-Blocking Remote Clusters	Shared Remote Clusters
Replicated Video Collection	Redirect Split Split-redirect	Redirect
Distributed Video Collection	Redirect	Redirect

Figure 5.24: Design considerations

- The relative order of performance remains the same when the popularity of the videos is reversed.

- Threshold-based admission reduces blocking rates for the more popular videos, though at the cost of increasing blocking for video classes that have threshold restrictions.

- The threshold performance conforms to the analytical results from Chapter 4.

- As the system size is scaled up, the overall blocking remains the same for increasing demand, indicating that the results of the proposed end-to-end analysis are scalable.

- The relative order of performance for different request handling policies also remains the same as the VoD system size is varied; redirect policy performs better than either the split-redirect or the split policies.

- Hierarchical organization provides storage for less popular and archival videos at remote clusters in addition to providing supplemental request handling capacity to minimize overall blocking.

- An adaptive request handling policy where one policy replaces another based on load conditions will enhance the overall performance of the VoD system.
- Analyzing blocking performance at individual resources provides valuable insight into the performance of different split-based policies at various loads.

Under replicated video collection and shared remote clusters, I showed that:

- Remote clusters with stream handling capacities less than the networks will induce blocking.
- Because of the shared environment, finding efficient split-based policies requires *a priori* knowledge of the available capacities on all resources with respect to a reference user population which may be difficult to find in a dynamically shared resource environment.
- Redirect policy is most appropriate with its simple connection establishment semantics; we do not need any *a priori* information about available capacities on resources in the VoD system.

Under distributed video collection, I showed that:

- Blocking for the top fraction of the most popular videos placed on the local cluster can be reduced. However, the videos served only from the remote clusters will experience higher blocking because of the reduced resource availability.
- The overall average blocking probability for distributed video collection is lower than that of replicated video collection scenario.
- The redirect policy is a better alternative to split-based policies. Finding efficient splits requires *a priori* computation about the video distribution and the additional traffic to remote clusters which takes away the appeal of simple implementation for split-based policies.

- The performance will diminish if remote clusters are shared with other user populations.

Finally, through cross referencing results from different experiments, I showed that it is better to have many smaller servers in a local cluster because of the replication advantage than to have one single large server. I also showed that by replicating high demand videos and putting them on different servers, balanced load on servers is achieved.

5.8 Performance Evaluation Using Multi-rate Playback

In this section, I present performance evaluation of the VoD system with multiple playback rates. An incoming video request may specify a playback rate at which data gets consumed at the set-top box on the client side. Classes of requests based on these playback rates are defined. The purpose of this simulation is to evaluate the performance of the VoD system by providing multiple levels of service for different classes of requests.

The VoD architecture and other related data from Section 5.5 remain the same for this evaluation including the same request handling policies. Instead of a fixed playback rate, requests are served at three different rates — 4 Mpbs, 8 Mbps, and 16 Mbps. Corresponding to the rates, the buffer sizes used in the simulation are 125 KB, 250 KB, and 500 KB respectively which coincide with the size of the retrieval blocks. Three classes of requests for each playback rate and retrieval block are defined accordingly. The rates being multiples of the minimum rate is not a limitation. We could have used any other set of rates and buffer sizes to model the proposed multi-rate service (see Section 3.6 from Chapter 3).

Storage requirements of videos depend on the compression scheme carried out at the desired playback rate. Since each video request can now be served at three different rates, we need to store three different versions of the same video corresponding to the playback rate. Storing three different versions for the same video will increase the storage requirement of the video collection two and a half times.

In the simulation, there are three ways to implement a multi-rate playback model. They are:

1. We can have dedicated servers in the clusters for each rate; roundlength and amount transferred per request depends on the rate assigned to that server. This will cause imbalance if demand for a certain rate outweighs the others.

2. We can have parallel rounds and buffers for each rate on the same server but this will incur implementation overhead.

3. Same round will accommodate all rates on all servers.

I implement the third idea in the simulation because of its simpler and better implementation. This idea is the same as the model presented in Section 3.6 in Chapter 3.

The overall blocking performance for the multi-rate playback simulation is found using the same metrics from Section 5.3.1 but modified as follows. Equations 5.1 and 5.2 are modified as shown below to account for request classes.

The overall blocking probability, P^m_{vod}, is computed as the weighted average over all classes and all videos :

$$P^m_{vod} = \sum_{j}^{R} \sum_{i}^{V} q_j p_i b_{ij} \tag{5.3}$$

and the overall blocking rate, BR^m_{vod}, is:

$$BR^m_{vod} = \sum_{j}^{R} \sum_{i}^{V} \lambda q_j p_i b_{ij} \tag{5.4}$$

where b_{ij} denotes the blocking probability of a request for i^{th} video from j^{th} class; R denotes the number of request classes; q_j denotes the probability of a request belonging to j^{th} class; V denotes the number of videos in the video collection; p_i denotes the probability of a request for i^{th} video. Class-based blocking probabilities computed as the weighted average over all videos for a particular class are used in threshold-based analysis.

Probability, p_i, of a request for i^{th} video is found using Zipf distribution as before. Probability, q_j, of a request belonging to j^{th} class is found by making assumptions about the request mix from the reference user population. For instance, we may find that 20% of the clients' equipment can only support class 1 requests at 4 Mbps, the other 20% can handle class 2 request at 8 Mbps, and the remaining 60% can handle class 3 request at 16 Mbps. We use the notation (20,20,60) to indicate the request class mix from the reference user population. Clients are assumed to be downward compatible meaning that a class 3 client can handle a transmission at 4 mbps but not the other way around. I later experiment with downgrading blocked class 2 and class 3 requests to class 1 and evaluating the performance of the VoD system.

5.8.1 System Performance

The purpose of this experiment is to evaluate the performance of the VoD system under multi-rate service. In particular, I examine the performance of the three basic request handling policies, redirect, split-55-50, and split-redirect-65-40 for different request mixes. Class-based performance under threshold restrictions are studied. Finally, appropriate service options such as downgrading a blocked class 2 and class 3 request are explored and the overall VoD system performance is analyzed.

The resource levels in the VoD system remain the same as in Section 5.6.2 for the single rate playback. Remote clusters are assumed to be non-blocking. Because of the increase in storage requirement per video, only a partial set of videos from the video collection are available at the local cluster. About top 50 videos are placed on the local cluster while the remaining videos are served from remote clusters directly.

In Figure 5.25, the blocking probabilities for the three request handling policies at varying load of 500 to 1000 requests per hour are plotted. For this experiment, I assumed that 20% of the incoming requests are for 4 Mbps playback rate, 20% for Mbps, and the remaining 60% for the 16 Mbps playback rate. Because of the high percentage of 16 Mbps requests, the performance of the VoD system even under redirect policy is unacceptable.

In Figure 5.26, the 20% of the incoming traffic is assumed to be for 4 Mbps playback rate, 40% for 8 Mbps, and the remaining 40% for the 16 Mbps playback rate. Slight performance improvement is observed because of the reduction in resource-intensive class 3 requests.

In Figure 5.27, the incoming traffic is assumed to be 20% at 4 Mbps, 60% at 8 Mbps, 20% at 16 Mbps. This request mix is closer to a single rate workload at 8 Mbps. Therefore, we notice much improvement in performance. We also notice that at very low loads, there is no difference between the two split-based policies because of the absence of redirected traffic.

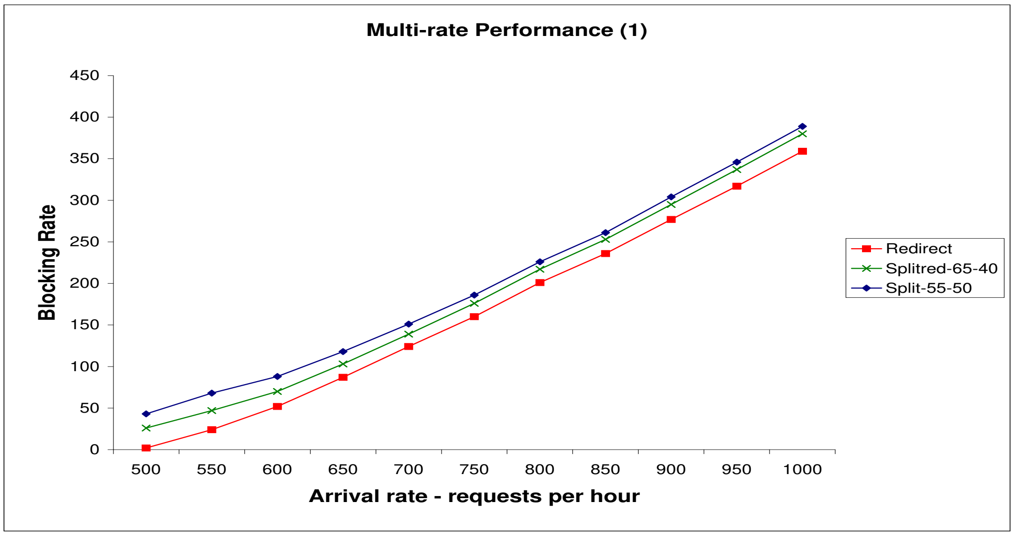

Figure 5.25: Multi-rate performance for (20,20,60) request mix

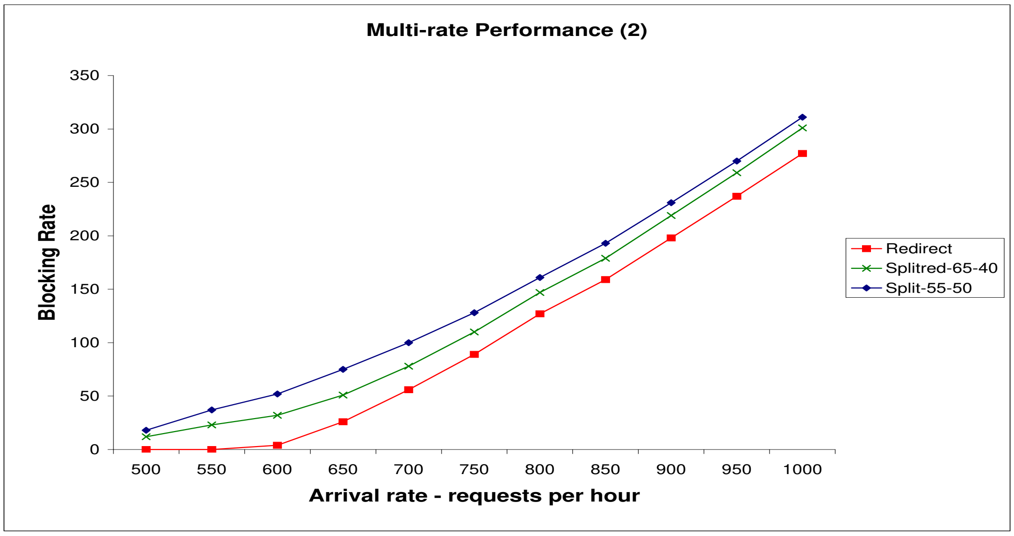

Figure 5.26: Multi-rate performance for (20,40,40) request mix

In Figure 5.28, I show class-based blocking rates for varying load at an incoming traffic of 20% at 4 Mbps as class 1 requests, 40% at 8 Mbps as class 2 requests, 40% at 16 Mbps as class 3 requests. Since each class requires different levels of resource, the blocking experienced by each class is now different. All class 3 requests experience the highest blocking because of their high bitrate requirement. As load increases, the number of blocked class 3 requests increases at a higher rate compared to the other classes.

The storage requirements for multi-rate playback service are two and a half times higher than the single rate playback service. The storage capacity available at the local cluster cannot store all videos in the video collection. Similar to the results presented in Section 5.6.4, we only have a partial set of video collection available at the local cluster. This induces higher blocking for requests for videos served only from the remote clusters. For instance, at a moderate load of 800 requests per hour under redirect policy for (20,40,40) request mix, a class 1 request for a video on the

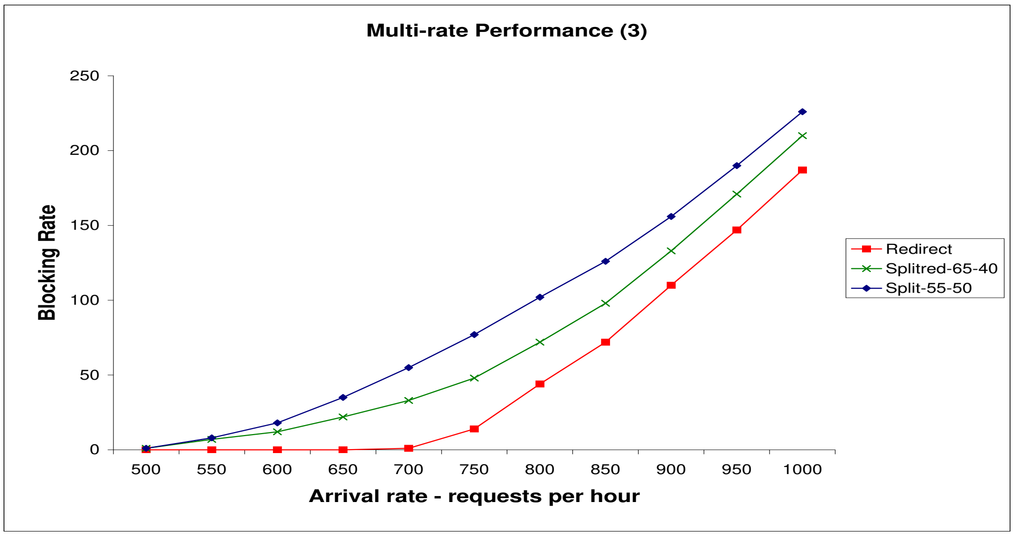

Figure 5.27: Multi-rate performance for (20,60,20) request mix

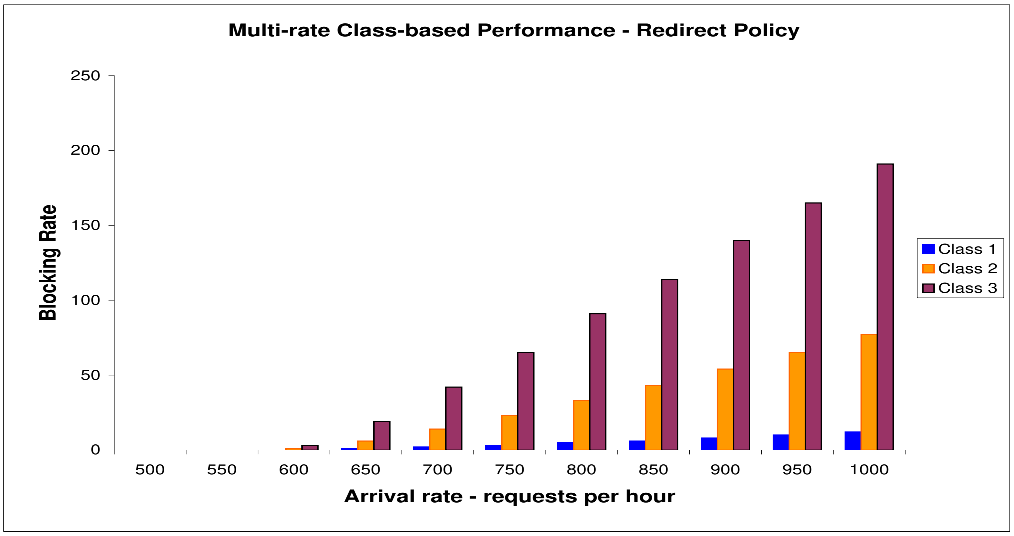

Figure 5.28: Class-based performance for (20,40,40) request mix

local cluster will experience about 2% blocking whereas another class 1 request for a video served only from remote cluster will experience about 8% blocking. We notice similar disparities for class 2 and class 3 requests for videos served on local cluster and those served only from remote clusters.

Split-based policies at splits that were efficient in the case of replicated video collection from the single-rate performance analysis now show very high blocking in Figures 5.25 to 5.27. This is because of the partial allocation of videos to the local cluster and the resulting additional traffic to the remote cluster. As mentioned in Section 5.6.4, finding efficient split-based policies requires additional information about the video distribution and the demand for videos solely served from the remote clusters. We limit further experiments in the following sections to redirect policy because of its superior performance and the difficulty in finding efficient splits.

5.8.2 Threshold Blocking Performance

In Figures 5.29 and 5.30, I show class-based blocking under class 3 threshold restrictions for request mixes (20,20,60) and (20,40,40) for a moderate load of 800 requests per hour. At 0 class 3 threshold, blocking probability for class 3 is equal to one because no class 3 requests are admitted into the system. As we relax the threshold restriction on class 3 requests, we observe that the class 3 blocking decreases but blocking for other two classes increases. Threshold restrictions are applied to only class 3 requests because they have the highest resource requirement.

The performance behavior is the same as before from Chapter 4 however, with one key difference. Because of the different resource requirements of these classes, we see that each class levels off at different blocking performances. We notice that class 3 requests at 16 Mbps rate experience more blocking than class 1 requests at 4 Mbps. Imposing thresholds on a class simply blocks a percent of the traffic from that request class and for beneficial effects to be realized on other classes there must be commensurate demands from those classes. Therefore, for a request mix of (20,20,60) blocking a large portion of class 3 requests will only marginally benefit class 1 or 2 requests.

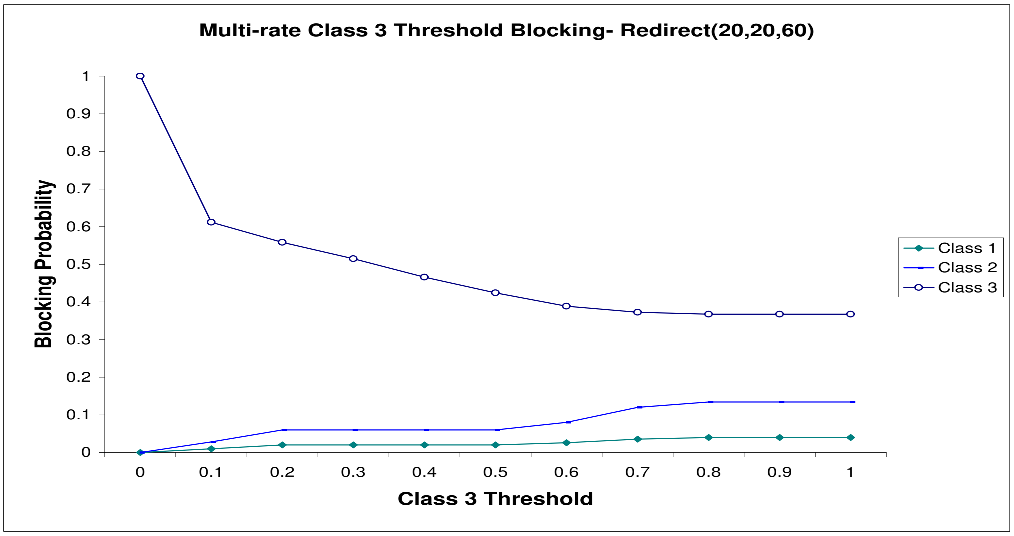

Figure 5.29: Multi-rate performance for Class 3 thresholds at (20,20,60) request mix

5.8.3 Performance with Downgrade Option

In this section, the option of downgrading a blocked request from class 2 and class 3 to a class 1 request is explored. Several scenarios where we combine downgrade option with different threshold values on class 3 requests are presented. The results are shown for a request mix of (20,20,60).

In Figure 5.31, blocking probability of the overall VoD system is shown as a function of varying load under options of downgrade with 30% threshold, downgrade with 60% threshold, downgrade with no threshold and compare them with no downgrade option and no downgrade option with threshold. Downgrade option with threshold at 30% has the least blocking because a large percentage of the blocked requests are

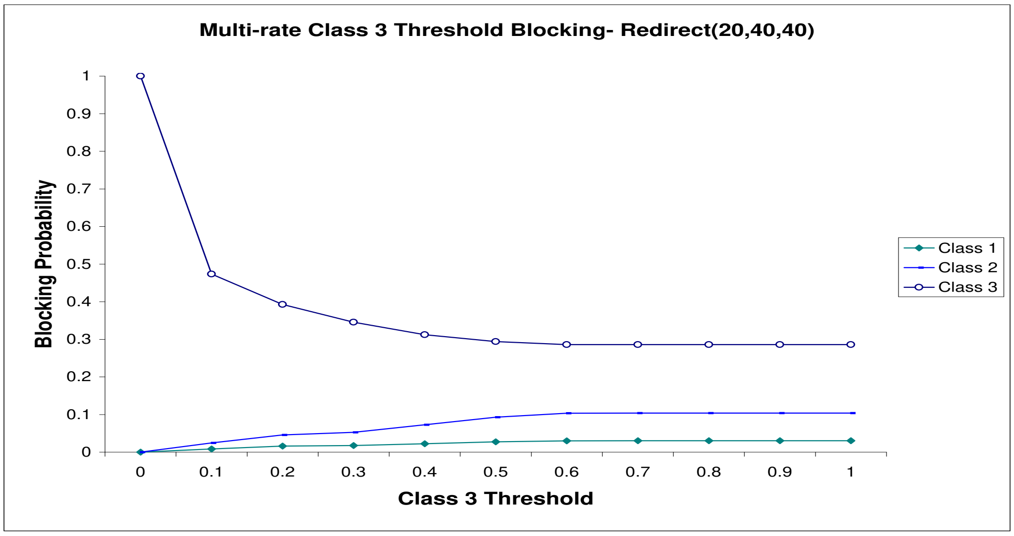

Figure 5.30: Multi-rate performance for Class 3 thresholds at (20,40,40) request mix

now converted to class 1 requests for service. If there was no downgrade option at 30% threshold for a request mix such as (20,20,60) which has predominantly class 3 requests, it simply means that a large portion of the request population was blocked with only marginal improvement in blocking performance for class 1 and class 2. This result was shown in the previous section on threshold blocking. As shown in Figure 5.31, threshold only option results in highly unacceptable blocking performance. However, threshold with the downgrade option, shows drastically improved performance (see Figure 5.31). This underscores the importance of the downgrade option for highly skewed request mixes under high threshold restrictions.

5.9 Multi-rate Performance Summary

In the previous section, I analyzed the performance of the same VoD system as in Section 5.6 with multiple playback rates and showed these following results:

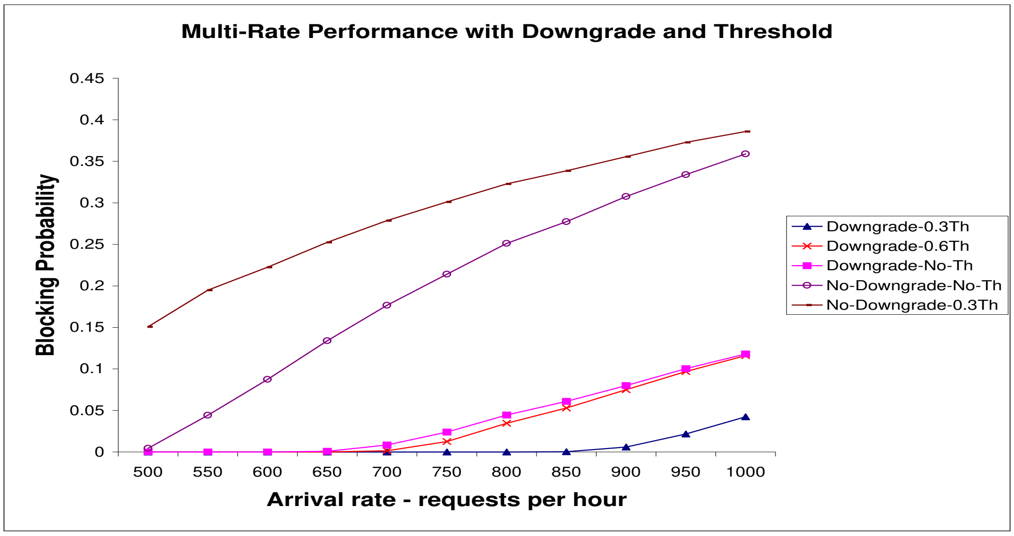

Figure 5.31: Multi-rate performance with downgrade option for (20,20,60) request mix

- Performance is unacceptable even under moderate loads for predominantly high bit rate request mixes. Redirect policy has superior performance over split-based policies. Performance improves as we lower the proportion of high bit rate class 2 and class 3 requests in the incoming traffic.

- Each class sees different blocking performance because of different resource requirements. A class 3 request has a higher probability of getting blocked than either class 1 or class 2 request. Also under threshold restrictions, each class levels off at different blocking probabilities for the same reason.

- Set of videos available on the local cluster is only a partial set from the video collection. Therefore, a request for video being served only from remote clusters will experience more blocking than a request for a video available at the local cluster. Because of the partial allocation, it is difficult to find efficient splits for split-based policies.

- Threshold restrictions on high bit rate requests only marginally improve performance for the other classes. However, threshold restrictions coupled with a downgrade option, result in drastically improved overall performance of the VoD system still providing some multi-rate service option.

- Downgrade option combined with threshold restrictions is a powerful tool for manipulating an incoming request mix over which we have no control into a workload that the VoD system can handle.

5.10 Summary

In this chapter, I presented the simulation results for a VoD system providing different QoS levels at single rate playback and multi-rate playback. The main significance of the simulation is that even limited redirection results in superior performance. I also showed that the redirect policy finds relevance under varying design considerations and has simple connection establishment semantics. The implementation overhead because of the multiple checks per blocked request for the redirect policy becomes a factor only if efficient splits are easy to determine. But finding efficient splits in dynamically shared resource environments and for unknown video distribution is difficult.

I showed that threshold-based admission control is an excellent tool for manipulating performance for some of the request classes. For multi-rate playback at the same VoD system, the performance is quite unacceptable for the same range of single rate workload. However, I showed that enforcing threshold restrictions with the option of downgrading blocked requests results in acceptable performance at the same time providing different levels of QoS.

CHAPTER 6

MODELING EFFECTS OF TRAFFIC RESHAPING ON VOD CLIENT

In this chapter, the role of the network subsystem is expanded to include effects of network delay jitter in the distributed VoD system. Typically, maximum and minimum delay bounds are associated with each network transfer. Since each transfer will possibly experience a different end-to-end delay, there will be delay jitter. Delay jitter is defined as the difference between the minimum and maximum end-to-end delay experienced by a packet. In this chapter, using the integrated framework from Chapter 3 I develop a model for analyzing the effects of network delays on client buffer requirements. In particular, I showed in the integrated model from Chapter 3 how starvation at the client can be prevented. It was shown that client playback requirements are met by providing service guarantees at the server and the network. The client buffer size was used as the amount of data transferred from the server to the client in each round in our integrated framework. This is possible only in the absence of network delay jitter. In this chapter, I explore the effects of delay jitter on client buffer requirements and present a worst-case analysis for preventing client buffer overflow. The distributed VoD system is now analyzed using a real-time producer-consumer approach with a single client as the focal point.

This chapter is organized as follows. In Section 6.1, I describe a client-centric view of the VoD system based on a real-time producer-consumer approach. In Sections 6.2 and 6.3, I develop a worst-case analysis of client buffer requirements under the integrated model assumptions from Chapter 3. In Section 6.4, two classes of network service disciplines with reference to their traffic reshaping capabilities are discussed. I present numerical results and summarize the significance of the analysis provided in this chapter in Sections 6.5 and 6.6.

6.1 Real-time Producer-Consumer Approach

As seen from Section 3.1 the VoD architecture consists of three critical subsystems: a real-time producer of synchronized multimedia data, a communication network, and a consumer of real-time synchronized multimedia data (see Figure 6.1). Here, this architecture is modeled as a real-time producer-consumer system using a single client's view. In this framework, the server and the network act in unison to provide seamless transfer of data for client's playback requirements. The server periodically retrieves video data in *rounds* and sends it over the network. To provide guaranteed service at the server, it is critical to maintain a strict time bound for retrieval of data. By guaranteeing a reserved rate at the network that is at least as large as the playback rate, the same time bound is maintained for network transfer. In addition to the reserved rate a maximum delay bound has to be guaranteed at the network. These are the minimum requirements to provide starvation-free service at the client as we saw from the analysis provided in Chapter 3. To provide loss-free service we need to prevent buffer overflows at the client. Using the proposed jitter control model, client buffer requirements are analyzed in presence of network delay jitter.

6.2 Analytical Model

In this section, a worst case analysis of buffer requirements at the client's set-top box is developed. I develop client buffer requirements in the context of a general-purpose model of the VoD system using a double buffer scheme at the server and rate-based scheduling at the network that can at least provide a maximum end-to-end delay bound.

A transmission schedule is defined based on the server model and the periodic data

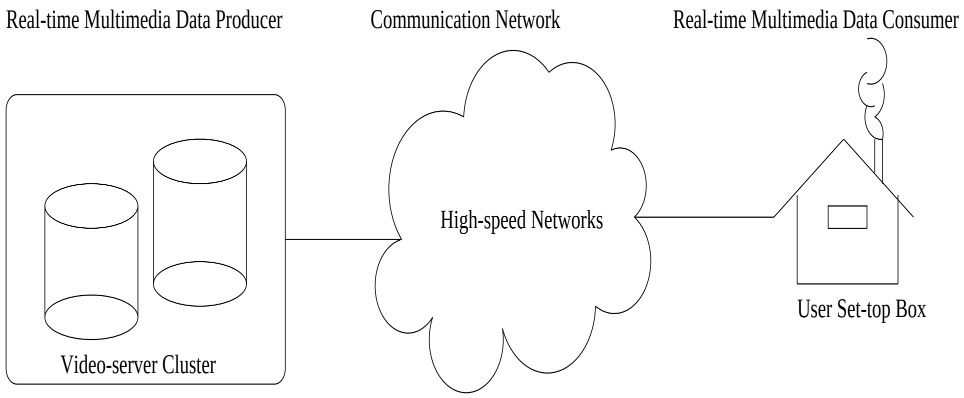

Figure 6.1: Real-time producer-consumer model of VoD system: single client's view

transfer from the server using the double buffer scheme. In a transmission schedule, a B sized message (which I termed retrieval block in Chapter 3) is transmitted from the server every round of length T. T_s is a round in the transmission schedule. The playback schedule at the client is defined taking into consideration the delay jitter in the network. The playback begins at the end of the maximum end-to-end delay, D_{max} after the first round at the server is completed. Playback schedule begins at the client at the end of D_{max} after round 1 of the transmission schedule, T_s^1. Each playback round T_r in a playback schedule lasts the same length as a T_s which is equal to T. The B sized message arrives at the client according to the network performance parameters, the minimum end-to-end delay, D_{min} and the maximum end-to-end delay, D_{max} (see Figure 6.2).

A brief explanation on how to read the figures in this chapter is given here. The line with the positive slope with height B indicates the arrival of a message. The slope is negligible as instantaneous arrival of the message is assumed to determine worst-case buffer requirements. The line with the negative slope, R_p indicates the consumption of the B sized message during a playback round. The horizontal axis refers to the time line, whereas the vertical axis refers to the amount of data.

Individual packets from the B sized message arrive instantaneously at the client buffer at the end of D_{min}. This assumption provides for a worst-case analysis of buffer

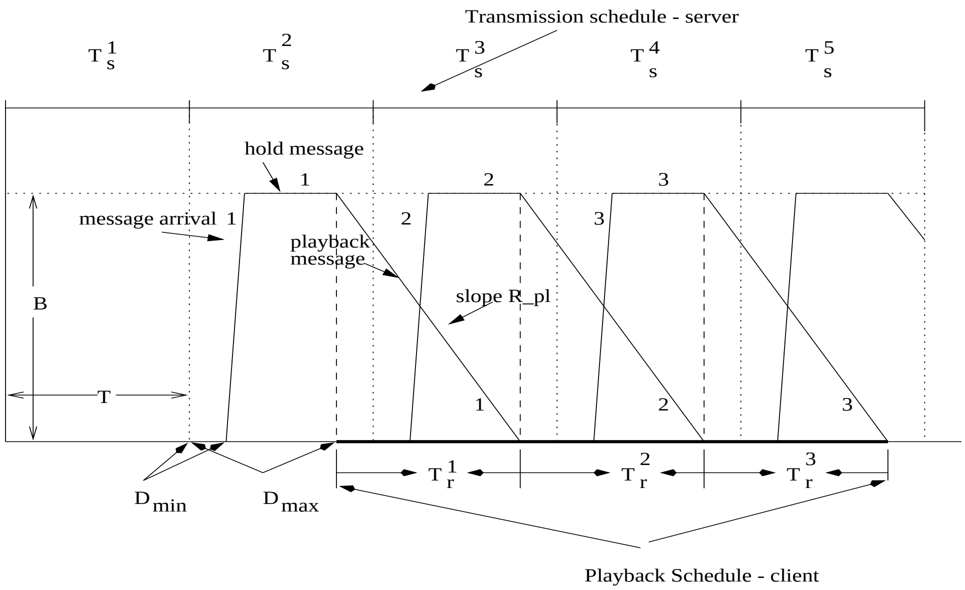

Figure 6.2: Transmission and playback schedule

requirements. Individual packet arrival or consumption pattern on a continuous time scale are not modeled. Rather, the granularity of the analysis is based on message sizes of B units and time lengths equal to a round length, T.

Next, I prove a Theorem to show that the relationship between delay jitter and the amount of data in the buffer is linear. The maximum data in the receiver buffer in a playback round is computed as the difference between the total number of arrived messages of size B and the data consumed. Model parameters are shown in Table 6.1.

Recalling from the integrated model in Chapter 3 that the round length, T at the server is defined as the playback duration of a message of size B, at a rate R_p at the client for single rate (for multi-rate, the definition of round length is based on the minimum buffer to playback rate ratio among client classes). Therefore,

$$T = \frac{B}{R_p} \tag{6.1}$$

The maximum data in the client buffer in a playback round, B_T is computed as:

Table 6.1: Model parameters for client buffer requirements

B	Max. data from the server in a round, Message size or Retrieval block size
B_r	Size of client buffer
B_T	Maximum data in the client buffer in a playback round of length T
$B_{consumed}$	Data consumed in a playback round up to the latest message arrival
R_p	Playback rate of request at the client
T	Round length at the server
D_{max}	Maximum end-to-end delay
D_{min}	Minimum end-to-end delay
D_{jitter}	Delay jitter, $D_{max} - D_{min}$
N	Total number of messages of size B in the time interval $D_{max} + T$

$$B_T = NB - B_{consumed} \tag{6.2}$$

The values of D_{max} and D_{min} are represented as $n_1T < D_{max} \leq (n_1 + 1)T$ and $n_2T < D_{min} \leq (n_2 + 1)T$ respectively where n_1 and n_2 are nonnegative integers and $n_2 \leq n_1$ since $D_{min} \leq D_{max}$. For $n_1 = n_2 = 0$, the value of D_{max} and D_{min} lies between 0 and T.

Equation (6.3) shows the number of messages arriving in the time interval $D_{max} + T$. The ceiling function represents the number of messages which have arrived before the playback begins. As seen from Figure 6.2, there will always be one message arriving per playback round. Of the total number of messages, N, notice that one of them is currently being consumed at a rate R_p.

$$N = \left\lceil \frac{D_{max} - D_{min}}{T} \right\rceil + 1 \tag{6.3}$$

Theorem 1 *The maximum data, B_T in the receiver buffer in a playback round of* *length T is* linear *in the delay jitter, D_{jitter}.*

$$B_T = B + D_{jitter} R_p \tag{6.4}$$

Proof of Theorem 1:

For purposes of clarity in presentation, I present a proof for two cases: Case 1 deals with values of $D_{jitter} > T$ and Case 2 with $D_{jitter} \leq T$.

Case 1: $D_{jitter} > T$

The relative positions of D_{max} and D_{min} in a transmission schedule round determines whether the new message arrives before or after the playback begins. If the arrival occurs before the playback begins, there is one more message arrival than if the new message occurs after the playback begins. Based on these relative positions of D_{max} and D_{min} in a T_s (see Figure 6.3), two conditions are defined. Using one of these conditions as appropriate, we can then compute the data consumed in a playback round, T_r. The difference between the two conditions is that in Condition 2, there is an additional message arrival before the playback begins.

Basically, the idea behind the two conditions is to determine the time interval for which the data has been consumed. The time interval begins at D_{max} since the playback begins at D_{max} and ends at the arrival of the new message which occurs at D_{min}. In Condition1, the relative position of D_{max} is before D_{min} in a transmission round. Taking the beginning of the transmission round as the reference point, the time interval for which data has been consumed is the difference between the distance to D_{min} (from the beginning of the transmission round, $D_{min} - n_2 T$) and the distance to D_{max} (that is, $D_{max} - n_1 T$). This is illustrated in Figure 6.3a. Similar logic explains Condition 2 and is illustrated in Figure 6.3b.

Condition 1: $(D_{max} - n_1 T) \leq (D_{min} - n_2 T)$

$$B_{consumed} = R_p((D_{min} - n_2 T) - (D_{max} - n_1 T))$$

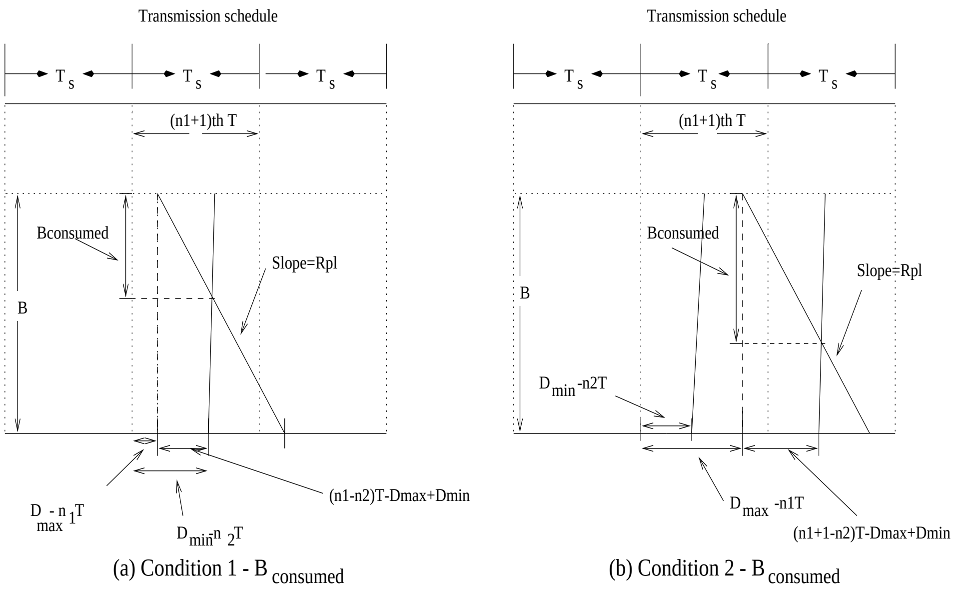

Figure 6.3: Conditions for computing $B_{consumed}$

Substituting for R_pT from (6.1) and for $(D_{max} - D_{min})$ gives,

$$B_{consumed} = (n_1 - n_2)B - D_{jitter}R_p \tag{6.5}$$

$(n_1 - n_2)$ in (6.5) is the number of rounds prior to playback where there will be an arrival of B sized message. This is the same as $\lceil \frac{D_{max}-D_{min}}{T} \rceil$. Therefore, the total number of messages, N is given by, $N = n_1 - n_2 + 1$.

Substituting for N and $B_{consumed}$ in (6.2) we have,

$$B_T = (n_1 - n_2 + 1)B - ((n_1 - n_2)B - D_{jitter}R_p)$$

or

$$B_T = B + D_{jitter}R_p \tag{6.6}$$

and the Theorem is proved for Condition 1.

<u>Condition 2:</u> $(D_{max} - n_1T) > (D_{min} - n_2T)$

$$B_{consumed} = R_p(((n_1 + 1)T - D_{max}) + (D_{min} - n_2T))$$

Substituting for R_pT and $D_{max} - D_{min}$ as in Condition 1, gives

$$B_{consumed} = (n_1 + 1 - n_2)B - D_{jitter}R_p \tag{6.7}$$

As in Condition 1, $(n_1 + 1 - n_2)$ is the total number of messages prior to playback round. After adding for the additional message arrival *during* the playback round, we have $N = (n_1 + 1 - n_2) + 1$.

Substituting for N and $B_{consumed}$ in (6.2) we have,

$$B_T = (n_1 - n_2 + 2)B - ((n_1 + 1 - n_2)B - D_{jitter}R_p)$$

After simplification we have the same result as in (6.6) and the Theorem is proved for Condition 2.

Case 2: $D_{jitter} \leq T$

Delay jitter, $D_{jitter} \leq T$ implies that $n_1 = n_2 + 1$ or that $n_1 = n_2$. The number of messages of size B arriving in any round length cannot be more than 1 because the message is produced only every round at the server. Playback schedule begins with a full buffer of B data and, during its playback, another message of size B arrives before the end of the current playback round (see Figure 6.4). Therefore, N, the number of messages arriving in $D_{max} + T$ is given by: $N = 2$.

To compute $B_{consumed}$, we define the same conditions as in Case 1.

Condition 1: $(D_{max} - n_1T) \leq (D_{min} - n_2T)$

Since $D_{jitter} \leq T$, this condition is possible only when $n_1 = n_2 + 1$. Substituting for n_1 in (6.5) reduces the expression for $B_{consumed}$ to the following.

$$B_{consumed} = B - D_{jitter}R_p \tag{6.8}$$

Condition 2: $(D_{max} - n_1T) > (D_{min} - n_2T)$

Since $D_{jitter} \leq T$, this condition is possible only when $n_1 = n_2$. Substituting for n_1 in (6.7) reduces the expression for $B_{consumed}$ to the same as (6.8).

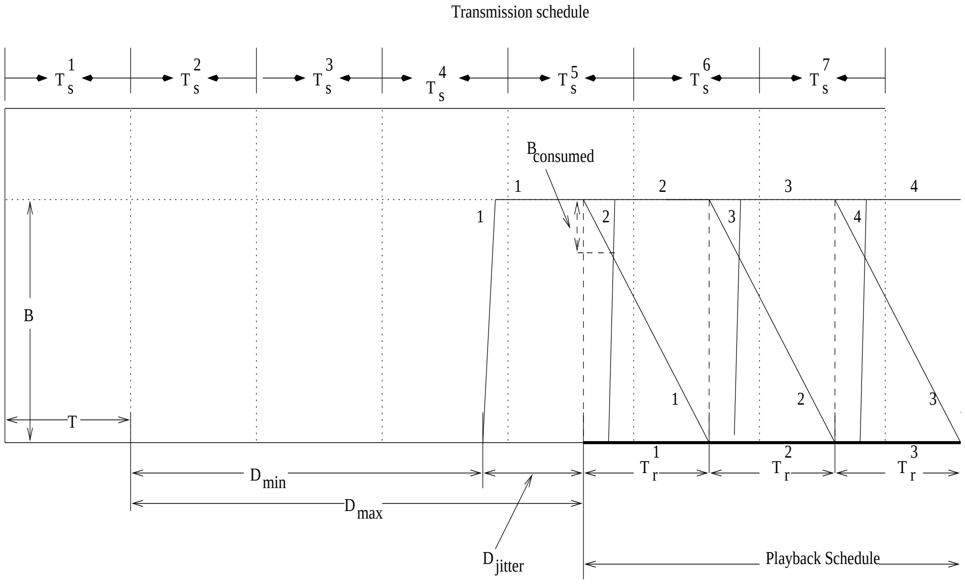

Figure 6.4: Buffer requirement for $3T < D_{max} < 4T$ and $2T < D_{min} < 3T$ and $D_{jitter} < T$

The maximum data in any round during a playback schedule is computed using (6.2) after replacing for N and $B_{consumed}$:

$$B_T = 2B - (B - D_{jitter}R_p)$$

or

$$B_T = B + D_{jitter}R_p \tag{6.9}$$

Equations (6.6) and (6.9) prove Theorem 1 for $D_{jitter} > T$ and $D_{jitter} \leq T$. **QED**

6.3 Client Buffer Overflow and Underflow

In this section, necessary conditions to prevent overflow and underflow are discussed. I then present examples using ranges of values for D_{max} and D_{min}. In all examples, the same procedure presented in the proof for Theorem 1 is followed to derive the expression for maximum data in the receiver buffer in a playback round. All examples show that the maximum data in the receiver buffer is linear in delay jitter.

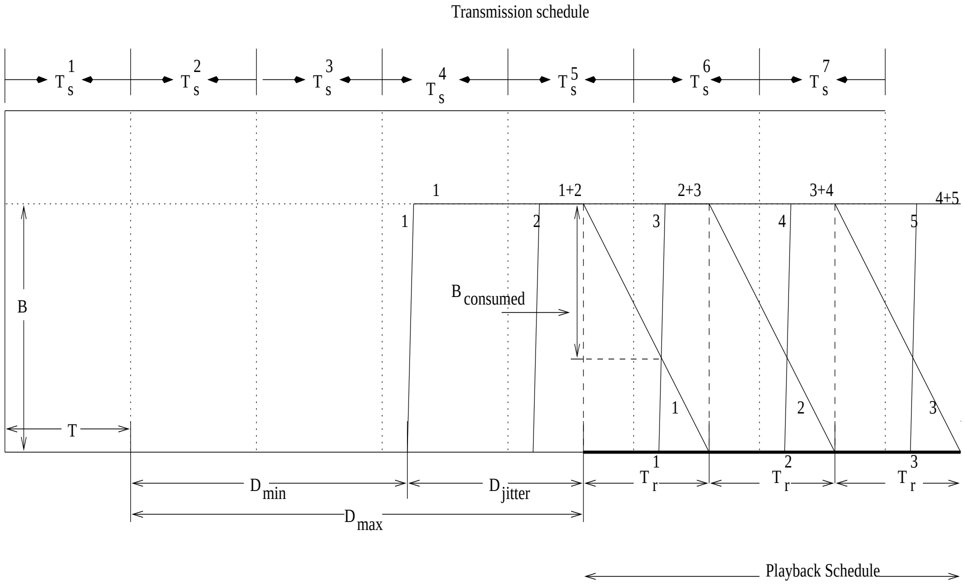

Figure 6.5: Buffer requirement for $3T < D_{max} < 4T$ and $2T < D_{min} < 3T$ and $D_{jitter} > T$

Client buffer overflow is prevented when the physical buffer size, B_r is larger than the maximum data in a playback round, B_T, that is when $B_r \geq B_T$. Starvation occurs when the arrival of the new message is delayed beyond D_{max}. As long as there is a network service discipline that can provide guaranteed rates and a scheduling mechanism at the network which provides maximum bounded delays, there will never be an underflow of the client buffer assuming that we delay the start of playback by the maximum end-to-end delay.

B_T for $D_{min} > T$

Figure 6.5 illustrates buffer requirement for $3T < D_{max} < 4T$ and $2T < D_{min} < 3T$ and $D_{jitter} > T$. $B_{consumed}$ is computed as $2B - D_{jitter}R_p$ using Condition 2 above by replacing for $n_1 = 3$ and $n_2 = 2$. The maximum data in a playback round for the example shown is $B_T = 3B - (2B - D_{jitter}R_p)$ which reduces to $B_T = B + D_{jitter}R_p$.

B_T for $D_{min} < T$

Since $D_{min} < T$, $n_2 = 0$. The total number of messages, N is given by (6.3).

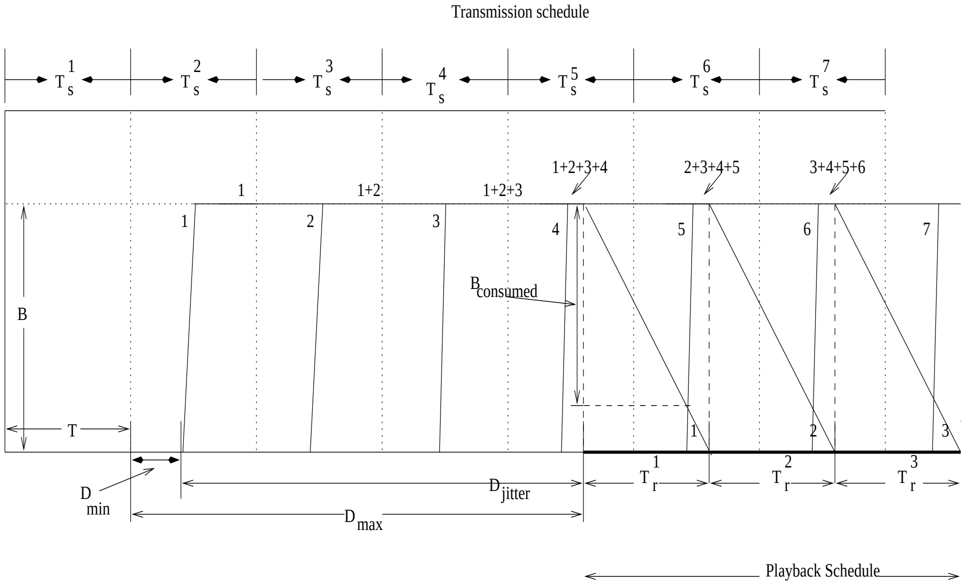

Figure 6.6: Buffer requirement for $3T < D_{max} < 4T$ and $D_{min} < T$ and $D_{jitter} > T$

$B_{consumed}$ is computed by replacing $n_2 = 0$ in (6.5) or (6.7) as appropriate and B_T is computed using (6.2). Figure 6.6 illustrates buffer requirement when $D_{min} < T$ and $3T < D_{max} < 4T$ and $D_{jitter} > T$. Using (6.3), we get $N = 5$. $B_{consumed}$ is computed using Condition 2 and is equal to $4B - D_{jitter}R_p$. The maximum data in a playback round, $B_T = 5B - (4B - D_{jitter}R_p)$ which reduces to (6.9).

<u>B_T for $D_{max} < T$</u>

$D_{max} < T$ implies $D_{min} < T$, and $n_1 = 0$ and $n_2 = 0$. Also, the condition $D_{max} < T$ implies that $D_{jitter} < T$. $D_{jitter} \leq T$ limits the number of message arrivals to $N = 2$. The playback of one of those arrivals is already in progress as the second message arrives. Since $D_{max} > D_{min}$, only Condition 2 from Case 1 is applicable. Data consumed during the playback round is computed by replacing for n_1 and n_2 in (6.7) and is the same as (6.8). The maximum data in any round during a playback schedule is the same as (6.9).

<u>B_T for $D_{jitter} = T$</u>

The total number of messages within a playback round is $N = 2$ as in Case 2. However, since $D_{jitter} = T$, the playback of the current round begins at about the same time as the new message arrives. Therefore, $B_{consumed} = 0$ which can be seen by substituting for $D_{jitter} = T$ in (6.8). A simpler derivation is to replace for D_{jitter} in (6.9). The maximum data in the receiver buffer therefore, is given by $B_T = 2B$.

$\underline{B_T \text{ for } D_{jitter} = 0}$

For the case of $D_{jitter} = 0$, we have a situation where the playback of the data begins as the new arrival occurs. This is similar to a just-in-time situation and the maximum data in the receiver buffer is given by $B_T = B$. Again replacing for $D_{jitter} = 0$ in (6.9) gives the same result.

6.4 Choice of Network Service Disciplines

The choice of a network service discipline is fundamental to providing loss free and starvation free service to clients. In this section, I discuss the merits of two classes of service disciplines: rate allocated and rate controlled disciplines. The significance of that discussion is summarized in the following paragraph.

As a general rule, reducing delay jitter reduces buffer requirements at the client; otherwise, worst-case often extreme buffer sizes have to be provided. Another interesting aspect of delay jitter control emerging from the proposed modeling context is that we can accommodate clients with different buffer sizes using the same server transmission schedule. Clients with smaller buffer sizes will request tighter delay jitter controls. This will ease the burden on the network by dispersing different QoS levels to clients.

Though scheduling schemes such as WFQ (Parekh and Gallager 1993; Parekh and Gallager 1994) or Delay-Earliest-Due-Date (Delay-EDD) (Ferrari and Verma 1990) based on rate-allocated service disciplines provide bounds on bandwidth and maximum end-to-end delays, they do not provide a non-trivial bound on the minimum delay. A non-trivial bound on minimum delay includes some queueing delay in addition to the propagation and transmission delays. Since there are no guarantees on the minimum delay bound, there is *potential* for large delay jitters. The buffer requirements can be close to Case 1 in proof of Theorem 1 where the start of the playback is deferred until D_{max}. Because of the conservative assumption on D_{min} and a large D_{max} resulting in a large delay jitter, buffer requirements can be extremely high (see Figures 6.5 or 6.6). For instance, upper bound on buffer requirements in Figures 6.5 and 6.6 is $3B$ and $5B$ respectively.

To reduce buffer requirement at the client, we need non-trivial bounds on delay jitter. Network scheduling schemes such as Jitter-Earliest-Due-Date (Jitter-EDD) (Verma, Zhang, and Ferrari 1991) or Rate-Controlled Static Priority (RCSP) (Zhang and Ferrari 1993) based on rate-controlled disciplines provide non-trivial bounds on delay jitter.

Basically, rate controlled disciplines reshape the traffic at each network element through the use of a regulator and a scheduler which will in turn reduce the delay jitter. The reshaping is done by holding those packets that arrive ahead of their deadline until they are eligible for transmission. The regulator is responsible for computing the eligibility times per packet and holding them until they are eligible for transmission by the scheduler. Under Jitter-EDD, the eligibility time per packet is found as follows (Zhang 1995):

$$e^k_{i,j} = a^k_{i,j} + Ahead^k_{i-1,j} \tag{6.10}$$

where $e^k_{i,j}$ is the eligibility time and $a^k_{i,j}$ is the arrival time for k^{th} packet from j^{th} flow on i^{th} switch and $Ahead^k_{i-1,j}$ is the amount of time the packet is ahead of schedule at the $(i-1)^{th}$ switch. All packets receive the same delay on all but the last link. The delay jitter for the flow is the delay jitter on the last link.

Two corollaries are presented next, regarding the two approaches to network delay jitter control discussed above. Corollary 1 highlights the fact that there is no jitter control under WFQ and we have to provide for worst-case scenario. Corollary 2 shows that the option of specifying desired jitter bound results in smaller bounds for client buffers.

6.4.1 Weighted Fair Queueing

Corollary 1 *If the trivial delay jitter bound is greater than* n *times the round length (but less than* $(n+1)$ *times the round length),* $D^{WFQ}_{jitter} > nT$ *where* $n \geq 1$*, under Weighted Fair Queueing (WFQ) network scheduling, the worst-case upper bound on buffer requirements,* B_r *is* $(n+2)B$*.*

Proof

Using WFQ, the maximum delay bound per packet for a flow with a guaranteed rate which is at least as large as the playback rate, R_p and routed on a path that has J links is given by (Parekh and Gallager 1993; Parekh and Gallager 1994):

$$D_{max} = \frac{B}{R_p} + \frac{(J-1)M}{R_p} + \sum_{j=1}^{J} \frac{M_{max}}{A_j} \tag{6.11}$$

where M is the maximum size of the packet for the flow, M_{max} is the maximum size of a packet permitted in the network (MTU), and A_j is the overall bandwidth on link j. D_{max} is longest when reserved rate at the network is equal to the playback rate, R_p. Ignoring transmission delays and based on the network characteristics, the second and third terms in (6.11) related to queueing delays could conceivably be several times larger than a round length, T making the worst case end-to-end delay bound very large. Notice that WFQ does not guarantee a firm minimum end-to-end delay bound. Given a maximum end-to-end delay bound, we can find a trivial delay jitter bound by making optimistic assumptions on minimum end-to-end delay such as no queueing or transmission delays. As a result, the delay jitter can be very large for large maximum end-to-end delays. As shown in Corollary 1, we can only derive *worst case* bounds for buffer requirements under WFQ.

6.4.2 Jitter-EDD

Corollary 2 *If using Jitter-EDD, the delay jitter is controlled to less than or equal to a round length, $D_{jitter}^{J-EDD} \leq T$, the upper bound on buffer requirements, B_r is $2B$.*

Proof

Since Jitter-EDD provides non-trivial delay jitter bounds, a request can receive a jitter bound that is less than the round length. As seen from our analytical model, the upper bound on buffer requirements is twice the message size. Because of jitter control, we can provide smaller bounds on buffer requirements.

6.5 Numerical Results

In this section, the extent of gain achieved through delay jitter control is shown. The numerical examples highlight the trade-offs involved in this analysis. From a server point of view, large message sizes result in optimal retrieval times at the disks. However, we find that in presence of large delay jitter, that larger messages will have greater buffer requirements at the client. Controlling delay jitter by using appropriate service disciplines to reduce buffer size, will entail network complexities in implementing those disciplines. If the tradeoff is between memory costs and network complexities, we should consider the fact that the context of this analysis should include mass market issues of the client set-top boxes in judging the benefits of delay jitter control.

There is a one-to-one mapping between the buffer size at the client and the delay jitter at the network. Inverting the result from Theorem 1, we derive D_{jitter} allowed on the network given a buffer size of B_r.

$$D_{jitter} = (B_r - B)/R_p \tag{6.12}$$

To achieve a desired delay jitter bound under Jitter-EDD, minimum delay bound could be used as a flow specification parameter when the connection is set up.

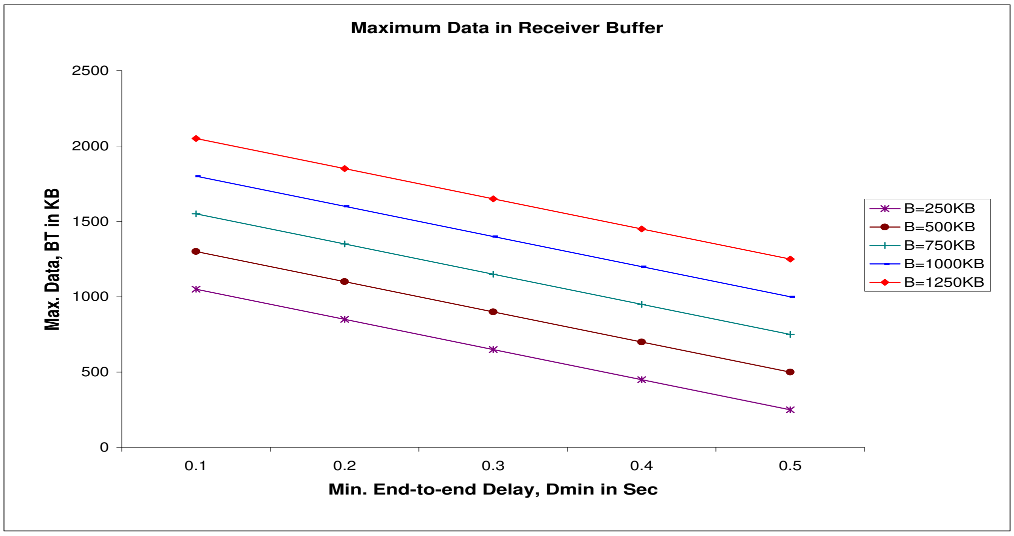

Figure 6.7: Jitter-EDD, increasing D_{min} bound

The minimum delay bound being used as a flow specification parameter has other important consequences. Imagine a set of heterogeneous client set-top boxes with different buffer sizes. Appropriate minimum delay bounds based on the buffer sizes for each client could be specified at the time of request set up. Clients with smaller buffer sizes can now request tighter jitter control. The same server transmission schedule of a certain message size and round length can now accommodate heterogeneous clients with different buffer sizes.

In Figure 6.7, we show how different minimum delay bounds affect buffer requirements. The maximum end-to-end delay is fixed at 0.5 sec. The minimum end-to-end delay is varied from 0.1 sec to 0.5 sec. The delay jitter range is progressively reduced from 0.4 sec to 0 jitter. Reducing jitter results in reduced buffer requirements. At minimum end-to-end delay of 0.1 sec, buffer requirements are the highest because the delay jitter is 0.4 sec. The smallest message size of 250 Kbytes has the least buffer

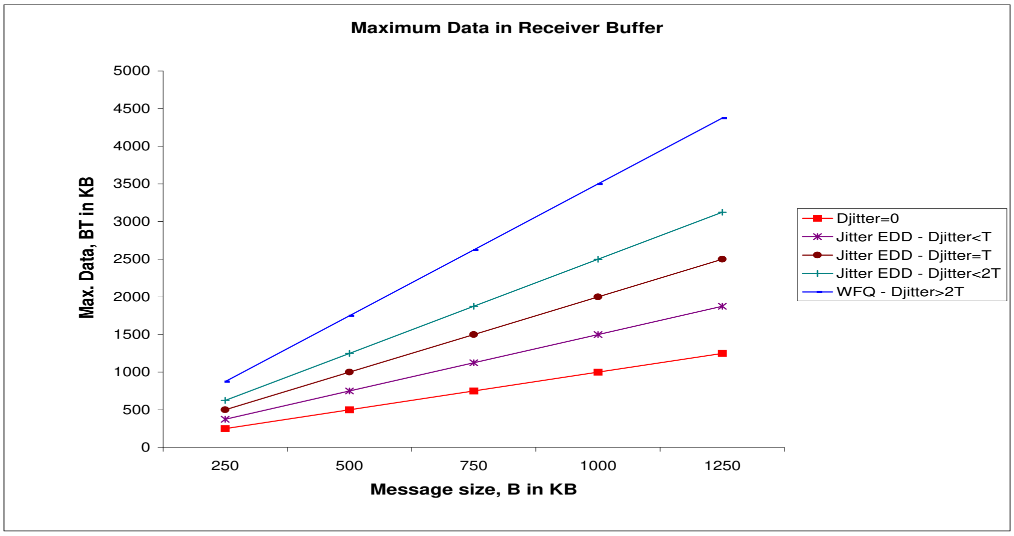

Figure 6.8: WFQ vs. Jitter-EDD

requirement.

In Figure 6.8, WFQ is compared with Jitter-EDD on client buffer requirements for various message sizes from 250 Kbytes to 1250 Kbytes. The playback rate is kept constant at 16 Mbps. The round length varies with the message size. The delay jitter is greater than $2T$ for WFQ. Since there is no jitter control under WFQ, we see that the buffer requirements are the highest for WFQ. Using Jitter-EDD we are able to control delay jitter and the buffer requirements for different delay jitters are shown. For $D_{jitter} = 0$, the buffer requirements are the same as message sizes. Jitter-EDD with $D_{jitter} < T$ has the lowest buffer requirements ranging from 375 Kbytes to 1875 Kbytes for message sizes of 250 Kbytes and 1250 Kbytes.

6.6 Summary

The significance of the analysis presented in this chapter is that I developed a methodology to understand the impact of different network performance parameters under a server transmission schedule most suitable for multimedia data. The role of the network subsystem in the VoD architecture is extended by considering the impact of different traffic scheduling disciplines. I used the integrated model from Chapter 3 of server operation based on round scheduling and network scheduling based on rate-based schemes to develop an analysis of client buffer requirements under such a scenario. I proved that the maximum data in the client buffer in a playback round is linear in delay jitter.

Using a client centric approach, I developed a methodology based on a real-time producer-consumer approach to analyze buffer requirements for a client in presence of network delay bounds and delay jitter bounds. As a general rule, reducing delay jitter reduces buffer requirements. But the analysis I have provided here completes the cycle of end-to-end analysis of the VoD system by combining server transmission schedule with network transfer mechanism to derive client buffer requirements. I showed through numerical analysis that controlling delay jitter not only reduces client buffer requirements but provides the option of accommodating heterogeneous clients with different buffer sizes for the same transmission schedule.

CHAPTER 7

CONCLUSIONS

7.1 Summary of Contributions

In this dissertation, I filled a void in distributed multimedia systems research by providing a system-wide analysis of a VoD system. I demonstrated that a VoD system must be analyzed as an end-to-end system to maximize resource utilization. Through analytical modeling and simulation, I showed that it is possible to analyze the overall performance of a distributed VoD system in an integrated framework.

I developed a unified model of a distributed VoD system by integrating the storage and the network subsystems. I showed that an end-to-end analysis is needed especially in the context of high quality VoD system where guaranteed service on all subsystems is required. The server model uses a double buffer scheme and is independent of disk scheduling schemes. The network model is also general-purpose but assumes the existence of a traffic envelope generated from a regulator at the server acting as the packet source. Because of the traffic envelope it is possible to determine maximum end-to-end delay for a given reserved rate at the network. In the network model, a reserved rate that satisfies a given desired bounded delay using Weighted Fair Queueing (WFQ) scheduling scheme is computed. Admission control conditions at the server and the network are derived.

I implemented the integrated admission control model in the simulation for a distributed VoD architecture consisting of local and remote clusters. In the simulation, I analyzed the performance of the VoD system under two major service options: single

QoS level and multiple QoS levels. I evaluated the performance of the VoD system for request handling policies based on redirection for blocked requests and on load sharing ideas (Mundur, Simon, and Sood 1999a; Mundur, Simon, and Sood 1999b). In the analysis, I examined several design options such as replicated video collection on all clusters versus partial video collection on local cluster and non-blocking versus shared remote clusters.

Simulation results showed that even limited redirection brings about superior performance. Policies based on redirection have simple connection establishment semantics, meaning that the blocked request will be routed to the next resource in the hierarchy. However, there is an implementation overhead because of the multiple checks performed per blocked request. On the other hand, split-based policies do not carry the overhead of multiple checks for blocked requests but require information on the resource levels available to find an efficient split. If the resource is dynamically shared by other user populations, it is not possible to know the available resource level ahead of time which makes it difficult to find efficient splits. The simulation results showed that policies based on redirection and splitting find relevance only in the context of replicated video collection and non-blocking remote clusters. In all other cases, redirection-based policy is the answer. Otherwise, the performance can easily deteriorate with inefficient splits as the results show.

Simulation results showed that by storing only a partial set of most popular videos from the video collection on the local cluster, the overall blocking probability for the VoD system is reduced. However, the disadvantage with partial allocation is that the videos being served only from the remote clusters experience higher blocking because of reduced stream handling capacity available to them. Partial allocation also makes it difficult to find efficient splits for the split-based request handling policies.

Operating a multi-rate service using the same VoD system resulted in unacceptable performance even under redirect policy. However, imposing threshold restrictions on requests for the highest bit rate service and downgrading a blocked request to lowest bit rate service resulted in good performance. In fact, I showed that a downgrade option coupled with threshold restriction is a powerful tool for transforming a request workload to suit the VoD system capacities.

I presented an analytical model for evaluating the performance of a VoD system resource such as a video server under threshold-based admission control and developed an efficient recursive method to compute blocking probabilities (Mundur, Sood, and Simon 1998). The analytical results for threshold-based admission control were validated with the simulation results. I showed that threshold-based analysis gives us a way of assigning priorities to request classes. More resources are provided to the higher priority request classes by restricting access to lower priority classes. The more popular videos experienced reduced blocking at the cost of increased blocking for the less popular videos because of threshold restrictions. For the multi-rate service, I showed that thresholds with downgrade option resulted in superior performance for the VoD system.

Client buffer requirements are an important consideration in VoD design. I proved that the buffer requirements at the client can be reduced if the underlying network service discipline provided non-trivial delay jitter bound (Mundur, Sood, and Simon 1999). To achieve a desired delay jitter bound, I showed that the minimum delay bound can be used as a flow specification parameter when the connection is set up. Clients with smaller buffer sizes can now specify higher minimum delay bounds to obtain tighter jitter control. The same server transmission schedule will now be able to accommodate heterogeneous clients with different buffer sizes.

7.2 Future Research

In this section, I present several ideas for extending the current research.

Workload Characterization: Proposed analytical and simulation models are based on Poisson process for request arrivals from a single reference user population. An interesting extension to this study is to build a realistic workload model especially incorporating time of day factors from multiple user populations. For instance, one user population which is predominantly a retirement community might provide heavy workload during the day at their local or remote clusters. The same resources may be shared by some other user population during evening or night. Performance may be manipulated using threshold restrictions on requests from one user population against another. I have already shown in this dissertation how priorities can be assigned to a class of requests using threshold restrictions.

Characterizing the workload is important if interactivity is included in the VoD system. Providing interactive service will enable a client to perform VCR-like functions such as rewind, fast forward, pause, and resume. Not only should the additional load be correctly predicted because of interactivity but also server and network data transmission schemes have to be modeled differently. Realistic workload data may be difficult to find but synthetic models may be developed by making assumptions about user behavior.

CBR versus VBR: In this dissertation, I have assumed constant bit rate (CBR) transfers and playback. Extending the ideas presented here to variable bit rate (VBR) transmission and display is natural. The bit rate for VBR transmission and display will depend on the scene content of the video and will change from scene to scene, and video to video and we need to assume appropriate distribution for the bit rate over a time period. Using VBR assumptions for a large video collection can however, get complicated.

Inclusion of Data Sharing Options: Now that we have a performance evaluation scheme for a distributed VoD system based on dedicated resources for each request stream, it will be interesting to extend it to include data sharing options, such as batching or merging and examine the improvement in performance. However, the unsaid assumption in all data sharing options is that there will be requests for the same video in quick time succession. With a highly skewed distribution such as Zipf and 150 videos in the collection, and at a high rate of arrival at 1000 requests per hour, we have three requests for the same video per minute. We may be able to achieve some improvement in performance by assigning a single stream to multiple requests. However, as we increase the videos in our collection, we will have fewer requests for the same video that can be batched within a reasonable response time. Inclusion of interactivity in a data sharing environment is another challenge for future research.

Dynamic Sharing of Networks and Clusters: In the simulation, the remote clusters had certain capacity dedicated to the reference user population. Partitioning the remote cluster capacity among user populations need not necessarily result in good performance. Previous resource allocation models have suggested that complete sharing of resources is found to result in better performance. Therefore, an interesting extension to the proposed simulation is to include dynamic sharing of remote clusters from multiple user populations. We can even include some hops in the network path to be shared by other user populations. This will result in more resource contention.

Dynamic Updates with Replication: Finally, in this dissertation I used a static picture of the VoD system for evaluating its performance. Given a certain video collection, the video distribution is based on the replication and placement algorithms. Based on that allocation and demand, I evaluated the performance of the VoD system. A good extension is to develop an algorithm for updating the video collection and even examine dynamic replication schemes. Closely related to this idea is the topic of crash recovery and fault tolerance. Dynamic replication and request migration and even some of the data sharing options will provide answers to fault tolerance and future research in that direction is much needed.

APPENDIX

APPENDIX

PROOF OF THEOREM 1 - THRESHOLD ANALYSIS

Let $\overline{e_k}$ be $(1,0)$ or $(0,1)$, where the 1 is in the k^{th} position. Under equilibrium conditions, we have

$$\sum_{(n_1,n_2)\in\Theta(i)} P((n_1,n_2)-\overline{e_k}) = P(I=i-1) - P(I=i-1, n_k=l_k) \tag{7.1}$$

Figure 7.1 illustrates (7.1) for class 2 requests. The LHS indicates the set of states belonging to $\Theta(5)$ and the state probabilities from which we can get to the states in $\Theta(4)$ and their probabilities by moving one arrival backward for class 2. But this set of states in $\Theta(4)$ does not include the blocking state as indicated on the RHS.

$$\sum_{(n_1,n_2)\in\Theta(5)} P((n_1,n_2)-\overline{e_2}) = P(I=4) - P(I=4, n_2=3)$$

Consider resources used $I=\sum_{k=1}^{2} n_k$ and multiplying (4.3) with i and substituting for i and rearranging on the RHS, we have

$$iq_i = \sum_{k=1}^{2} \sum_{(n_1,n_2)\in\Theta(i)} n_k P(n_1,n_2)$$

Substituting from the balance equation, $n_k P(n_1,n_2) = \rho_k P((n_1,n_2)-\overline{e_k})$ we have,

$$iq_i = \sum_{k=1}^{2} \sum_{(n_1,n_2)\in\Theta(i)} \rho_k P((n_1,n_2)-\overline{e_k})$$

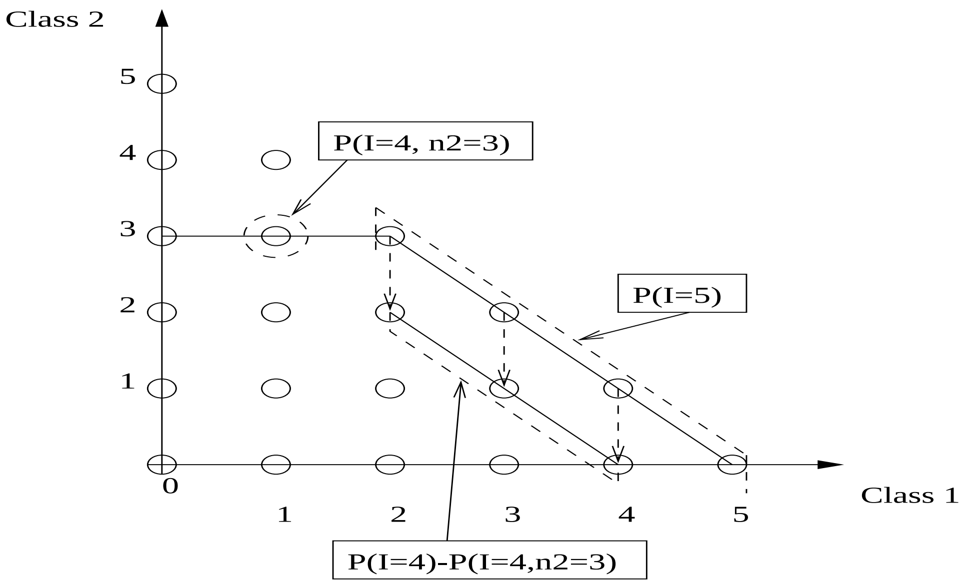

Figure 7.1: One arrival backward in the 2nd dimension

$$iq_i = \sum_{k=1}^{2} \rho_k \sum_{(n_1,n_2)\in\Theta(i)} P((n_1,n_2) - \overline{e_k})$$

Substituting from (7.1) for $\sum_{(n_1,n_2)\in\Theta(i)} P((n_1,n_2) - \overline{e_k})$ we have,

$$iq_i = \sum_{k=1}^{2} \rho_k [P(I = i-1) - P(I = i-1, n_k = l_k)]$$

Multiplying with G and using (4.4) and moving i to RHS we have the recursion in Theorem 1.

$$G(i) = \frac{1}{i} \sum_{k=1}^{2} \rho_k [G(i-1) - B_k(i-1)] \quad i = 1, \ldots, C$$

and $G(i) = 1$ for $i = 0$ and 0 for negative i.

QED

BIBLIOGRAPHY

BIBLIOGRAPHY

Aein, J. M. (1978). A multi-user-class, blocked-calls-cleared demand access mode. *IEEE Transactions on Communications COM-26*(3), 378–385.

Aggarwal, C. C., J. L. Wolf, and P. S. Yu (1996). On optimal batching policies for video-on-demand storage servers. In *Proceedings of the IEEE Conference on Multimedia Computing and Systems*, pp. 253–258.

Almeroth, K. C. and M. H. Ammar (1996). The use of multicast delivery to provide a scalable and interactive video-on-demand service. *IEEE Journal on Selected Areas in Communications 14*(6), 1110–1122.

Almeroth, K. C., A. Dan, D. Sitaram, and W. H. Tetzlaff (1997). Long term resource allocation in video delivery systems. In *Proceedings of the IEEE INFOCOM*, Volume 3, pp. 1333–1340.

Anderson, D. P., Y. Osawa, and R. Govindan (1992). A file system for continuous media. *ACM Transactions on Computer Systems 10*(4), 311–337.

Barnett, S. A. and G. J. Anido (1996). A cost comparison of distributed and centralized approaches to video-on-demand. *IEEE Journal on Selected Areas in Communications 14*(6), 1173–1183.

Barnett, S. A. and G. J. Anido (1998). Performability of disk-array-based video servers. *Multimedia Systems 6*(1), 60–74.

Baskett, F., M. Chandy, R. Muntz, and J. Palacios (1975). Open, closed, and mixed networks of queues with different class of customers. *Journal of the Association for Computing Machinery 22*(2), 248–260.

Bennett, J. C. R. and H. Zhang (1996). WF^2Q: Worst-case fair weighted fair queueing. In *Proceedings of the IEEE INFOCOM*, pp. 120–128.

Bennett, J. C. R. and H. Zhang (1997). Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking 5*(5), 675–689.

Berson, S., L. Golubchik, and R. R. Muntz (1995). Fault tolerant design of multimedia servers. In *Proceedings of the ACM SIGMOD*, pp. 364–375.

Bianchi, G., R. Melen, and A. Rainoni (1995). Performance of video storage hierarchy in interactive video services networks. In *Proceedings of the IEEE Global Telecommunications Conference*, Volume 2, pp. 805–810.

Biersack, E. and W. Geyer (1999). Synchronized delivery and playout of distributed stored multimedia streams. *Multimedia Systems 7*, 70–90.

Bisdikian, C. C. and B. V. Patel (1995). Issues on movie allocation in distributed video-on-demand systems. In *Proceedings of the IEEE International Conference on Communications*, Volume 1, pp. 250–255.

Braden, R., D. Clark, and S. Shenker (1994, Jul). *Integrated Services in the Internet Architecture: An Overview.* IETF - RFC 1633.

Brubeck, D. W. and L. A. Rowe (1996). Hierarchical storage management in a distributed VoD system. *IEEE Multimedia*, 37–47.

Buzen, J. P. (1973). Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM 16*(9), 527–531.

Chan, S. H. G., F. Tobagi, and T. M. Ko (1998). Providing on-demand video services using request batching. In *Proceedings of the IEEE Conference on Communications*, Volume 3, pp. 1716–1722.

Chang, E. and H. Garcia-Molina (1999). Medic: A memory and disk cache for multimedia clients. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Volume 1, pp. 493–499.

Chang, E. and A. Zakhor (1994a). Admission control and data placement for VBR video servers. In *Proceedings of the 1st IEEE International Conference on Image Processing*, pp. 278–282.

Chang, E. and A. Zakhor (1994b). Scalable video data placement on parallel disk arrays. In *Proceedings of the SPIE*, Volume 2185, pp. 208–221.

Chang, E. and A. Zakhor (1996). Cost analyses for VBR video servers. *IEEE Multimedia*, 56–71.

Chatterjee, S. (1997). Requirements for success in gigabit networking. *Communications of the ACM 40*(7), 64–73.

Chen, H. J. and T. D. C. Little (1996). Storage allocation policies for time-dependent multimedia data. *IEEE Transactions on Knowledge and Data Engineering 8*(5), 855–864.

Chen, H. J., T. D. C. Little, and D. Venkatesh (1995). A storage and retrieval technique for scalable delivery of MPEG-encoded video. *Journal of Parallel and Distributed Computing 30*(2), 180–189.

Chen, I. R. and C. M. Chen (1996). Threshold-based admission control policies for multimedia servers. *The Computer Journal (British Computer Society) 39*(9), 757–766.

Chen, I. R. and T. H. Hsi (1998). Performance analysis of admission control algorithms based on reward optimization for real-time multimedia servers. *Performance Evaluation 33*(2), 89–112.

Chen, M.-S. and D. D. Kandlur (1995). Downloading and stream conversion: Supporting interactive playout of videos in a client station. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 73–80.

Chen, M. S., D. D. Kandlur, and P. S. Yu (1993). Optimization of the grouped sweeping scheduling (GSS) with heterogeneous multimedia streams. In *Proceedings of the ACM Multimedia Conference*, pp. 235–241.

Chen, P. M., E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson (1996). RAID:high-performance, reliable secondary storage. *IEEE Journal on Selected Areas in Communications 14*(6), 1110–1122.

Chervenak, A. L., D. A. Patterson, and R. H. Katz (1995). Storage systems for movies-on-demand video servers. In *Proceedings of the IEEE Symposium on Mass Storage Systems*, pp. 246–256.

Chiariglione, L. (1997). MPEG and multimedia communications. *IEEE Transactions on Circuits and Systems for Video Technology 7*(1), 5–18.

Chiussi, F. M. and A. Francini (1998). Minimum-delay self-clocked fair queueing algorithm for packet-switched networks. In *Proceedings of the IEEE INFOCOM*, Volume 3, pp. 1112–1121.

Cruz, R. L. (1991). A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory 37*(1), 114–131.

Cruz, R. L. (1995). Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communication 13*(6), 1048–1056.

Dan, A., M. Kienzle, and D. Sitaram (1995). A dynamic policy of segment replication for load-balancing in video-on-demand servers. *Multimedia Systems 3*(3), 93–103.

Dan, A. and D. Sitaram (1995). An online video placement policy based on bandwidth to space ratio (bsr). In *Proceedings of the ACM SIGMOD Conference*, Volume 24, pp. 376–385.

Demers, A., S. Keshav, and S. Shenker (1990). Analysis and simulation of a fair queueing algorithm. *J. on Internetworking Research and Experience*, 3–26.

Denning, P. J. (1967). Effects of scheduling on file memory operations. In *Proceedings of the AFIPS Conference*, pp. 9–21.

Doganatha, Y. N. and A. N. Tantawi (1994). Making a cost-effective video server. *IEEE Multimedia 1*(4), 22–30.

Feng, W.-C. (1997). *Buffering Techniques for Delivery of Compressed Video in Video-on-Demand Systems*. Kluwer Academic Publishers.

Feng, W.-C. and J. Rexford (1997). A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video. In *Proceedings of the IEEE INFOCOM*, Volume 1, pp. 58–66.

Ferrari, D. and D. Verma (1990). A scheme for real-time channel establishment in wide-area networks. *IEEE J. Selected Areas in Communication 8*, 368–379.

Freedman, C. S. and D. J. DeWitt (1995). The SPIFFI scalable video-on-demand system. In *Proceedings of the ACM SIGMOD*, pp. 352–363.

Furht, B., D. Kalra, A. A. Rodriguez, and W. E. Wall (1995). Design issues for interactive television systems. *IEEE Computer*, 25–39.

Gemmell, D. J. (1993). Multimedia network file servers: Multi-channel delay sensitive data retrieval. In *Proceedings of the ACM Multimedia Conference*, pp. 243–250.

Gemmell, D. J. and S. Christodoulakis (1992). Principles of delay-sensitive multimedia data storage and retrieval. *ACM Transactions on Information Systems 10*(1), 51–90.

Gemmell, D. J., H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. Rowe (1995). Multimedia storage servers: A tutorial. *IEEE Computer 28*(5), 40–49.

Georgiadis, L., R. Guerin, V. Peris, and R. Rajan (1996, August). Efficient support of delay and rate guarantees in an internet. In *ACM SIGCOMM*, pp. 106–116.

Ghandeharizadeh, S. and L. Ramos (1993). Continuous retrieval of multimedia data using parallelism. *IEEE Transactions on Knowledge and Data Engineering 5*(4), 658–669.

Golestani, S. J. (1991). Congestion-free communication in high-speed packet networks. *IEEE Transactions on Communications 39*(12), 1802–1812.

Golestani, S. J. (1994). A self-clocked fair queueing scheme for broadband applications. In *Proceedings of the IEEE INFOCOM*, pp. 636–646.

Gollapudi, S. and A. Zhang (1998). Buffer model and management in distributed multimedia presentation system. *Multimedia Systems 6*, 206–218.

Golubchik, L., J. C. S. Lui, and R. R. Muntz (1996). Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers. *Multimedia Systems 4*(3), 140–155.

Griwodz, C., M. Bar, and L. C. Wolf (1997). Long-term movie popularity in video-on-demand systems or the life of an on-demand movie. In *Proceedings of the ACM Multimedia Conference*, pp. 349–357.

Griwodz, C., O. Merkel, J. Dittman, and R. Steinmetz (1998). Protecting VoD the easier way. In *Proceedings of the ACM Multimedia Conference*, pp. 21–28.

Gross, D. and C. M. Harris (1998). *Fundamentals of Queueing Theory.* John Wiley & Sons, Inc.

Haskin, R. and F. Schmuck (1996). The Tiger Shark file system. In *Proceedings of COMPCON*.

Holton, M. and R. Das. XFS: A next generation journalled 64-bit file system with guaranteed rate I/O. Technical report, Silicon Graphics, Inc.

Hsieh, J., M. Lin, J. C. L. Liu, and D. H. C. Du (1995). Performance of a mass storage system for video-on-demand. *Journal of Parallel and Distributed Computing 30*(2), 147–167.

Hua, K. A., Y. Cai, and S. Sheu (1998). Patching: A multicast technique for true video-on-demand services. In *Proceedings of the ACM Multimedia Conference*, pp. 191–200.

Hwang, E., K. Kilic, and V. S. Subrahmanian (1998). Handling updates and crashes in VoD systems. *Multimedia Tools and Applications 7*, 103–132.

Jadav, D. and A. Choudhary (1995). Designing and implementing high-performance media-on-demand servers. *IEEE Parallel and Distributed Technology*, 29–39.

Jordan, S. and P. P. Varaiya (1991). Throughput in multiple service, multiple resource communication networks. *IEEE Transactions on Communications 39*(8), 1216–1222.

Jun, S. B. and W. S. Lee (1998). Video allocation methods in a multi-level server for large-scale vod services. *IEEE Transactions on Consumer Electronics 44*(4), 1309–1318.

Kalmanek, C. R., H. Kanakia, and S. Keshav (1990). Rate controlled servers for very high speed networks. In *Proceedings of the IEEE GLOBECOM*, Volume 1, pp. 12–20.

Kalva, H. and B. Furht (1996). Techniques for improving the capacity of video-on-demand systems. In *Proceedings of the 29th Annual Hawiii International Conference on System Sciences*, Volume 2, pp. 308–315.

Kandlur, D. D., M. S. Chen, and Z. Y. Shae (1994). Design of a multimedia storage server. In *Proceedings of the SPIE*, pp. 164–178.

Kaufman, J. S. (1981). Blocking in a shared resource environment. *IEEE Transactions on Communications COM-29*(10), 1474–1481.

Keeton, K., A. Drapeau, D. Patterson, and R. H. Katz (1994). Storage alternatives for video service. In *Proceedings of the thirteenth Symposium on Mass Storage Systems*, pp. 100–105.

Keeton, K. and R. H. Katz (1995). Evaluating video layout strategies for a high-performance storage server. *Multimedia Systems 3*(2), 43–52.

Kenchammana-Hosekote, D. R. and J. Srivastava (1994). Scheduling continuous media in a video-on-demand server. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 19–28.

Keshav, S. (1997). *An Engineering Approach to Computer Networking*. Addison Wesley.

Kim, H. J. and Y. Zhu (1998). Channel allocation problem in VoD system using both batching and adaptive piggybacking. *IEEE Transactions on Consumer Electronics 44*(3), 969–976.

Kim, S.-E. and C. R. Das (1998). Analyzing cache performance for video servers. In *Proceedings of the ICPP Workshop*, pp. 38–47.

Kleinrock, L. (1975). *Queueing Systems: Volume 1, Theory*. John Wiley & Sons, Inc.

Kohiyama, K., H. Shirai, K. Ogawa, A. Manakata, Y. Koga, and M. Ishizaki (1996). Architecture of MPEG-2 digital set-top box for CATV VoD system. *IEEE Transactions on Consumer Electronics 42*(3), 667–672.

Krikelis, A. (1998). Scalable multimedia servers. *IEEE Concurrency 6*(4), 8–10.

Krishnan, R., D. Venkatesh, and T. D. C. Little (1997). A failure and overload tolerance mechanism for continuous media servers. In *Proceedings of the ACM Multimedia*, pp. 131–142.

Kunii, T. L., Y. Shinagawa, R. M. Paul, M. F. Khan, and A. A. Khokhar (1995). Issues in storage and retrieval of multimedia data. *Multimedia Systems 3*(5), 298–304.

Lai, Y. C., Y. D. Lin, and H. Z. Lai (1997). A hierarchical network storage architecture for video-on-demand services. *IEEE Transactions on Broadcasting 43*(2), 145–154.

Lam, S. S., S. Chow, and D. K. Y. Yau (1994). An algorithm for lossless smoothing of MPEG video. In *Proceedings of ACM SIGCOMM*, Volume 24, pp. 281–293.

Lam, S. S., S. Chow, and D. K. Y. Yau (1996). A lossless smoothing algorithm for compressed video. *IEEE/ACM Transactions on Networking 4*(5), 697–708.

Lau, S. W., J. C. S. Lui, and P. C. Wong (1995). A cost-effective near-line storage server for multimedia system. In *Proceedings of the Eleventh International Conference on Data Engineering*, pp. 449–456.

Lee, H.-J. and D. H. C. Du (1997). The effect of disk scheduling schemes on a video server for supporting quality MPEG video accesses. In *Proceedings of the IEEE International conference on multimedia computing and systems*, pp. 194–201.

Lee, J. Y. B. (1998). Parallel video servers: A tutorial. *IEEE Multimedia 5*(2), 20–28.

LeGall, D. (1991). MPEG: A video compression standard for multimedia applications. *Communications of the ACM 34*(4), 47–58.

Leung, M. Y. Y., J. C. S. Lui, and L. Golubchik (1997). Buffer and I/O resource pre-allocation for implementing batching and buffering techniques for video-on-demand systems. In *Proceedings of the ICDE*, pp. 344–353.

Li, V. O. K. and W. Liao (1996). Performance model of interactive video-on-demand systems. *IEEE Journal on Selected Areas in Communications 14*(6), 1099–1109.

Li, V. O. K. and W. Liao (1997). Distributed multimedia systems. *Proceedings of the IEEE 85*(7), 1063–1108.

Lie, P. W. K., J. C. S. Lui, and L. Golubchik (1998). Threshold-based dynamic replication in large scale video-on-demand systems. In *Proceedings of the Eighth International Workshop on Research Issues in Data Engineering*, pp. 52–59.

Liebeherr, J. (1995). Multimedia networks: Issues and challenges. *IEEE Computer 28*(4), 68–69.

Liebherr, J., D. E. Wrege, and D. Ferrari (1996). Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking 4*(6), 885–901.

Liew, S. C. and C.-Y. Tse (1996). Video aggregation: Adapting video traffic for transport over broadband networks by integrating data compression and statistical multiplexing. *IEEE Journal on Selected Areas in Communications 14*(6), 1123–1137.

Liew, S. C. and C.-Y. Tse (1998). A control-theoretic approach to adapting VBR compressed video for transport over a cbr communications channel. *IEEE/ACM Transactions on Networking 6*(1), 42–55.

Lin, F. Y. (1998). Optimal real-time admission control algorithms for the video-on-demand (VoD) service. *IEEE Transactions on Broadcasting 44*(4), 402–408.

Little, T. D. C. and D. Venkatesh (1994). Prospects for interactive video-on-demand. *IEEE Multimedia 1*(3), 14–24.

Little, T. D. C. and D. Venkatesh (1995). Popularity-based assignment of movies to storage devices in an video-on-demand system. *Multimedia Systems 2*(6), 280–287.

Liu, C. L. and J. W. Layland (1973). Scheduling algorithms for multimprogramming in a hard real-time environment. *Journal of the ACM 20*(1), 46–61.

Lodge, N. K. (1992). Video compression techniques. In *Proceedings of the IEE Colloquium on Applications of Video Compression in Broadcasting*, pp. 2/1–2/14.

MacDougall, M. H. (1987). *Simulating Computer Systems*. The MIT Press.

Merchant, A., Q. Ren, and B. Sengupta (1996). Hierarchical storage servers for video-on-demand: feasibility, design and sizing. In *Proceedings of Global Telecommunications Conference*, Volume 1, pp. 272–278.

Miyazaki, Y. and K. Nahrstedt (1997). Dynamic coordination of movies according to popularity index and resource availability within a hierarchical VoD system. In *Proceedings of the IEEE TENCON*, Volume 1, pp. 199–202.

Morris, O. J. (1995). MPEG-2: where did it come from and what it is? In *Proceedings of the IEE Colloquium on MPEG-2 - What it is and What it isn't*, pp. 1/1–1/5.

Mourad, A. N. (1996). Issues in the design of a storage server for video-on-demand. *Multimedia Systems 4*(2), 70–86.

Mundur, P., R. Simon, and A. Sood (1999a, January). End-to-end request handling in distributed video-on-demand systems. In *Proceedings of the CNDS Conference (Society for Computer Simulation*, Volume 1, pp. 151–156.

Mundur, P., R. Simon, and A. Sood (1999b, June). Integrated admission control in hierarchical video-on-demand systems. In *Proceedings of the IEEE Conference on Multimedia Computing and Systems (ICMCS)*, Volume 1, pp. 220–225.

Mundur, P., A. Sood, and R. Simon (1998, February). Threshold-based admission control for multi-class video-on-demand systems. In *Proceedings of the IEEE Performance, Communications, and Computing Conference (IPCCC)*, pp. 154–160.

Mundur, P., A. Sood, and R. Simon (1999, July). Effects of traffic reshaping on distributed video-on-demand systems. Technical report, George Mason University, Fairfax, VA 22030.

Nahrstedt, K. and R. Steinmetz (1995). Resource management in networked multimedia systems. *IEEE Computer 28*(5), 52–63.

Nussbaumer, J. P., B. V. Patel, F. Schaffa, and J. P. G. Sterbenz (1995). Networking requirements for interactive video on demand. *IEEE Journal on Selected Areas in Communications 13*(5), 779–787.

Ozden, B., R. Rastogi, P. Shenoy, and A. Silberschatz (1996). Fault-tolerant architectures for continuous media servers. In *Proceedings of the ACM SIGMOD*, Volume 25, pp. 79–90.

Ozden, B., R. Rastogi, and A. Silberschatz (1995). Research issues in multimedia storage servers. *ACM Computing Surveys 27*(4), 617–620.

Ozden, B., R. Rastogi, and A. Silberschatz (1997). Periodic retrieval of videos from disk arrays. In *Proceedings of the International Conference on Data Engineering*, pp. 333–343.

Pang, H. (1997). Tertiary storage in multimedia systems: Staging or direct access. *Multimedia Systems 5*(6), 386–399.

Parekh, A. and R. Gallager (1993). A generalized processor sharing approach to flow control - the single node case. *IEEE/ACM Transactions on Networking 1*, 344–357.

Parekh, A. and R. Gallager (1994). A generalized processor sharing approach to flow control - the multiple node case. *IEEE/ACM Transactions on Networking 2*(2), 137–150.

Partridge, C. (1994). *Gigabit Networking*. Addison Wesley.

Ramanathan, S. and P. V. Rangan (1994). Architectures for personalized multimedia. *IEEE Multimedia 1*(1), 37–46.

Ramarao, R. and V. Ramamoothy (1991). Architectural design of on-demand video delivery systems: The spatio-temporal storage allocation problem. In *Proceedings of the IEEE International Conference on Communications*, Volume 1, pp. 506–510.

Rangan, P. V. and H. M. Vin (1991). Designing file systems for digital video and audio. *Operating Systems Review 25*(5), 69–79.

Rangan, P. V. and H. M. Vin (1993). Efficient storage techniques for digital continuous media. *IEEE Transactions on Knowledge and Data Engineering 5*(4), 564–573.

Rangan, P. V., H. M. Vin, and S. Ramanathan (1992). Designing an on-demand multimedia service. *IEEE Communications Magazine*, 56–64.

Reddy, A. L. N. and J. Wyllie (1993). Disk scheduling in a multimedia I/O system. In *Proceedings of the ACM Multimedia Conference*, pp. 225–233.

Reddy, A. L. N. and J. Wyllie (1994). I/O issues in a multimedia system. *IEEE Computer 27*(3), 69–74.

Riley, M. J. and I. E. G. Richardson (1997). *Digital Video Communications*. Artech House Publishers.

Roberts, J. W. (1981). A service system with heterogenous user requirements - application to multi-services telecommunications systems. In *Performance Data Communications Systems Conference Proceedings*, pp. 423–431. North-Holland Publishing Co.

Rompogiannakis, Y., G. Nerjes, P. Muth, M. Paterakis, P. Triantafillou, and G. Weikum (1998). Disk scheduling for mixed-media workloads in a multimedia server. In *Proceedings of ACM Multimedia Coneference*, pp. 297–302.

Ross, K. W. and D. H. K. Tsang (1989). The stochastic knapsack problem. *IEEE Transactions on Communications 37*(7), 740–747.

Salehi, J. D., Z. L. Zhang, J. Kurose, and D. Towsley (1998). Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing. *IEEE/ACM Transactions on Networking 6*(4), 397–410.

Santos, J. R. and R. Muntz (1998). Performance analysis of the RIO multimedia storage system with heterogeneous disk configurations. In *Proceedings of the ACM Multimedia Conference*, pp. 303–308.

Schaffa, F. and J. P. Nussbaumer (1995). On bandwidth and storage tradeoffs in multimedia distribution networks. In *Proceedings of the INFOCOM*, Volume 3, pp. 1020–1026.

Schwartz, M. and B. Kraimeche (1983). An analytic control model for an integrated node. In *Proceedings of the INFOCOM*, pp. 540–546.

Schwetman, H. (1998). *CSIM18 User's Manual.* Mesquite Software, Inc.

Shenoy, P. J., P. Goyal, S. S. Rao, and H. M. Vin (1998). Symphony: An integrated multimedia file system. In *Proceedings of the ACM/SPIE Multimedia Computing and Networking Conference*, pp. 124–138.

Shenoy, P. J., P. Goyal, and H. M. Vin (1995). Issues in multimedia server design. *ACM Computing Surveys 27*(4), 636–639.

Shenoy, P. J., P. Goyal, and H. M. Vin (1999). Architectural considerations for next generation file systems. In *To appear in Proceedings of the ACM Multimedia Conference*.

Shenoy, P. J. and H. M. Vin (1998). Cello: A disk scheduling framework for next generation operating systems. In *Proceedings of ACM SIGMETRICS, the International Conference on Measurement and Modeling of Computer Systems*, pp. 44–55.

Shyu, I.-J. and S.-P. Shieh (1999). A distributed fault-tolerant design for multiple-server VoD systems. *Multimedia Tools and Applications 8*(2), 219–247.

Sikora, T. (1997). MPEG digital video-coding standards. *IEEE Signal Processing Magazine 14*(5), 82–100.

Sincoskie, W. D. (1990). Video on demand: is it feasible? In *Proceedings of Global Telecommunications Conference, GLOBECOM '90*, Volume 1, pp. 201–205.

Sonah, B. and M. B. Ito (1998). New adaptive object replacement policy for video-on-demand systems. In *Proceedings of the IEEE Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 13–18.

Srivastava, A., A. Kumar, and A. Singru (1997). Design and analysis of a video-on-demand server. *Multimedia Systems 5*, 238–254.

Stiliadis, D. and A. Verma (1998a). Efficient fair queueing algorithms for packet-switched networks. *IEEE/ACM Transactions on Networking 6*(2), 175–185.

Stiliadis, D. and A. Verma (1998b). Rate-proportional servers: A design methodology for fair queueing algorithms. *IEEE/ACM Transactions on Networking 6*(2), 164–174.

Suri, S., G. Varghese, and G. Chandranmenon (1997). Leap forward virtual clock: A new fair queueing scheme with guaranteed delays and throughput fairness. In *Proceedings of the IEEE INFOCOM*, pp. 557–565.

Tewari, R., D. M. Dias, R. Mukherjee, and H. M. Vin (1996). High availability in clustered multimedia servers. In *Proceedings of the International Conference on Data Engineering*.

Tewari, R., R. Mukherjee, D. M. Dias, and H. M. Vin (1996). Design and performance tradeoffs in clustered video servers. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pp. 144–150.

Tobagi, F. A. and I. Dalgic (1996). On the performance of networks carrying multimedia traffic. In W. Effelsberg, O. Spaniol, A. Danthine, and D. Ferrari (Eds.), *High Speed Networking for Multimedia Applications*, pp. 61–97. Kluwer Academic Publishers.

Tobagi, F. A., J. Pang, R. Baird, and M. Gang (1993). Streaming RAID: A disk storage system for video and audio files. In *Proceedings of the ACM Multimedia Conference*, pp. 393–400.

Tsai, W. J. and S. Y. Lee (1998). Dynamic buffer management for near video-on-demand systems. *Multimedia Tools and Applications 6*(1), 61–83.

Tsang, D. H. K. and K. W. Ross (1990). Algorithms to determine exact blocking probabilities for multirate tree networks. *IEEE Transactions on Communications 38*(8), 1266–1271.

Venkatasubramanian, N. and S. Ramanathan (1997). Load management in distributed video servers. In *Proceedings of the International Conference on Distributed Computing Systems*, pp. 528–535.

Verma, D., H. Zhang, and D. Ferrari (1991). Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of TRICOMM '91*, pp. 35–46.

Vin, H. M., A. Goyal, A. Goyal, and P. Goyal (1994a). An observation-based admission control algorithm for multimedia servers. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 234–243.

Vin, H. M., A. Goyal, and P. Goyal (1995). Algorithms for designing large-scale multimedia servers. *Computer Communications 18*, 192–203.

Vin, H. M., P. Goyal, A. Goyal, and A. Goyal (1994b). A statistical admission control algorithm for multimedia servers. In *Proceedings of the ACM Multimedia Conference*, pp. 33–40.

Vin, H. M. and P. V. Rangan (1993). Designing a multiuser HDTV storage server. *IEEE Journal on Selected Areas in Communications 11*(1), 153–164.

Waltrich, J. B. (1993). Digital video compression - an overview. *Journal of Lightwave Technology 11*(1), 70–75.

Wang, Y. and D. H. C. Du (1998). On providing highly available fault-tolerant video-on-demand services. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 76–85.

Wang, Y., J. C. L. Liu, D. H. C. Du, and J. Hsieh (1997). Efficient video file allocation schemes for video-on-demand services. *Multimedia Systems 5*, 283–296.

White, P. P. (1997, May). RSVP and integrated services in the internet: A tutorial. *IEEE Communications Magazine 35*(5), 100–106.

Wijayaratne, R. and A. L. N. Reddy (1999). Integrated QoS management for disk I/O. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Volume 1, pp. 487–492.

Wolf, J. L., P. S. Yu, and H. Shachnai (1997). Disk load balancing for video-on-demand systems. *Multimedia Systems 5*, 358–370.

Wolf, L. C., C. Griwodz, and R. Steinmetz (1997). Multimedia communication. *Proceedings of the IEEE 85*(12), 1915–1933.

Won, Y. and J. Srivastava (1996). Minimizing blocking probability in a hierarchical storage based VoD server. In *Proceedings of the International Workshop on Multimedia Database Management Systems*, pp. 12–19.

Wroclawski, J. (1997, Sep). *The Use of RSVP with IETF Integrated Services.* IETF - RFC 2210.

Yu, P. S., M. S. Chen, and D. D. Kandlur (1992). Design and analysis of a grouped sweeping scheme for multimedia storage management. In *Proceedings of the third international workshop on network and operating system support for digital audio and video*, pp. 38–49.

Yu, P. S., M. S. Chen, and D. D. Kandlur (1993). Grouped sweeping scheduling for DASD-based multimedia storage management. *Multimedia Systems 1*(3), 99–109.

Zhang, H. (1995). Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE 83*(10), 1374–1396.

Zhang, H. and D. Ferrari (1993). Rate-controlled static priority queueing. In *Proceedings of INFOCOM*, pp. 227–236.

Zhang, L. (1991, May). Virtualclock: A new traffic algorithm for packet switched networks. *ACM Transactions on Computer Systems 9*(2), 101–124.

Zhang, L., S. Deering, D. Estrin, S. Shenker, and D. Zappala (1993, Sep). RSVP: A new resource reservation protocol. *IEEE Network 7*(5), 8–18.

Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison Wesley, Reading MA.

CURRICULUM VITAE

Padmavathi V. Mundur was born on October 26, 1962, in Bangalore, India and is currently a U.S. permanent resident. She received a Bachelor of Engineering degree in Industrial and Production Engineering from Bangalore University in 1985, a Post-graduate Diploma in Industrial Management from Indian Institute of Science in Bangalore in 1986, and a Master of Engineering in Systems Engineering from University of Virginia in Charlottesville in 1990. Padma has published six conference and journal papers in the area of distributed multimedia systems.

Prior to her doctoral studies at GMU, Padma worked for the Georgetown University Hospital as a Management Engineer providing Industrial Engineering services in the areas of process engineering and quality control. During her Ph.D. program at GMU, she has taught introductory computer science courses to undergraduates. She also worked as a UNIX systems administrator for the CS Department and the Center for the New Engineer(CNE).