Distributed streaming media architecture

Vijay Gondi

Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation
Gondi, Vijay, "Distributed streaming media architecture" (2001). Retrospective Theses and Dissertations. 21229.
https://lib.dr.iastate.edu/rtd/21229

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Distributed streaming media architecture

by

Vijay Gondi

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Soma Chaudhuri, Co-major Professor
G. Manimaran, Co-major Professor
Pete Boysen

Iowa State University
Ames, Iowa
2001

Copyright © Vijay Gondi, 2001. All rights reserved.
Graduate College
Iowa State University

This is to certify that the Master's thesis of

Vijay Gondi

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Characteristics of Streaming Media Architecture</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Problem Statement and Thesis Contribution</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER 2. END-TO-END STREAMING MEDIA ARCHITECTURE</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Encoder</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Media Server</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1 Operating System</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2 Storage System</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Application-Layer QoS Control and Transport Protocols</td>
<td>11</td>
</tr>
<tr>
<td>2.4 Continuous Media Distribution Services (CMDS)</td>
<td>13</td>
</tr>
<tr>
<td>2.5 Motivation for Distributed Streaming Media Architecture</td>
<td>14</td>
</tr>
<tr>
<td>CHAPTER 3. PROPOSED DISTRIBUTED MEDIA ARCHITECTURE</td>
<td>17</td>
</tr>
<tr>
<td>3.1 Coordinator and Proxy</td>
<td>19</td>
</tr>
<tr>
<td>3.1.1 Coordinator State Diagram</td>
<td>19</td>
</tr>
<tr>
<td>3.1.2 Proxy State Diagram</td>
<td>21</td>
</tr>
<tr>
<td>CHAPTER 4. PERFORMANCE STUDIES</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Performance Metrics</td>
<td>24</td>
</tr>
<tr>
<td>4.1.1 Multiple Description Coding</td>
<td>24</td>
</tr>
<tr>
<td>4.1.2 Layered Coding</td>
<td>26</td>
</tr>
<tr>
<td>4.2 Simulation Parameters</td>
<td>27</td>
</tr>
</tbody>
</table>
CHAPTER 5. RESULTS AND DISCUSSIONS .......................... 30
  5.1 Multiple Description Coding .................................. 30
    5.1.1 Effect of Distribution of Media Descriptions ..... 33
  5.2 Layered Coding .................................................. 34

CHAPTER 6. CONCLUSIONS AND FUTURE WORK ................... 37
  6.1 Future Work ...................................................... 37
LIST OF TABLES

Table 4.1  Simulation Parameters .................................................. 29
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A schematic of end-to-end Streaming Media Architecture</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Storage-Area-Network</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Network-assisted-Storage</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Distributed Media Architecture</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Coordinator State Diagram</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>Proxy State Diagram</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Circular Buffer for MDC</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>Circular Buffer for Layered Encoding</td>
<td>27</td>
</tr>
<tr>
<td>4.3</td>
<td>Network Topology used for simulations</td>
<td>28</td>
</tr>
<tr>
<td>5.1</td>
<td>Quality perception for Zipf Skew factor=0.9</td>
<td>31</td>
</tr>
<tr>
<td>5.2</td>
<td>Fairness metric for Zipf Skew factor=0.9</td>
<td>31</td>
</tr>
<tr>
<td>5.3</td>
<td>Jitter Characteristics for Zipf Skew factor=0.9 (MDC)</td>
<td>32</td>
</tr>
<tr>
<td>5.4</td>
<td>Maximum breaks observed by the clients for a Zipf Skew factor=0.9 (MDC)</td>
<td>32</td>
</tr>
<tr>
<td>5.5</td>
<td>Average Number of breaks observed by the clients for a Zipf Skew factor=0.9</td>
<td>33</td>
</tr>
<tr>
<td>5.6</td>
<td>Fairness metric for a varying Zipf Skew factor</td>
<td>33</td>
</tr>
<tr>
<td>5.7</td>
<td>Quality Perception for a Zipf Skew factor=0.7</td>
<td>34</td>
</tr>
<tr>
<td>5.8</td>
<td>Fairness metric for a Zipf Skew factor=0.7</td>
<td>35</td>
</tr>
<tr>
<td>5.9</td>
<td>Quality perception for a Zipf Skew factor=0.9 (Layered Distribution)</td>
<td>35</td>
</tr>
<tr>
<td>5.10</td>
<td>Fairness metric for a Zipf Skew factor=0.9 (Layered Distribution)</td>
<td>36</td>
</tr>
</tbody>
</table>
ABSTRACT

The recent trends in content delivery indicate that media distribution is among the fastest growing services over the Internet. The lack of QoS support in the Internet has accelerated the development of content distribution architectures and protocols employing techniques such as caching, mirroring and application layer multicast. Though there have been significant efforts in this direction, the large scale deployment of such architectures is still a challenging problem. This motivates us to develop a novel architecture for content delivery over the best-effort Internet. Towards achieving this goal, we first identify the key components that build up the end-to-end architecture of a media distribution system and discuss their functionalities. Then, we propose a distributed streaming media architecture that is capable of addressing the requirements of client heterogeneity, scalability, and fault-tolerance, overcoming the deficiencies of traditional streaming media architectures. The proposed architecture is highly suitable for scalable encoding techniques such as Multiple Descriptive Coding and Layered Coding. To evaluate the performance of the proposed architecture, we define several performance metrics and carry out extensive simulation studies. Our studies show that clients experience better quality characteristics in the distributed architecture compared to the single server architecture. The proposed distributed architecture brings up several issues, such as quality adaptation, cache replacement and server fault-tolerance, which need further research.
CHAPTER 1. INTRODUCTION

The Internet has experienced an explosive growth in the recent years; originally envisaged as a medium of transfer for text and static images, the Internet has experienced unprecedented growth in terms of the available content. Even more impressive is the growth of vast demographically varying Internet users. Improved computing technologies have made it feasible to perform complex compression schemes and increased bandwidth has paved way to provide online multimedia services over the Internet. Video and audio services currently attribute to significant amount of the Internet traffic; their growth is dependent on the QoS guarantees that can be provided over the Internet. Multimedia services on the Internet span wide walks of life; distance education, the entertainment industry, security systems (surveillance), museums, libraries, etc. While a multitude of multimedia applications exist, most of the applications would fall under one of the following categories:

1. Live Streaming.
2. Interactive Applications. (e.g., Video Conferencing, Interactive Games)
3. Digital Audio/Video Broadcasting. (e.g., Online Radio Channels)
4. Stored Media Streaming. (e.g., Video on Demand (VOD))

Each of the above categories have different requirements and challenges. Live streaming and interactive applications would require smaller end-to-end delays; digital broadcasting would require multicast support of the Internet or an overlay network for efficient use of network resources [15]. In this work, we outline the components that build up an end-to-end architecture for delivering stored media and propose a distributed architecture that would support streaming of stored media on a large scale.
Stored media can be transmitted over the Internet in two modes – download mode and streaming mode. In download mode, the user downloads the entire file before playback of the media file. However the full transfer in download mode usually suffers long and unacceptable transfer time. In contrast, in streaming mode, the video content need not be downloaded in full, but is played from a small buffer. The buffer is played out while parts of the content is still being received and decoded. A detailed comparison of download versus streaming mode is provided in [16].

1.1 Characteristics of Streaming Media Architecture

1. **Quality of Service (QoS) Guarantees:** As the name suggests real-time multimedia has timing constraints and is soft-real time in nature, i.e., they are capable of tolerating some amount of packet loss and delay. However beyond a certain threshold, the experience of the application could be annoying to the human ears and eyes, due to the pause and delays that could be observed during playback of the media stream. The best-effort nature of the Internet does not guarantee QoS requirements (loss, delay and jitter). An end-to-end architecture needs to account for QoS guarantees, by providing resources over the Internet that would account to minimize the loss, delay and jitter.

2. **Fault-Tolerance:** To provide uninterrupted services, fault-tolerance is an important characteristics for an end-to-end streaming media architecture. In the event of failure, the ability to maintain the desired quality depends on scope of the system to tolerate loss and the effective use of redundancy defined in the system.

3. **Scalability:** The scalability of a system determines the feasibility of the architecture for being deployed on a large scale and the scope of the system to sustain future expansions.

4. **Client Heterogeneity:** Current Internet users show wide diversity in terms of connection speeds and the systems used to receive Internet services. While a vast majority of the users still use 56Kbps modems, fast access networks like DSL and cable networks are on the rise. PCs are not the only devices to access such services on the Internet. With
the advent of third generation wireless services, wireless Internet access is also unto a fast growth. Devices with multimedia capability are being developed for wireless access; e.g., PDAs (Personal Digital Assistant). So a streaming media architecture should be capable of servicing diverse devices and handle varying bandwidth in the network.

5. **Resource Utilization**: Even though the available bandwidth on the Internet is on the rise, the dearth for it would always exist considering the applications that are being deployed over the Internet. For streaming services, where a single flow could be long lived, there are possibilities for batching (using a single stream to serve multiple clients) and multicasting, for optimal resource utilization.

### 1.2 Problem Statement and Thesis Contribution

The Internet is a highly scalable packet-switched network, but however it was not designed to handle isochronous (continuous time-based) information. Due to the best-effort nature of the Internet, the soft-real-time requirements of multimedia are not guaranteed. Packets may be dropped or may experience excessive delays on the Internet due to congestion. To improve the quality, Continuous Media Distribution Service (CMDS) (e.g., proxy caching, mirroring) are being deployed over the Internet. CMDS provide adequate network support to reduce transport delays and packet loss. Built on top of the Internet they are able to achieve QoS and efficiency for streaming audio and video over best-effort networks.

In this work, we identify the various components, their design issues; that build up the end-to-end architecture of media distribution system. We present a CMDS architecture, that would improve the over-all quality perceived by the client. The proposed CMDS architecture consists of a set of cooperative distributed servers, which are designed to account for load balancing of the server and network. The architecture adopts a scalable encoding scheme to handle client heterogeneity. The proposed CMDS architecture is resilient to faults, by using the inherent redundancy defined in the system. We define performance metrics to evaluate the proposed distributed architecture and validate the core of the architecture using simulations on NS-2 simulator.
Chapter 2 outlines the components and design issues involved in the end-to-end delivery of stored media and the motivation for a new CMDS architecture. Chapter 3 describes the proposed CMDS scheme. Chapter 4 describes the experiments that were performed to study the proposed architecture. Chapter 5 presents the results and observations and chapter 6 provides the conclusions and future work.
CHAPTER 2. END-TO-END STREAMING MEDIA ARCHITECTURE

The quality perceived by a client is influenced by the components in the end-to-end media distribution system. Figure 2.1 shows the schematic diagram of an end-to-end streaming media architecture. The main components of the architecture are the encoder, media server, continuous media distribution services and the client.

Figure 2.1 A schematic of end-to-end Streaming Media Architecture

2.1 Encoder

Due to large disk requirements of raw media data, streaming media is stored in compressed form on storage devices. The types of encoding used to achieve the compression are discussed, with their relative merits and demerits outlined.
1. **Multiple Encodings:** Current services on the Internet provide multiple encodings of a video stream, i.e., each video is encoded in multiple bit rates. The choice of the bit stream to be transmitted is dependent on the client. This technique has a number of drawbacks. Firstly, the client might not be aware of the end-to-end capability of the system (e.g., a client connected to the Internet on a T1 link, would assume the presence of sufficient bandwidth for obtaining a video stream, but might be unaware of any intermediate bottleneck between the source and destination). Also the server has to maintain the same video in multiple encodings resulting in large storage requirements. With the drop of storage costs, this might seem feasible, but the growing diversity of devices and access bit rates prohibits such a scheme.

2. **Layered Coding:** In layered coding [1] [4] data is partitioned into a base layer and a few enhancement layers. The base layer contains visually important video data that can be used to produce video output of acceptable quality, whereas the enhancement layers contain complementary information that allows a higher quality video data to be generated. Layered coding is suitable for networks with priority support, with the base layer being assigned a higher priority so that it has a larger probability of being delivered error free when the network conditions worsen. Layered coding is popular with asynchronous transfer mode (ATM) networks but not suitable directly for the Internet. First, the Internet does not provide prioritized delivery of different layers. Second, when the packet-loss rate is high and part of the base layer is lost, it is hard to reconstruct the lost data since no redundancy is present. To adapt layered encoding to the Internet, quality-adaptive architectures have been proposed [21] [20] [13]. While most standardized layered encoders produce two or more layers [1][4]. The discrete nature of each layer causes variation in bandwidth when adopted for architectures like receiver-driven layered multicast [13]. This resulted in an interest for having layers of fine granularity and hence the standard MPEG-4 Finely Granular Scalability (FGS) [18]. The bit stream structure for MPEG-4 FGS makes it highly robust and flexible to changes in available bandwidth. However such robustness comes with some penalty for compression efficiency.
3. Multiple Description Coding (MDC): In MDC, a raw video sequence is compressed into multiple streams (referred to as descriptions) as follows: each description provides acceptable visual quality; more combined descriptions provide better visual quality. The advantages of MDC are:

- robustness to loss: even if a receiver gets only one description (other descriptions being lost), it can still reconstruct video with acceptable quality.
- enhanced quality: if a receiver gets multiple descriptions, it can combine them together to produce a better reconstruction than that produced from any one of them.

However, to make each description provide acceptable visual quality, each description must carry sufficient information about the original video. This will reduce the compression efficiency compared to conventional single description coding. In addition, although more combined descriptions provide a better visual quality, a certain degree of correlation between the multiple descriptions has to be embedded within each description, resulting in further reduction of the compression efficiency. Research is currently ongoing to achieve better compression efficiency [19].

2.2 Media Server

Media servers play a key role in providing streaming services; to offer quality streaming services, servers are required to process multimedia data under timing constraints in order to prevent artifacts (e.g., jerkiness in video motion and pops in audio) during playback at the client. A media server typically consists of the following three subsystems.

1. Operating System: Unlike traditional operating systems, Media servers need to employ techniques that would satisfy the real-time requirements for streaming applications.

2. Storage System: A storage system for streaming services has to support continuous media storage and retrieval with high throughput, large capacity and fault-tolerance.
3. **Communicator**: A communicator consists of the application layer QoS and transport protocols implemented in the server (shown in Figure 2.1). Through a communicator, the clients communicate with the server in a continuous and synchronous manner. We address the application layer QoS and transport protocols in the subsequent section.

### 2.2.1 Operating System

The operating system offers various services related to the essential resources, such as the CPU, main memory, storage. In the following sections, we discuss issues unique to real-time operating systems.

1. **Process Management**: Process management deals with the main processor resource [24]. The process manager maps each single process onto the CPU resource according to a specified scheduling policy, such that all the process can meet their requirements. To fulfill the timing requirements of continuous media, the operating system must use real-time scheduling techniques. Most real-time scheduling problems are addressed based on two basic algorithms: earliest deadline first (EDF) and rate-monotonic scheduling (RMS). In EDF scheduling, each task is assigned a deadline and the tasks are processed in the order of increasing deadlines. In RMS scheduling, each task is assigned a static priority according to the request rate. Specifically, the task with the shortest period (or the highest rate) gets the highest priority, and the task with the longest period (or the lowest rate) gets the lowest priority. Then the tasks are processed in the order of their priorities.

2. **Resource Management**: Resources in a multimedia server include CPUs, memories, and storage devices. Since resources are limited, a multimedia server can only serve a limited number of clients with the requested QoS. Therefore, resource management is required to manage resources so as to accommodate timing requirements. Resource management involves admission control and resource allocation. Specifically, before admitting a new client, a multimedia server must perform admission control test to decide whether a new connection can be admitted without violating performance guarantees.
already committed to existing connections. If a connection is accepted, the resource manager allocates resources required to meet the QoS for the new connection. Admission control and resource allocation schemes could either be deterministic or statistical, in the former case hard guarantees are made by making resource reservation. Statistical admission control provides statistical guarantees, by allocating resources such that there is higher utilization and temporary overload with a small percentage violation in QoS requirements.

3. **File Management:** The file system provides access and control functions for file storage and retrieval. There are two basic approaches to supporting continuous media in file systems. In the first approach, the organization of files on disks remains as it is for discrete data (i.e., file is not scattered across several disks), with the necessary real-time support provided through special scheduling algorithms and enough buffer capacity to avoid jitter. The second approach is to organize audio and video files on distributed storage like disk arrays. Under the second approach, the disk throughput can be improved by scattering/striping each audio/video file across several disks and disk seek-times can be reduced by disk-scheduling algorithms. Traditional disk-scheduling algorithms such as first-come-first-serve and SCAN do not provide real-time guarantees. Hence, many disk-scheduling algorithms have been proposed to address this issue. These include SCAN-EDF, grouped sweeping scheduling, and dynamic circular SCAN (DC-SCAN).

### 2.2.2 Storage System

For large scale media services, the storage system must have high-throughput, large bandwidth and should be fault-tolerant. Research has lead to the development of parallel video servers [12], Storage-area-networks (SAN) [8] [5], Network-assisted-Storage (NAS) [7].

1. *Increased throughput with data striping:* If an entire video file is stored on a disk, the number of concurrent accesses to that file are limited by the throughput of that disk. This dictates the number of clients that are viewing the same video file. To overcome
this limitation, data striping was proposed, where a multimedia file is scattered across multiple disks and the disk array can be accessed in parallel with better throughput.

2. *Increased capacity with tertiary storage:* The introduction of multiple disks can increase the storage capacity and the costs. To keep the storage costs down, tertiary storage (e.g., an automated tape library or CD-ROM jukebox) is usually added. To deploy streaming services on a large scale, a SAN architecture was proposed. A SAN can provide high-speed data pipes between storage devices and hosts at far greater distances than conventional host-attached small-computer-systems-interface (SCSI). The connections in an SAN can be direct links between specific storage devices and individual hosts, through fiber-channel arbitrated loop (FC-AL) connections; or the connections in an SAN can form a matrix through a fiber channel switch. With these high-speed connection, an SAN is able to provide a many-to-many relationship between heterogeneous storage devices (e.g., disk-arrays, tape libraries, and optical storage arrays), and multiple servers and storage clients. Figure 2.2 represents an SAN architecture.

![SAN Architecture Diagram](image)

**Figure 2.2 Storage-Area-Network**

Another approach to deploying large-scale storage is network assisted storage (NAS). Different from SAN, an NAS equipment can attach to a local area network (LAN) or a wide area network (WAN) directly. This is because an NAS equipment includes file system such as network file system (NFS) and can run on Ethernet, asynchronous transfer mode (ATM), and a fiber distributed data interface (FDDI). Figure 2.3 represents and
2.3 Application-Layer QoS Control and Transport Protocols

The lack of guaranteed services on the Internet makes deployment of multimedia services a challenging task and it is left to the application to provide the QoS requirements. Application-layer QoS control attempts to avoid congestion and maximize video quality in the presence of packet loss. The application-layer QoS control techniques include congestion control and error control. These techniques are employed by the end systems and do not require QoS support from the network.

1. Congestion Control: Multimedia applications need to perform congestion control because of the following reasons:

   - Bursty loss and excessive delay have a devastating effect on the video presentation quality caused due to network congestion. Thus, congestion-control mechanisms at end systems are necessary to help reduce packet loss and delay.

   - Transmission Control Protocol (TCP) is the dominant transport protocol in the Internet, the current stability of the Internet depends on its end-to-end congestion control, which uses an Additive Increase Multiplicative Decrease (AIMD) algorithm. End-to-end congestion control of best-effort traffic is required to avoid the congestion collapse of the Internet [6]. However TCP is unsuitable for multimedia, since the flow control and windowing schemes destroy the temporal relation between the
media packets, moreover reliable delivery is not required for media streams which can tolerate some packet loss. On the other hand user datagram protocol (UDP) is not directly suitable as it lacks congestion control mechanism.

Application level congestion control is provided over UDP by performing rate control and rate shaping. Rate control attempts to minimize the possibility of network congestion by matching the rate of the media stream to the available network bandwidth, (i.e) by determining the sending rate. Current research in rate control techniques include source based, client based and hybrid rate control schemes.

- **Source Based Rate Control**: Under the source-based rate control, the sender is responsible for adapting the video transmission rate. Typically, feedback is employed by source-based rate-control mechanism. Based upon the feedback information about the network, the sender could regulate the rate of the video stream. The source-based rate control can be applied to both unicast [25] and multicast [2].

- **Receiver Based Rate Control**: Under the receiver-based rate control, the receivers regulate the receiving rate of the video streams by adding/dropping channels while the sender does not participate in rate control [26]. Typically, receiver-based rate control is used in multicasting scalable video, where there are several layers in the scalable video and each layer corresponds to one channel in the multicast tree.

- **Hybrid Rate Control**: Under the hybrid rate-control, the receivers regulate the receiving rate of video stream by adding/dropping channels, while the sender also adjusts the transmission rate of each channel based on feedback from the receivers. Rate shaping tries to match the rate of precompressed bitstream to the target rate constraint. A rate shaper (or filter), which performs rate shaping, is required for the source-based rate control. The rate shaper would need to match the stored pre-compressed video rate to the available bandwidth in the network. Rate shaping techniques include layer-dropping, frame dropping and re-quantization.
2. **Error Control:** Forward error correction (FEC), retransmission, and Error Concealment are currently studied error control schemes. In the FEC scheme redundant information is added so that the original message can be reconstructed in the presence of packet loss. Error concealment is performed by the receiver when packet loss has occurred, the receiver attempts to conceal the loss by spatial and temporal interpolation techniques, to make it less displeasing to the human eyes.

2.4 *Continuous Media Distribution Services (CMDS)*

Internet's lack of QoS support has lead to the implementation of CMDS. These services are designed to provide QoS and achieve better efficiency for streaming media over the Internet.

- **Network Filters:** A congestion control technique, network filtering aims to maximize video quality during network congestion. A filter at the video server can adapt the rate of video streams according to the network congestion status. However, the video server may be too busy to handle the computation required to adapt each unicast video stream. Hence, the service providers may like to place filters in the network [9].

- **Application level Multicast:** The design of the Internet is well suited for point-to-point applications like e-mail, file transfer and Web browsing, but fails to support large scale content delivery like streaming media multicast. IP multicast is capable of providing efficient multipoint packet delivery, by ensuring only one copy of the original IP packet is transported along any physical path in the IP multicast tree. However, with a decade of research there are still many barriers in deploying IP multicast. These problems include scalability, network management, deployment; and support for higher layer functionality (e.g., error, flow and congestion control). To address these issues application-level multicast mechanism was proposed [15]. Application-level multicast is aimed at building a multicast service on top of the Internet. It enables independent content delivery service providers, Internet service providers, or enterprises to build their own Internet multicast networks.
• **Content Replication**: is an important technique for improving the scalability of the media delivery system, here the goal is to place the content close to the client so as to achieve reduced load on repository servers, reduced bandwidth consumption on the network links, reduced latency for the clients and increased availability.

1. **Mirroring**: Mirroring is to place copies of the original files on other machines scattered around the Internet. In this way, clients can retrieve multimedia data from the nearest duplicate server, which gives the clients the best performance. However for large repositories mirroring would not be feasible and moreover mechanisms for establishing dedicated mirrors are expensive, *ad hoc*, and slow.

2. **Caching**: is based on the belief that different clients would load many of the same contents, so a local copy of the most popular content is maintained and the client receives the local copy in case of cache hit for the file. In the case of a cache miss, the file is obtained by the cache from the server, stored in the cache and a copy sent to the client.

2.5 **Motivation for Distributed Streaming Media Architecture**

In this section we present the different streaming architectures, and describe the need for a distributed streaming media architecture. In the naive approach of media streaming, a single repository stores all the media streams of interest to serve large number of geographically distributed population. However such a scheme is limited by the repository resources and link bandwidth capacity from the repository to users. Proxy caching alleviate the problem to certain extent by bringing the content closer to the client and relieving certain amount of load of the central server. Proxy caching is a distributed streaming architecture, with information shared only between the repository and the proxy cache, no information is shared between peer caches. Some of the recently studied proxy architecture are described below.

• **Video-Staging and Selective Caching**: Zhang et al. [26] developed video staging, related video delivery technique that retrieves only a portion of a video stream from the central
video server across the backbone network. Miao et al. [14] proposed selective caching, where a few frames are cached based on the bursty nature of the video. Video-staging and selective caching provide robustness of the video stream against network congestion.

- **Proxy-prefix caching**: Sen et al. [23] proposed a prefix caching scheme for multimedia streams. The idea is to cache the first few segments (i.e. prefix) of popular streams at a proxy close to the clients. The cached data is used for work-ahead smoothing and it also reduces the start up latency. The scheme described usually supplements most of the other streaming architectures.

- **Caching Multicast Protocol** (CMP) [11] uses active routers to cache blocks of media data. In CMP, video blocks are stored as they pass through the active routers. When a cache overflow occurs, replacement algorithms are adopted to accommodate the new block. Since a block travels through several routers before reaching the client, multiple copies of the same block are stored in various routers. These copies are used to service subsequent requests arriving at the active router. CMP makes good utilization of the bandwidth resources and performs well in the case where a small number of videos are more popular than the others.

- **Self-Organizing Cooperative Caching Architecture (SOCCER)**: Hofmann et al. [10] described the SOCCER architecture, which essentially consisted of cooperative proxies used to deliver multimedia streams over the Internet. The architecture aimed at load distribution and coordination among the proxies, by dividing a video into blocks and each of the proxies storing a certain set of the video blocks. A distributed protocol is used to maintain the directory information of the video blocks. The QoS parameters are dependent on the accuracy of the directory, an inaccurate directory can cause interruptions during video playback. The overhead of maintaining an accurate directory is large and is dependent on the number of segments, videos and the number of proxies.

- **Reza’s Architecture**: Rejaie’s [22] architecture adopts layered encoding scheme and provides support for client heterogeneity. In the proposed architecture, each layer is further
sub-divided into blocks and the blocks are cached based on the popularity of the layer. The base layer has the highest popularity in this scheme and hence the base layer blocks are initially cached before the subsequent blocks get cached.

In the above described architectures, with the exception of Reza's architecture, SOCCER, Video Staging and selective caching fail to address the client heterogeneity aspect of media streaming. In Reza's architecture, the proxy-caches do not share data and the capacity is limited to a single cache. Also the cache is vulnerable to faults and thus is not resilient. To address the issue of robustness and large scale video delivery system we consider a distributed architecture with a set of servers which would share the video streams among themselves.
CHAPTER 3. PROPOSED DISTRIBUTED MEDIA ARCHITECTURE

In the proposed architecture, we consider a distributed set of servers (these could be a set of coupled proxy servers). The layers or descriptions (layers correspond to layered encoding; descriptions correspond to MDC) of a particular media stream are distributed among the set of servers. This distribution of the layers or descriptions would inherently aid in load balancing the servers. The popular media streams would have some of the layers or descriptions replicated on the servers to reduce overload on the servers and also to account for fault-tolerance in the system. The use of scalable encoding scheme for storage of media streams would support in handling client heterogeneity. Figure 3.1 provides a representation of the proposed distributed streaming media architecture. Each server is responsible for servicing a small group of clients and the associated server would be referred to as the coordinator. In the figure media stream A is composed of layers/descriptions (a1, a2, a3, a4) and each layer/description is placed on a different server. A request from a client is characterized by the requested media stream and the quality parameters (the minimum and the maximum qualities desired).

1. Service_Algorithm(): The coordinator would perform the Service_Algorithm() to handle a client's request. The algorithm is as follows.

Service_Algorithm(Movie m, Quality min, Quality max)
    if (Is_Buffer_Service_Available(Movie, min, max))
        Batch the request
    else if (Admission test())
        Request_Servers()
    else
        return Service Not Available
2. **Is_Buffer_Service_Available()**: For each service currently being served, the coordinator would maintain a circular buffer. This would help in batching requests that would fall in the buffer time frame. The size of the buffer would have to be determined by the popularity of the media stream. Also, the buffer would aid in performing VCR functionalities to an extent (governed by the buffer size). If a request fails to get batched to the existing buffers, the coordinator performs **Admission_test()**.

3. **Admission_test()**: The coordinator would need to ensure the availability of resources at the server for serving the stream, these include the ability to allocate the buffer resources, and the statistical guarantees of the network bandwidth. If the server **Admission_test** succeeds the **Request_Servers()** is performed.

4. **Request_Servers()**: Coordinator server would have to determine the servers in the set that are capable of serving the request. To achieve this the coordinator multicasts the request to all the servers in the set. The server (such a server would be referred to as a proxy) upon receiving a request from the coordinator would perform the necessary admission test, in the presence of the requested layer/description. If the admission test succeeds the proxy notifies the coordinator of its willingness to serve. Once the minimum quality parameters are assured for the coordinator, the buffers are allocated and the streams are requested from the respective servers.

```plaintext
Request_Servers()
    Multicast_Request()
    if(Min_Feasible_Servers())
        Synchronize_the_Servers
        Stream the media
    else
        Service Not Available
```

Consider a scenario where client cl requests for video stream B with a minimum of one and a maximum of three layers/descriptions. In the case of layered coding, the minimum request
Distributed Architecture to Support Media Streaming

Figure 3.1 Distributed Media Architecture

quality is the base layer (i.e., b1) and the maximum is \((b1, b2, b3)\). In multiple descriptive coding, one out of \((b1, b2, b3, b4)\) would satisfy the minimum quality and three out of \((b1, b2, b3, b4)\) would satisfy the maximum quality. For the request \(p1\) would perform the operations of the coordinator, in the absence of a buffer to batch the current request, it would multicast the request to the members of the set. Having multicasted the request, the coordinator would wait for response from the members. In this case \(p1, p3\) and \(p4\) would have to determine their ability to serve the request. If the minimum quality parameters for the request are met, the coordinator requests the proxies for transmission of the streams and synchronizes the streams before they are transferred to the client.

3.1 Coordinator and Proxy

3.1.1 Coordinator State Diagram

In this section we describe the states through which the coordinator transitions for serving a client's request.
1. Normal: A state where the coordinator waits for a request from any of its client.

2. Admission Test: Upon receiving a request the coordinator moves to the admission test state and multicasts the request to the members of the mesh. The coordinator waits until a 1 is received from the proxy servers, that are capable of serving or would time out after a certain time limit.

3. Synch. Wait: Once the coordinator proxy receives the minimum acknowledgments required to service the request, it enters the Synch. Wait state. During this state the coordinator requests the proxies that have acknowledged to start streaming and a circular buffer is maintained at the coordinator to synchronize the different layers that are transmitted.

4. Streaming: Once the buffer is sufficiently full (i.e. a sufficient portion of the layers have been synchronized), the coordinator would start the transmission to the client. The connection is terminated if the client requests to close the connection or if the streaming is complete. During this period it is possible that there might be additional streams that might join the connection, there by improving the quality.
Any of the servers in the set could behave as a proxy server if they receive a request for service.

1. **Normal**: A state where the proxy waits for a request to be received, so as to provide a service.

2. **Test State**: Once a multicast request has been received, it would enter the **Test state** and would perform the necessary admission tests to test for the possibility of serving the request.

3. **Streaming Wait**: Once the admission test has been successful, the proxy moves over to the **Streaming Wait** state. In this state the proxy awaits a request from the server for transmission of the media stream.

4. **Stream State**: Once the Coordinator has requested to start streaming the proxy begins transmission. The period during which the stream is transmitted is called the
Stream State.

5. Stream StandBy: If the coordinator fails to request for the transmission of the stream, a timeout event occurs and the proxy moves over to the Stream StandBy state. In case the coordinator makes a delayed request for transmission of media. The proxy might need to perform a re-admission, since only in the Streaming Wait state the resources associated with the proxy are guaranteed. Once the proxy moves out from this state to the Stream StandBy there is a possibility that the resources are no longer available.

6. Wait: A proxy in this state implies that the proxy is running short of resources and is unable to service the request immediately. In such a situation, the server waits for some stream to leave and would once again perform the admission test. If there is no stream that leaves for a certain duration of time, the connection gets closed. The proxy in the Stream State could also enter this state, if the stream has been demoted and the stream is no longer in service. Streams can be demoted in a case where the proxy is experiencing overloads and there exist streams that are being serviced above the minimal quality requirements.

In the architecture described above the distributed nature of the media streams ensures the load balancing of the servers and the network links. In the event of a server failure, the clients would be associated with a new coordinator in the set. In the subsequent chapter, we perform simulations to study the benefit of distributing the layer/descriptions across the servers. We study the effect of multiple servers serving a media stream as against a single server serving the media stream.
CHAPTER 4. PERFORMANCE STUDIES

In order to evaluate the effectiveness of streaming media to a client from distributed servers, as against a single server, we performed simulations studies on ns-2 version 2.1b8a [17]. We do not implement the entire architecture as described in Chapter 3, but rather we study the performance implications of serving a client from multiple servers as against a single server, which would justify the proposed architecture.

The implementation of the distributed server architecture was performed on ns-2 version 2.1b8a. NS is a discrete event simulator written in C++ and OTcl (interpreter based front-end). While the core protocols are implemented in C++, OTcl provides an interface for setup, configuration and manipulation of the C++ objects.

We have implemented a VDSAgent class in C++, which simulates the behavior of client, server or an coordinator. When a client generates a request, a quality object is created and is associated with that particular request and the object keeps track of the quality being received. The client maintains a circular buffer and a player reads of this buffer in fixed intervals of time. This simulates the client reading of a fixed number of frames from the buffer (like 30 frames per second). So the quality perceived by the client is the number of blocks available in the buffer at the time of playback.

Each client is associated with a nearest coordinator and the request generated by a client is forwarded to the coordinator, the coordinator would look up the distribution table and determine the servers that are capable of serving the request and the clients request is forwarded to the respective server. In the current implementation of admission control, we assume that the server has infinite capacity and admit all streams. Once admitted the server streams the video file at a constant bit rate. In the rest of this chapter, we present the performance metrics
defined for multiple descriptive coding and layered coding and the simulation parameters.

4.1 Performance Metrics

4.1.1 Multiple Description Coding

In this section we present the performance metrics for evaluating the streaming of multiple descriptions to a client. Figure 4.1 shows a sample state of the client's circular buffer, with the location of the player. We try to capture the average quality perceived by the client in distributed model and single server model.

![Circular Buffer for MDC](image)

Figure 4.1 Circular Buffer for MDC

1. **Quality Perception:** The quality perceived by a client is a function of the number of packets received by the client by playback time. For multiple descriptions, since the quality is a proportional to the number of descriptions received, the quality perceived by client is

\[
q_{\text{request}} = \sum_{k=1}^{L} \sum_{j=1}^{d} q_{ij}
\]  

(4.1)
where

\[ q_{ij} = \begin{cases} 
1, & \text{if block } q_{ij} \text{ arrived before playback} \\
0, & \text{if block } q_{ij} \text{ has not arrived by playback time}
\end{cases} \]

The percentage quality received is

\[ Q_{\text{request}} = \frac{q_{\text{request}}}{\text{Total quality}} \tag{4.2} \]

where

\[ \text{Total quality} = \sum_{i=1}^{d} \sum_{j=1}^{d} q_{ij} \tag{4.3} \]

and \( q_{ij} = 1, \forall i, j \)

2. **Fairness**: We also measure the fairness of the service across the clients by the evaluating the standard deviation in quality that is observed by the clients for a set of requests \( n \).

\[ \sigma_n = \sqrt{\frac{\sum (q_i - \bar{q})^2}{(n - 1)}} \tag{4.4} \]

3. **Jitter Characteristics**: Jitter is the difference between the maximum and minimum delays for packet transmission observed by the client. The jitter determines the buffer size required by the client. A large value indicates the need for a larger buffer size at the client side, to account for the discrepancies in the delays. Jitter produces unpleasant viewing, similar to that of lost packets. To study the jitter characteristics, we time-stamped the packets at the server and the client measured the delay caused due to transmission. Jitter is recorded for each request and the average for the entire simulation is taken to determine the jitter characteristics for the architecture. In jitter measurements, a lower jitter value implies, a smaller buffer size at the client, while a high jitter values would mean a large buffer and playback latencies. Consider a case where majority of the packets were lost, but amongst the packets that arrived, the transmission delay differed marginally, then the jitter would be small and would imply lesser buffer requirement and better quality. This does not give a right perspective, so the jitter is observed only where \( Q_{\text{request}} \geq q_{\text{threshold}} \) i.e the jitter parameters are considered only for requests which showed acceptable quality determined by a threshold.
4. **Media Continuity**: The quality perception metric does not capture the smoothness experienced by the client, to capture this we determine the number of breaks observed by the client for both single server streaming and distributed streaming. A break is defined as a section where the number of descriptions available are less than $ND$; where $ND$ is a parameter which determines the number of descriptions required for acceptable viewing. In Figure 4.1 $b_1b_2$ represents a break for $ND = 2$. The number of breaks tries to capture the continuity in the quality experienced by the client. A large number of breaks would mean unpleasant viewing experience, due to a large variation in quality.

4.1.2 **Layered Coding**

The quality perceived for layered coding is based on the presence of the previous layer. Since the encoding is a hierarchical based encoding and absence of the base layer renders the other layers unusable. Here base layer has the highest priority and the subsequent layers have decreasing priority, to capture this aspect weights were associated with each layer and the weight for the base layer is the highest. Hence the quality perception for layered coding is as follows

$$q_{\text{request}} = \sum_{k=1}^{l} \sum_{j=1}^{d} (w_l \times q_{ij})$$  \hspace{1cm} (4.5)

where

$$q_{ij} = \begin{cases} 
1, & \text{if block } q_{ij} \text{ arrived before playback and } q_{i,j-1} = 1 \text{ at playback.} \\
0, & \text{if block } q_{ij} \text{ has not arrived by playback or } q_{i,j-1} = 0 \text{ at playback.} \\
q_{ij} = 1, & \text{for } j=1 
\end{cases} \quad \forall j > 1$$

$q_l$ is the weight associated with layer $l$.

The percentage quality received is

$$Q_{\text{request}} = q_{\text{request}} / \text{Total quality}$$  \hspace{1cm} (4.6)

where

$$\text{Total quality} = \sum_{i=1}^{l} \sum_{j=1}^{d} (w(l) \times q_{ij})$$  \hspace{1cm} (4.7)
Figure 4.2 Circular Buffer for Layered Encoding

and $q_{ij} = 1, \forall i,j$

We cannot make a direct comparison for the quality observed for layered coding against MDC due to the weights associated with each layer in the layered encoding. Figure 4.2 represents the state of the buffer for the client obtaining layered encoded data, the snapshot illustrates that certain blocks are unusable due to the absence of lower blocks in the hierarchy. As in the case of MDC we observe the fairness among the clients for layered encoding.

4.2 Simulation Parameters

We consider a network similar to the NSFnet with a few extra nodes added to simulate the client behavior. The network topology is shown in Figure 4.3, the topology consists of 20 nodes, with 6 of these nodes acting as the servers and 14 are clients. The bandwidth of each link is assumed to be 45MB and the delay is varied between 10-35ms. Now the performance of the network would depend on the distribution of the movies among the servers. We consider a static distribution state as described below:
- **Distributed Descriptions and Layers**: In this scheme the layers/descriptions belonging to a particular media stream are on different servers. To achieve this the layers/descriptions are placed in a round-robin manner starting with the most popular movie. This distribution ensures that the descriptions/layers of a particular movie adopt an unique path from the server to the client. The distributed multiple description schemes is referred as DMDC, while the distributed layer scheme is referred as DL henceforth.

- **Single Server**: In this scheme all layers/descriptions belonging to a particular movie are placed on a single server. To achieve load balancing the movies are placed in a round-robin manner starting with the most popular movie. The single server media description scheme is referred to as SMDC and the single server layer scheme as SL.
Client request rate follows a Poisson distribution with a mean request inter-arrival time varying between 1.2 s to 0.4s. The load on the network is increased by decreasing the mean request inter-arrival time. The client making the request is chosen randomly from the set of 14 clients. Movie request pattern follows a Zipf distribution (a distribution observed by most web objects [3]) and the simulations where performed for different skew factors. The Zipf distribution with a high skew factor (0.7-0.9), indicates that certain movies are more popular than the rest, which is true in the case of the Internet. The various simulation parameters for the experiment are summarized in Table 4.1

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Default Value</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Time</td>
<td>10mins</td>
<td>Fixed</td>
</tr>
<tr>
<td>Approximate length of Movies</td>
<td>5 mins</td>
<td>Fixed</td>
</tr>
<tr>
<td>Average Request Inter-arrival Rate for the Entire Network</td>
<td>1.2s</td>
<td>0.4-1.2</td>
</tr>
<tr>
<td>Block Size</td>
<td>1.5KB</td>
<td>Fixed</td>
</tr>
<tr>
<td>Constant Bit Rate per layer</td>
<td>300KBps</td>
<td>Fixed</td>
</tr>
<tr>
<td>Player Advancement Time</td>
<td>1sec</td>
<td>Fixed</td>
</tr>
<tr>
<td>Player Jump Size</td>
<td>30</td>
<td>Fixed</td>
</tr>
<tr>
<td>Zipf Skew Factor</td>
<td>0.7</td>
<td>0.5-0.9</td>
</tr>
<tr>
<td>Number of Servers</td>
<td>6</td>
<td>Fixed</td>
</tr>
<tr>
<td>Number of clients</td>
<td>14</td>
<td>Fixed</td>
</tr>
<tr>
<td>Link Bandwidth</td>
<td>45MB</td>
<td>Fixed</td>
</tr>
<tr>
<td>Link Delay</td>
<td>10-35ms</td>
<td>Fixed</td>
</tr>
</tbody>
</table>
CHAPTER 5. RESULTS AND DISCUSSIONS

In this chapter, we present the results obtained for the multiple descriptive coding and layered coding for distributed server scheme and single server schemes. Each simulation has 3500 client requests for a media stream, with the client being chosen randomly and the first 1000 requests ignored as a warm up period, during the warm period the network reaches a steady state. The next 2000 client requests are observed for different performance factors with varying random seeds.

5.1 Multiple Description Coding

1. Quality Perception: Figure 5.1 we observe the average quality perceived by the clients for the single and distributed server architectures. From Figure 5.1 it can be seen that the average quality perceived by the clients, for an request inter-arrival rate of 1.2 seconds in DMDC model is better by a factor of 4% over the quality observed by a single server model. As the inter-arrival rate decreases (i.e., when the load on the network is increased), the network moves into a state of congestion collapse, where most of the packets transmitted in network are lost and as a result there is a significant drop in quality perceived. The network is considered to be in a stable state for request inter-arrival rate greater or equal to one. In the stable state region (request inter-arrival rate between 1 and 1.2 seconds) DMDC shows a better quality reception, since the probability of all the paths facing congestion is less likely compared to the SMDC service.

2. Fairness: Figure 5.2 shows the fairness among the various clients, the variance in quality experienced in the DMDC model is lesser compared to the SMDC model. It indicates that clients did not experience vast differences from the average quality delivered in the case of
DMDC model. In the case of SMDC scheme the overloading of a single server or a common link in the network accounts for the large variance in quality observed. From Figure 5.2 we could say to a certain confidence that there has been server and network load balancing in the distributed scheme as compared to single server scheme. As the request inter-arrival rate is decreased the variance in quality also increases, due to the increased congestion in the network, but DMDC is capable of showing lesser variance, implying greater robustness to loss experienced in the network.

3. Jitter Characteristics: The jitter characteristics are illustrated in Figure 5.3 for the different inter-arrival rates, DMDC model shows greater jitter values compared to the SMDC model. The large jitter values in DMDC model is attributed to the varied delays experienced by
the packets being streamed in from different servers. This would mean large buffer requirements for the coordinator, which might not be desirable in some cases. To mitigate the large buffer requirements, the servers that are used to serve a client to should all be either equally close or equally far off from the client. This should be one of the criterion, when deciding the servers that would be utilized for servicing a client request.

4. Media Continuity: The continuity experienced by a client (or the smoothness in viewing) is illustrated in Figure 5.4 and Figure 5.5. The maximum break lengths observed by a client for the DMDC model are lesser compared to SMDC, also the average number of breaks are less for DMDC when the network is not in a state of congestion collapse. Such breaks in streaming are attributed to congested links resulting in packet drops at the router.

Figure 5.3  Jitter Characteristics for Zipf Skew factor=0.9 (MDC)

Figure 5.4  Maximum breaks observed by the clients for a Zipf Skew factor=0.9 (MDC)
The above performance metrics were observed for different Zipf skew factors. The results observed for different skew factors were similar, this is due the distributed manner in which the movies were placed, thereby achieving load balancing among the servers. Figure 5.6 presents the fairness metric for different skew factors, showing consistent results for different skew factors.

5.1.1 Effect of Distribution of Media Descriptions

In order to study the effect of distribution of the descriptions among the servers, the descriptions were fully distributed (with each description being placed on a different server), partially distributed (each server possessing two descriptions) and all descriptions placed on
a single server. Figure 5.7 compares the quality perceived and Figure 5.8 the fairness metric for each of the three cases. The partial distribution scheme shows performance equivalent to distributed server and to an extent better performance. This indicates that the multiple path based distribution does help, but doesn’t buy much after a certain threshold.

In a DMDC model, each descriptions follows a distinct path to the client, and it is less likely that all the paths would be experiencing loss and hence the improved quality experience and fairness among the clients. The internet experiences bursty losses, the quality from a single server is likely to be degraded due to such bursty losses. During such bursty losses, the possibility of loosing more descriptions is high in SMDC model compared to the multi-path method of DMDC. Moreover the distribution of the descriptions across the servers, achieves the load balancing of the servers as can be inferred from the fairness criteria. To further improve the quality observed we would need to implement server based admission control and dynamic rate control at servers, so as to prevent congestion collapse.

![Quality Perception for a Zipf Skew factor=0.7](image)

Figure 5.7 Quality Perception for a Zipf Skew factor=0.7

### 5.2 Layered Coding

The quality and the fairness metrics for DSLC and SSLC do not bear any relation as observed in Figure 5.9 and 5.10, i.e., both the distributed server and single server scheme show similar results. For layered coding, we would need to ensure the guaranteed delivery of the base layer and dropping priority should be set for the subsequent layers. Such a priority based
transmission of the individual layers is possible in Differentiated Services, that is currently being developed by the IETF. Differentiated Services supports different classes of services within the Internet in a scalable and flexible fashion. Scalability is achieved by maintaining per-flow state at the edge routers, where the number of flows are relatively small and the router is capable of managing the complexity and resource requirement. The base layer would be provided with the premium service class, while the enhancement layer with assured service class. In such a scheme, the delivery of the base layer has higher priority over the enhancement layers.

To ensure the reliable delivery of the base layer, the base layer could be replicated on multiple servers and multiple copies of the base layer could streamed in from different servers.
depending on the reliability of the communication-links. Such a scheme tries to achieve robustness against the network losses at the expense of increased bandwidth utilization. The other technique would be to allocate sufficient buffer at the client side and adopt a acknowledgement based scheme. To ensure the reliable delivery of the more important layers.
CHAPTER 6. CONCLUSIONS AND FUTURE WORK

Media distribution architectures have been evolving over the Internet. We have proposed a distributed architecture for streaming media, which addresses the QoS requirements, client heterogeneity, scalability, fault-tolerance by making efficient resource utilization. The architecture is inherently capable of handling client heterogeneity and scalability, while fault-tolerance has been incorporated by distributing a single media stream across the servers, i.e., by having each description of a media stream to be placed on a different server.

The studies on the architecture have shown that the distributed set of servers using multiple descriptive coding scheme provides better quality characteristics over a single server scheme. The architecture shows robustness due to multiple paths adopted for the delivery of each description. However, in the absence of reliable delivery technique for the base layer over the current Internet the distributed server scheme is not suitable for layered coding due to hierarchical dependencies between different layers. The architecture balances the load across the servers and communication links and as a result is highly scalable. When the set of servers are viewed as proxy caches, the over-all storage capacity is increased due to the cooperative nature of the servers.

6.1 Future Work

The proposed distributed architecture needs to be evaluated in the presence of background Internet traffic. Under such workload conditions, the distributed multiple descriptive coding model would show greater robustness, since Internet traffic is usually short-lived and bursty. The multipath delivery mechanism would prevent all the descriptions being lost at the same instance of time. However introducing the background traffic in NS-2 simulations actually
increases the simulation run-time to an great extent. As a future research, the following aspects for the architecture need to be explored.

1. **Analytical Modeling:** There is a need to study the proposed architecture using an analytical model to determine the cardinality of the set of cooperative server and the number of layers or descriptions to be provided at each server.

2. **Quality Adaptation:** Quality adaptation schemes based on the determined bandwidth have to be formalized. For performing adaptation, any server need to have two or more descriptions.

3. **Cache Replacement:** There is a need to determine the cache replacement algorithms in the case where the servers are considered to be a set of proxy caches.

4. **Server Fault-Tolerance:** Although the distributed architecture is inherently fault-tolerant, there is a need to address the situation when the coordinator fails while serving a client.
Bibliography


