Distributed Continuous Media Streaming – Using Redundant Hierarchy (RED-Hi) Servers

Mohammad Ahmed Shah

Submitted to the
Computer Engineering Department
in partial fulfillment of the requirements for the Degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
January 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Işık Aybay
Supervisor

Examining Committee

1. Prof. Dr. Işık Aybay
2. Prof. Dr. Mehmet Ufuk Çağlayan
3. Prof. Dr. Turhan Tunalı
4. Assoc. Prof. Dr. Muhammed Salamah
5. Asst. Prof. Dr. Gürcü Öz

ABSTRACT

The first part of this thesis provides a survey of continuous media serves, including discussions on streaming protocols, models and techniques. In the second part, a novel distributed media streaming system is introduced. In order to manage the traffic in a fault tolerant and effective manner a hierarchical topology, so called redundant hierarchy (RED-Hi) is used. The proposed system works in three steps, namely, object location, path reservation and object delivery. Simulations are used to show that the scheme, proposed here, performs better than the traditionally used multimedia transmission models in terms of various parameters. Results show that this scheme gives better transmission rates and much lower blocking rates. Furthermore it exhibits higher fault tolerance and greater load balancing of the streaming tasks among the servers of the streaming system.

Keywords: Distributed Multimedia, Video Streaming Object Location, Object Delivery, Load Balancing, Fault Tolerance.

ÖZ

Bu tezin ilk kısmında sürekli medya içerik sağlayıcıları ile ilgili çalışmalar ile akış protokolleri, model ve teknikleri üzerine tartışmalaryer almaktadır. İkinci kısımda ise yeni bir dağılımlı medya akış modeli tanıtılmıştır. Bu modeled hata toleranslı ve etkili bir trafik yönetimi için "fazlalık hiyerarşisi" olarak adlandırılan bir hiyerarşik topoloji kullanılmıştır. Önerilen sistem içerik yer belirlemesi,, yol belirlemesi ve içerik dağıtımı olmak üzere üç adımda çalışır. Tezde yapılan simülasyonlar, sunulan yeni modelin çeşitli parametreler açısından geleneksel multimedya iletim modellerinden daha iyi işlediğini göstermektedir. Elde edilen sonuçlar, bu modelin daha yüksek hata toleransı ve çok daha düşük engelleme hızlarıyla daha yüksek iletim hızları sağladığını göstermiştir. Ayrıca akış sistemlerinin sunucuları üzerinde akış görevlerinde daha büyük yük dengeleme sağlanmıştır.

Anahtar Kelimeler: Dağılımlı multimedya, video akış sistemi, içerik belirleme, içerik dağıtımı, yük dengeleme, hata toleransı

To My Beloved Parents…

ACKNOWLEDGMENTS

I want to thank Dr. Işık Aybay, my supervisor. He was a consistent and constant source of motivation for me. It would not have been possible to finish this thesis without his guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

RED-Hi	Redundant Hierarchy
ACDSR	Average Communication Delay Of Successful Requests
ANCMSR	Average Number Of Control Messages Of Successful Requests
ANHSR	Average Number Of Hops Of Successful Requests
ANTNSR	Average Number Of Traversed Nodes Of Successful Requests
ATD	Average Transmission Delay
BR	Blocking Ratio
CBR	Constant Bit Rate
CDN	Content Delivery Network
CMS	Continuous Media Servers
CO	Central Offices
DCMS	Distributed Continuous Media Servers
GPSS	General Purpose Simulation System
HTTP	Hyper-Text Transfer Protocol
ISP	Internet Service Providers
LRU	Least Recently Used
POP	Points Of Presence
QoS	Quality-Of-Service
STB	Set-Top Box
TCP	Transmission Control Protocol

VBR	Variable Bit Rate
VBRBS	Variable Bit Rate Bandwidth Skimming
VoD	Video-On-Demand
WWW	World Wide Web

Chapter 1

INTRODUCTION

Multimedia systems have been widely researched in recent years and as a result they have been used in a number of applications that cater to the needs of areas as diverse as distance learning to internet television and as demanding as teleconferencing, and video-on-demand. Performance of most of these applications is highly dependent on the streaming technique used to deliver multimedia files to their respective clients. Streaming, in contrast to downloading, allows early commencing of multimedia content playback; without waiting for the completion of delivery of the multimedia file from the media server to the client. Downloading approach, on the other hand, would wait for the entire file to download before starting the playback. In large-scale streaming multimedia systems, routing algorithms used for streamed packets is as important as the streaming techniques used for content delivery. Finally, in today's competitive environment, the clients demand the content delivery system to be reliable and fault tolerant.

Basically, multimedia streaming applications can be classified into three categories: on-demand streaming, such as video-on-demand; live streaming, such as Internet television; and real-time interactive streaming as in video conferencing and on-line gaming. This thesis mainly focuses on topological design and related performance issues involved in routing of multimedia content in a fault tolerant fashion, over graphically dispersed networks.

Distributed multimedia streaming necessitates a set of servers, a network, and a set of clients. Section 1.1. reflects on the respective functionalities of these three basic building blocks of a multimedia streaming system. Quality of service in streaming multimedia dictates scalability, reliability and fault avoidance / tolerance among other critical performance issues such as rate control and congestion control. This thesis proposes a distributed multimedia content delivery system and as such particularly concerns with scalable streaming of on demand multimedia data to a large number of geographically distributed clients using scalable delivery protocols. Section 1.1 details a discussion on multimedia streaming. Section 1.2 summarizes the contributions of the thesis to the existing scalable content delivery systems for multimedia objects, and Section 1.3 provides the overall structure of the thesis.

1.1 Review on Distributed Streaming

Distributed streaming has a variety of applications and as such there are numerous papers on this topic. Tsai et al. [1] describes the efficiency and applicability of distributed video content management in a surveillance system, with a discussion on the development of an IP-based physical security following on ONVIF standard. This ICL ONVIF middleware uses iSCSI distributed network to establish the distributed surveillance system. Their target is to provide multimedia content processing with load balance control and to build a distributed network storage space surveillance system.

Gramatikov et al. [2] proposes a hierarchical network system for VoD content delivery in managed networks, which implements a redistribution algorithm and a redirection strategy for optimal content distribution within the network core and optimal streaming to the clients. Their system monitors the state of the network and the behavior of the users to estimate the demand for the content items and to take the right decision on the appropriate number of replicas and their best positions in the network. The system's objectives are to distribute replicas of the content items in the network in a way that the most demanded contents will have replicas closer to the clients so that it will optimize the network utilization and will improve the users' experience. It also balances the load between the servers concentrating the traffic to the edges of the network.

Jin X. [3] presents a scalable distributed multimedia service management architecture using XMPP. They study the XMPP and revealed the limitations of related multimedia service management models. Furthermore they describe a scalable distributed multimedia service management architecture along with a video conferencing system case using the XMPP model.

Song et al. [4] details a system that models a layer based system for managing and presenting video on-demand to users of Linux system. The purpose of their work is to use a P2P based architecture and an application embedded in Linux to provide HD video to end-users for their consumption.

Bo Tan and Laurent Massoulie [5] address the problem of content placement in peer-to-peer systems, with the objective of maximizing the utilization of peers’ uplink bandwidth resources. They identify some fesiable content placement strategies that can maximize system performance under certain constraints.

Applegate et al. [6] present an approach for intelligent content placement that scales to large library sizes (e.g., 100Ks of videos). They formulated the problem as a mixed integer program (MIP) that takes into account constraints such as disk space, link bandwidth, and content popularity. To overcome the challenges of scale, they employed a Lagrangian relaxation-based decomposition technique combined with integer rounding. They also present a number of strategies to address issues such as popularity estimation, content updates, short-term popularity fluctuation, and frequency of placement updates.

Brost et al. [7] developed light-weight cooperative cache management algorithms aimed at maximizing the traffic volume served from the cache, minimizing the bandwidth cost. Their focus is on a cluster of distributed caches, either connected directly or via a parent node, to formulate the content placement problem as a linear program in order to benchmark the globally optimal performance.

Zhang et al. [8] study the problem of maximizing the broadcast rate in peer-to-peer (P2P) systems under node degree bounds, i.e., the number of neighbors a node that can simultaneously connect to it is upper-bounded. They address the problem by providing a distributed solution that achieves a near-optimal broadcast rate under arbitrary node degree bounds, and over an arbitrary overlay graph. It runs on individual nodes and utilizes only the measurement from their one-hop neighbors, making the solution easy to implement and adaptable to peer churn and network dynamics. Their solution consists of two distributed algorithms proposed: a network-coding based broadcasting algorithm that optimizes the broadcast rate given a topology, and a Markov-chain guided topology hopping algorithm that optimizes the topology. they demonstrate the effectiveness of the poropsed solution in simulations using uplink bandwidth statistics of Internet host.

1.2 Contributions of the Thesis

This thesis is inspired from the work of Shahabi et al. [9] which proposed the use of a Redundant Hierarchy for content management systems of distributed video servers. This thesis uses their proposed hierarchy and integrates load balancing techniques along with the fault tolerant nature of RED-Hi [9]. The fault tolerant behavior of RED-Hi for DCMS has been demonstrated by Shahabi et al. on page 54 of their article [9]. They also showed that it may achieve load balancing. In this study, we illustrate and prove that by using RED-Hi together with proposed threshold level mechanism of popularity, it is possible to achieve load balancing and efficiency as well as dynamic placement of media content.

This thesis also provides a survey of continuous media serves, including discussions on streaming protocols, models and techniques. The main contribution is providing a novel distributed media streaming model. In order to manage the traffic in a fault tolerant and effective manner the hierarchical redundant hierarchy (RED-Hi) topology is used. The study introduces the use of popularity threshold integrated into a RED-Hi based DCMS content management system. The simulation results show that the proposed scheme performs better than the traditionally used multimedia transmission models in terms of different parameters and under various conditions.

1.3 Thesis Organization

The remainder of the thesis is organized as follows.

In Chapter 2, an analysis of subcomponents of Continuous Media Servers is presented.

The proposed DCMS architecture is presented in Chapter 3.

Chapter 4 contains a Petri-net model of the DCMS system.

Chapter 5 presents RED-Hi based load management policy.

The simulation tool used in simulations, the performance measures taken into account and the simulation parameters are discussed in Chapter 6.

In Chapter 7, we present the simulation results of the RED-Hi and Pure Hierarchy and also provide the performance comparison of RED-Hi and Pure Hierarchy.

Chapter 8 summarizes this thesis and gives an outline for possible areas that can emerge in future.

Chapter 2

CONTINUOUS MEDIA SERVER ANALYSIS

In order to develop a model for, and then simulate a Distributed Continuous Media Streaming System, it is essential to understand the key challenges in building such a multimedia content delivery system, especially the key component of the system, i.e. Continuous Media Servers (CMS). In this chapter, an analysis of subcomponents of CMS is presented. The building blocks of a multimedia system are presented with a brief discussion on challenges involved in building such a system. A limited discussion on tradeoffs, performance guarantees and Admission Control is also presented to understand the complexity of the system.

2.1 Streaming of Stored Multimedia

Three major components constitute a distributed multimedia streaming system: the server, the network, and the clients.

2.1.1 Server

A streaming server typically performs three tasks: process client requests, retrieve the requested media data and then transmit it into the network. While processing the client request, the server needs to parse it utilizing the CPU. Once the request is parsed, the server populates the in-memory buffers with the data requested by the client using disk bandwidth. Finally network interface bandwidth is required to transmit data into the network. The resources utilized by the server along the access path in order to satisfy a single media stream are collectively termed as “server channel”.

The allocated server channel may remain occupied over a varied length of time depending on the size of the media file. This allocation is typically long in terms of time because multimedia files are inherently large and their delivery takes time to complete. With increase in the demand, a server may need to support hundreds of thousands of requests at a given time. Indeed, if separate channels are dedicated for individual client requests, the channels at the disposal of the server will saturate very quickly.

One approach to manage the predicament of channel saturation caused by increased popularity of multimedia streaming applications is to simply increase server capacity by building a server cluster using thousands of low cost desktop machines. This approach alone, however, cannot yield a desirable solution to the saturation problem mentioned above, as it fails to address the potential bottlenecks that may incur at the server network access bandwidth i.e. at the interface between the server outward router and the network outside the system. Increasing network access bandwidth to meet the scaling demand of the system is not only costly but it also increases server network access bandwidth. So, it turns out that this is a temporary solution to a large resource problem.

2.1.2 Network

When the server sends the requested media file into the network, it is routed to the client site that initiated the request for the file in the first place. The media file is transported to the requesting client over the network from the server in form of packets. These packets of the media data flow through the network and during this flow, resources such as, processing power and buffer space at each router, and, bandwidth on each link is used. The resources utilized by the data packets during their flow through the network are collectively termed as the “network channel”.

The most prevalent method of streaming by current multimedia streaming applications uses unicast, i.e. each stream holds a distinct network channel. In a typical scenario, where many users access the same ‘popular’ file at the same time, multiple replicas of the same data packets flow over the same links at the same time through distinct channels which clearly wastes network resources. This may even cause bottlenecks in the network and saturate the network channel The data packets would not seize the flow in a short period of time, as multimedia files are typically very large and their uninterrupted streaming may last from a couple of minutes to even a couple of hours.

Unicast streaming may also cause a waste of server resources. To address the above problems of unicast, multicast can be used to efficiently deliver multimedia data packets from one server to multiple clients.

Multicast can be employed inside the network, such as IP multicast, or deployed at the end hosts, as application layer multicast. It is known that network layer multicast uses network bandwidth very efficiently. Nevertheless, it must be noted that it has problems such as router overhead, scalability, reliability, and security. The main reason behind network layer multicast having so many problems is that it is many-to-many in nature. This means that during an ongoing session, any participant can send and receive messages from all other participants. One way to overcome this shortcoming is by using a one-to-many communication model where during a streaming only a single server sends data to a large number of clients. Such applications motivate a new and simpler multicast service model called source-specific multicast (i.e., SSM) [10]. As there is only a single sender in source-specific multicast, issues such as group management, pricing, routing, and security can all be handled more easily and effectively in contrast to many-to-many type multicast.

As an approach to provide one-to-many along with many-to-many communication service, researchers have proposed application layer multicast technique [11, 12, 13, 14, 15], where the end hosts, instead of network layer routers, copy and forward data to their downstream hosts. This idea implements a virtual network or an overlay network across the end hosts in the system. In this approach, every virtual link from one host to another is a unicast path. A multicast distribution tree is created from server to all member nodes to provide a one-to-many multicast. After this, data is transmitted along the established distribution tree. During the transmission, the data is replicated and forwarded by the nodes at each branch point until it is received by every member node. This form of multicast (application layer multicast) does not require network support for one-to-many communication.

It must be noted that application layer multicast has its own shortcomings also, such as creating an efficient overlay network across all participating end hosts, and effectively maintaining an overlay network with changing network conditions [16, 17, 18].

2.1.3 Client

A client requesting a service from a multimedia streaming system needs special software such as Microsoft Media Player [19] or RealPlayer. Sometimes it also necessitates use of specialized hardware such as a set-top box (STB) to playback the requested resource i.e. multimedia file. The media file is typically composed of audio and video data and by nature has high storage and bandwidth requirements. The audio and video components of the multimedia files are usually compressed when stored or when they are delivered across the network. The compressed multimedia file needs to be decompressed at the client side before being rendered onto the screen. Video and audio data are bound by temporal limitations and are, by nature, delay-sensitive. This means that a particular piece of data belonging to a multimedia file must be received and displayed at a particular time in a fixed order. After passing of the time or the order, the received data becomes unusable. The current Internet supports only best-effort service. This means that it cannot guarantee any end-to-end delay bound, or any stability in terms of delays. The instable and unpredictable delay caused by network is called delay-jitter. At client side, a start-up delay of possibly up to tens of seconds is usually introduced to accommodate this very delay-jitter. This requires the need for a buffer to hold file data until it is played out.

2.2 Scalable Streaming of Stored Multimedia

Although many applications of media-on-demand are readily available [20, 21], maintaining and achieving quality in terms of reliability and scalability of multimedia streaming files remains a crucial, as well as practical problem and as such, remains to be an active research field. This section discusses the main approaches proposed to address this issue. Two prominent approaches, namely, content replication and caching and scalable delivery protocols are detailed in the following subsections. Both of these approaches are complementary to each other and are of nontrivial importance to successful implementation of a distributed multimedia streaming system. This thesis proposes a streaming multimedia system that merges these two approaches and provides an object placement, location and content delivery architecture that is reliable and fault tolerant.

2.2.1 Content Replication and Caching

In content replication and caching, policy popular files are copied close to end users. This is either accomplished through use of proxy caches, as it is usually done in World Wide Web, or is achieved using mirrors at distributed servers that maintain copies of the original files.

2.2.1.1 Proxy Caching

Local storage of web objects is usually placed in proxy caches so as to provide local storage of the resource requested by the client. All web requests are intercepted using these proxies. In case the requested object by the client is not available locally in the proxy cache, the request is forwarded to the respective server on behalf of the client. Once the requested data is received by the proxy server from the server, it forwards the same data to the client and if needed makes a copy of it in its local storage in order to satisfy subsequent requests for the same object locally. This approach reduces both the average response time of client requests and the traffic between the proxy and the servers. Many researchers have deliberated at the application of proxy caching to distribute continuous media files. One such approach is preloading prefixes where first several minutes of a media file are preloaded in the proxy servers. By maintaining prefixes of a suitable number of the most popular files into the proxy cache, the system attempts to reduce the start-up delay caused in multimedia streaming [22]. There is sufficient research done at determining the suitable number of the files to be prefixed as well as determining what constitutes a popular file. Furthermore, researchers have investigated optimal placement of media files across the proxies in a distributed multimedia streaming system. Related research shows that optimal placement usually comes from an all-or-nothing rule i.e. each file is either entirely stored in the proxies or is not stored at [23].

Noting that the capacity of a proxy server’s cache is physically limited, furthermore noting that the size of a continuous media file is typically very large, it is concluded that choice of cache replacement algorithm is an important factor in effective delivery of multimedia object in any distributed multimedia streaming system benefiting from proxy servers.

2.2.1.2 Content Replication

Content Replication deals with placing copies of multimedia content on servers. A Content delivery network (CDN) is deployed using servers that have copies of popular content throughout the system. Each request originating from a client is handled by the CDN. It first determines the server(s) that have the content requested and then directs the request to one that can satisfy the request at the minimum cost.

Many examples of CDN translated into application are readily available [24, 25]. Nevertheless it is necessary to deal with a few critical issues. One of these issues is content placement. Content placement deals with optimal placement of the content over the network. Another critical issue is how to direct client requests to the most appropriate server in a seamless and efficient manner. Yet another concern is maintaining the server and network status information and keeping track of the updates to the replicated file locations as they are propagated through the proxies. Finally, an important issue is communicating the most up-to-date content location information to all servers in a way that the correct information is available at the server when it is responding to a request [26, 27, 28, 29, 30, 31, 32].

2.2.2 Scalable Delivery Protocols

Employment of a scalable delivery protocol is yet another approach to deliver multimedia content over the network. In contrast to the classical method where a dedicated stream is separately used to deliver the content for every request made by the client, a scalable delivery protocol serves multiple client requests for a given identical file by using or dedicating a single server channel.

There are two main categories of scalable delivery protocols, namely: immediate service protocols [33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45] and periodic broadcast protocols [47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58].

Periodic broadcast protocols stem from the fact that the probabilities of media file accesses are highly skewed in their distribution [59]; and that the clients tend to be patient when they have to wait a small time, typically tens of seconds, for the beginning of a playback. In periodic broadcast protocol, the media server streams a number of the most popular (some 10 to 20) media files through the network channels. Clients that have requested a media file being served through these channels simply tune into the respective channels. The media files are segmented in increasing sizes to efficiently utilize the server bandwidth. The efficiency is achieved by first repeatedly broadcasting a very short segment of the video on a dedicated channel which clients can receive and playback without much playback delay. Later on, the subsequent segments are fetched while the initial short segments, and then the other already broadcasted segments are played back at the client end. The main novelty of the idea in periodic broadcasting protocol lies in its media segmentation, channel allocation, and the scheduling of the clients to the channels broadcasting the segments.

Periodic broadcasting channel is a scalable protocol, and gets its strength from the way it streams and broadcasts the media files. Nevertheless, it suffers from the following problems as in this protocol the server bandwidth usage is independent of the client request rate:

- It is highly inefficient for streaming media files that are not popular;
- The start-up delay that comes from the periodic nature of the protocol may be undesirable in some applications;
- The protocol does not address interactive usage of streams and has no support for functions such as fast-forward or rewind.

These downsides are addressed in a different category of scalable delivery protocols that tries to respond to client requests in a strictly reactive manner, called immediate service protocols. In this approach, as a response to every request, a new stream is created that delivers the requested media file from the very beginning. This makes it possible to get minimal start-up delay. In order to address the issue of scalability, the service protocol lets a client eavesdrop into other streams of the same media file that are being transmitted while storing the respective data in the client's local buffer. As soon as the client reaches the point where it has all the data prior to the point where it began to snoop on the previous stream, its own stream terminates, and the client continues receiving service from the original stream. Many immediate service protocols have been discussed in literature, including patching [36, 37, 38, 39], tapping [35], adaptive piggybacking [33, 34, 44], hierarchical stream merging (HSM) [40, 42, 43, 45, 46], and bandwidth skimming [41]. These protocols are more adaptable to seamless support of interactive functions such as forward and rewind as they re-allocate a new stream to the client making the interactive request. It is worth mentioning that these protocols do not require prior knowledge of the popularity of the media files being delivered. In comparison to periodic broadcast protocols, they work better for streaming lukewarm and cold media files, but are less efficient for hot media files.

Researchers have also proposed some designs that combine elements of these two approaches where the basic idea is to dynamically distinguish between popular and not-so-popular media files, and then use periodic broadcast to deliver popular files and immediate service to deliver the others [51]. In such implementations, the main hurdle is accurate identification of media files that are popular and then keeping track of their popularity in time, reacting appropriately to changes in file popularity.

2.3 Complex Multimedia

Multimedia data is usually compressed before being stored or delivered over the network. Multimedia data is generally made-up of audio and/or video data. Audio files are usually compressed using a constant bit rate (CBR) approach, where each time unit of an audio file requires the same number of bits to represent it digitally. Video files, on the other hand, may either make use of CBR or a variable bit rate (VBR) compression. Although it is easier to manage CBR video files and to transmit them over the network, they are not efficient in terms of quality where motion-intensive scenes are the object of compression. VBR, as the name suggests, compresses the files in a variable fashion. The idea is that compression will allocate bandwidth based on properties of different scenes where, naturally, high bandwidth is allocated for motion-intensive scenes. Such a schema allows maintaining higher quality across the complete video segments, as the peak rate of a compressed VBR file is usually two to three times as high as the average compression rate in VBR [60] yet; such files are hard to stream efficiently.

An approach termed as work-ahead smoothing has been proposed that attempts to smoothen a given variable bit rate file into a near constant bit rate stream, as such making the delivery across a network easier [61, 62 ,63]. This approach preempts the bandwidth use by allocating available network bandwidth during the delivery of the low bit rate portions of a media file to deliver data in high bit rate portions later. Using the smoothing technique may require a small start-up delay to ‘smooth’ the first several seconds of the stream. Then, work-ahead smoothing efficiently reduces latencies in peak rate, and those of the rate variability of stream.

Yet a different form of complex media is co-called composite media. With ease in authoring, editing and presenting, multimedia data is becoming more and more prevalent, benefiting from powerful tools readily available in the market. Composite media files that consist of a mix of audio, video, static images, and plain text are becoming more popular. This popularity will only increase in time; It does not take much imagination to visualize the Olympic games being broadcast live over the Internet, where a typical broadcast might consist of multiple multimedia sub-streams, including background sounds, video from particular events, close-ups, game statistics, and narrations.

The existing scalable delivery protocols assume that all and any type of media data consists of a single sequence of encoded data regardless of the type of the media files being delivered.

Although this linear approach to deliver media is sufficient for many existing multimedia applications, with ever increasing consumption of the Internet and at the same time, the linear increase in network capacity for new multimedia applications requires a need for providing higher interactivity and customization using more and more complex media structures. Today movies are limited to a linear structure because they make use of the broadcast technology used in theatres or on TV. Internet delivery of movies is relatively new and limited. In the future, Internet technology will enable new types of entertainment and educational video, such as multi-ending movies, where the clients will determine a suitable end to the movie they are watching by connecting to different threads of the story line, branching from certain points in different directions that are determined dynamically. Another example is a “virtual tour” where clients choose their own navigation path [64]. One can also imagine that in a future news-on-demand system, viewers in different cities may be able to watch the same national news followed by different regional news and different local news. Similarly, in a video-on-demand system, different viewers may be given localized and customized advertisements during movie breaks. Although these applications could use a collection of linear media files, for each story line in a multi-ending movie for example, it may be more efficient to recognize the structures inherent in the media threads and exploit them during delivery.

2.3.1 Network Bandwidth Bounds

The objective of a scalable delivery protocol is to deliver media files with bandwidth requirements that do not grow linearly with increase in client request rate, to put it in another way, as is in with periodic broadcast; it should be inversely proportional to startup delay at the client. A basic question is how slow this growth can be. Also of interest is the question that how close existing scalable delivery protocols come in terms of minimizing the bandwidth usage.

Previous research lists the necessary minimum server bandwidth used by streaming protocols that work with video-on-demand and guarantee a maximum start-up delay [65, 66, 67] as well as protocols that provide immediate service [43]. However any study that details the minimum network bandwidth requirement of scalable streaming protocols for on-demand systems could not been found in literature. Such an analysis is complicated by the fact that the network bandwidth requirement depends not only on the streaming protocol, but also on the network topology and on the multicast distribution tree that is employed.

2.3.2 Scalable On-demand Streaming of VBR Media

Most of the scalable delivery protocols proposed in prior work assume CBR media. However, video data would typically use a VBR compression in order to efficiently achieve uniform quality. Furthermore, in multimedia streaming applications such as news-on-demand and live broadcast, various media types (i.e., audio, video, text, and static images) may be combined in a composite multimedia presentation, also resulting in a VBR media file. Due to its intrinsic rate variability, VBR media cannot be managed and delivered as easily as CBR media.

Work-ahead smoothing techniques are often used to decrease the peak rate and to minimize the variability in the rate when streaming a VBR file. It has been established through literature that work-ahead smoothing actually reduces the potential benefits of scalable delivery protocols. This is because those protocols achieve bandwidth reduction largely through sharing the delivery of the later portions of a media file among multiple clients, while work-ahead smoothing moves data from these later portions to earlier portions that are less widely shared, in order to generate a smoother stream.

2.3.3 Scalable On-demand Streaming of Non-linear Media

Current highly-scalable content delivery services such as TV employ a broadcast model where end-users play a passive role in receiving the content. Internet delivery of multimedia file entwined with other modern advances enables many new opportunities. It is not farfetched to fathom customized and interactive multimedia application of future. This customization maybe one where the client picks their own ending for a movie that is streamed on-demand, in which the multimedia objects to be delivered include parallel sequences of data units, such as a branching set of video frames, where clients may select at chosen branch points which branch of video stream to pursue making the streaming process dynamic in nature. This means clients that are requesting the same media file potentially would receive variable sequences of data units, based on their own selections of the branch points. The currently available scalable delivery protocols are not capable of being directly used in delivering such dynamic streaming files.

There exist many approaches to scalable non-linear media delivery. Some of these approaches assume advance knowledge the path selected by the at every branch point, either by measurement of the overall client path choice frequencies in the respective system or relying on client classification or pre-selection. Others assume a priori knowledge and must achieve a suitable compromise between aggressive sharing of server transmissions, and client reception of data that turns out to be from a different path than the path the client will select. Researchers have looked into this type of scalable non-linear media delivery approach and analyzed the minimum required server bandwidth and associated overhead client data. This overhead is the data that the clients receive but do not use. Obviously, if a fairly accurate prior knowledge of the path client would select is available, then minimal server bandwidth usage can be achieved; this would also cause substantial savings in the overhead. In the absence of such knowledge, the client data overhead can still be greatly reduced at a relatively small server bandwidth cost, through control data that could potentially determine through look-ahead checks to make sure what data is on their path in future .

2.4 Multimedia System Components

This study aims at developing a DCMS system that can efficiently deliver multimedia data over a network with fault tolerance. In order to achieve this target, a systematic approach is required. This approach should consider all interactions that occur between many different subsystem components of a DCMS. This systemic control of the interactions is necessary in order to achieve the stringent performance restrictions inherent in multimedia data transmission.

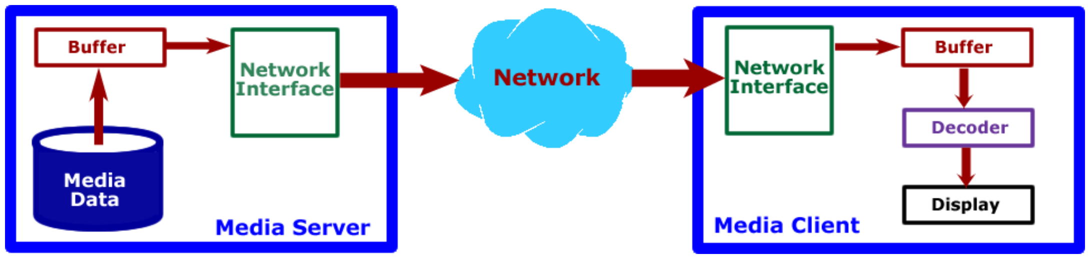

Figure 2.1: Multimedia System Building Blocks.

Figure 2.1 provides a generic illustration of a model of multimedia data delivery from server end to client end. At the server sidethe media data is available and is stored in an encoded format using storage hardware. Software on media servers are used to retrieve the media data from the disk and prepare it for the transmission over the network. A media application and a transport protocol are then used to stream the media data and deliver to the client hosts. At these client hosts, this transmitted media data is first buffered and is then decoded for the presentation application that provides the media for end user consumption.

If a difficulty surfaces in any single component of a DCMS for any reason, that single problem can cause degradation in the performance of the entire system. In the sections that follow, a more detailed investigation of some of these DCMS system components is presented.

2.5 Media Data

Multimedia comes from the union of multi and media where the word ‘multi’ specifically underlines the many forms of media which may or may not be of same type included in one data file. This means that possibly many forms of media should be authored, delivered, and presented as one unit. The many different types of media mentioned before include but are not limited to plain text, images, audio, or video data. It is possible to categorize the many different types of media into two multimedia data delivery classifications.

The first classification can be termed as discrete media. Discrete media would refer to that type of media which has no overt constraints on its timing for presentation [68] [69]. As an example let’s take retrieving an image from a web server to be displayed in a web browser. A browser presenting the image can take different time units to display this image because the network bandwidth availability may be different at different times of the presentation. This has nothing to do with the decoding process that takes place after the image has been downloaded. The download time can be a fraction of a second or maybe hundreds of seconds. The image size and the available network bandwidth would determine the time needed for download. This delay should be as short as physically possible, while making sure that the time it takes to retrieve the image data is long enough not to distort or damage the data itself while receiving, rendering, or displaying. Once all the data is correctly displayed, it is sure that the request was satisfied successfully. This means that a restriction that would limit the time or delay for an image media does not exist in terms of its presentation.

Another type of media data is continuous media. This type of media has stringent presentation requirements for its timing. These requirements are embedded inside the media itself [68] [69]. One category of this type of media data is audio/video files. If one considers video data, the video frames need to be displayed in a proper sequence where each frame has as fixed frequency. In a PAL video format, this frequency is 25 frames per second (fps) [70] whereas in NTSC it is 29.9 fps [71]. As such, to display a video file in its correct presentation would necessitate not only the orderly and error free reception of the video file but also the correct selection of the decoding algorithm based on which the media object was encoded in its correct time frame. If the system fails to process the video media file properly in terms of any one of these constraints, it would greatly degrade the quality of the video presentation, in some cases it might even fail to present the video data at all even after successfully receiving the video data at the presentation end of the transmission [72]. Hence, it is necessary to preserve the integrity of data and timing constraints when dealing with multimedia delivery of data. This becomes particularly important when continuous multimedia is being delivered. It should be noted that there are usually multiple media streams in a single multimedia content where each stream is composed based on its own schedule of presentation. With multiple data streams embedded in a single continuous media streaming system, special care has to be given in terms of presenting each stream together with rest of the streams in synchronized fashion while maintaining the relative timing integrity constraints that exist between all the streams that makeup the multimedia data.

2.6 Media Delivery

We can classify continuous media data delivery into two general categories. These categories can be termed as – 'hard' and 'soft' – real-time delivery [73] [74]. Whenever there are strict timing constraints in terms of delivery delay from origin to presentation and high role interchange of client and source, such as is the case in internet telephone applications or video over internet applications (Figure 2.2), we term it hard real-time delivery [73] [74]. If the delivery delay in a hard real-time delivery is very large, then this would render such an interactive continuous media delivery application fruitless and unusable.

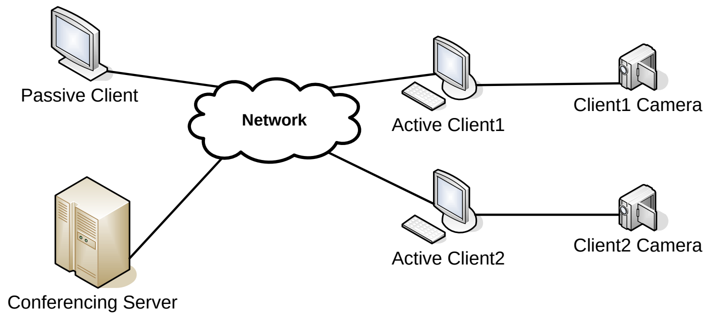

Figure 2.2: Video conferencing in Real-time continuous media.

Considering Internet phone [75], if the total delay caused during voice data being captured and transmitted from the speaker to the listener is larger than 150 ms[76] then that would cause collisions of voice from both speakers, where both users of the internet telephone would be speakers and listeners at the same instance of time. These sorts of delays are frequently exhibited during telephone conversation that have multiple service providers involved or are long haul. Such delays damage the quality of the service.

When multimedia data is delivered with a requirement of only the data integrity and timing of the presentation are to be preserved but delays can be tolerated, it is termed soft real-time delivery [73][74]. VoD is an example of such delivery. Naturally, it is intended that the delays are reduced as much as possible but they do not compromise the presentation itself. The users can playback at their convenience as and when enough data is downloaded (Figure 2.3). Such deliveries are more tolerant to startup delays even if they are much longer than that for a hard real-time delivery application. Soft real-time delivery would function fine as long as a seamless playback is ensured once the presentation starts.

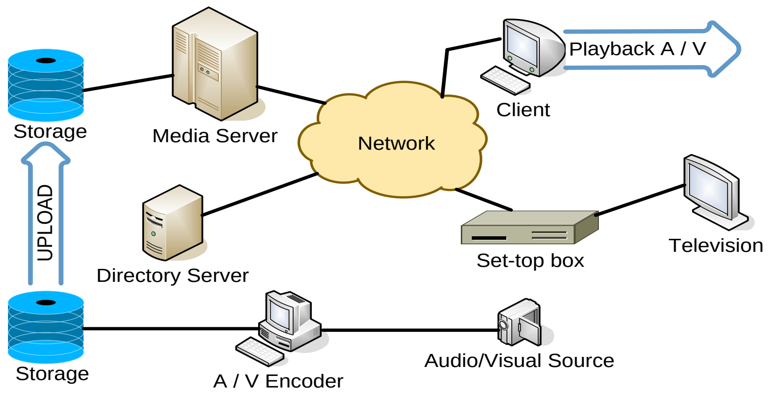

Figure 2.3: VoD an example of soft-real-time continuous media delivery.

2.7 Streaming Versus Download

In terms of multimedia data delivery download is the model that is most frequently used to transfer multimedia from a server to a client. As Figure 2.4 shows, the client requests a multimedia file from a server. Upon receiving the request the server prepares the multimedia object for transmission by first retrieving it from its storage and then transmits it to the requesting client. In WWW, the client would initiate a HTTP GET request and send it to the server by use of TCP protocol. The web server would then locate and retrieve the multimedia data requested from its storage and transmit it to the client using the same TCP protocol connection with HTTP reply message. Once the complete data file is downloaded, i.e. completely retrieved at the client end, the client browser would decode and present the multimedia data to its user [77].

During download the file is copied with data downloaded and placed at the local memory of the downloading client. The downloading client is then able to decode and present this file as it can with its local multimedia objects. This method works for many VoD system but is not efficient when continuous media is being delivered. We discuss the reasons in following sections.

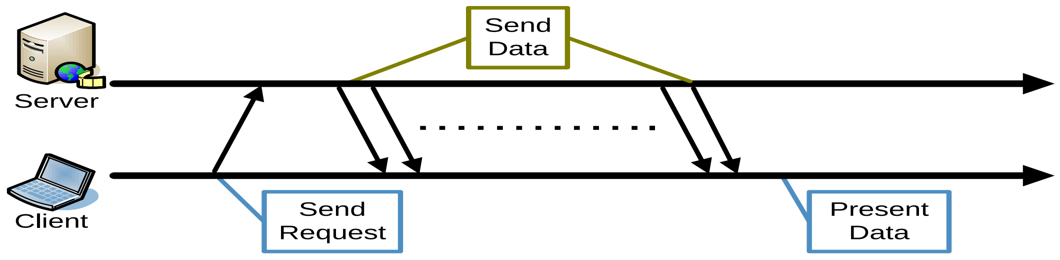

Figure 2.4: Client server interaction in download model.

Illustrated in figure 2.5, take the example of the download model. Here, we are not considering the time it takes to process the request, but we are just taking into consideration the delay that incurs from the moment a client initiates a request to the moment when that same requested data becomes available for presentation at the client. It is observed that this sort of delay purely depends on the media size, i.e. the size of the file as well as the transmission rate at the disposal of the network. Usually, for web applications, files requested are from either small images or simple text pages (HTML) and as such this delay is irrelevantly insignificant.

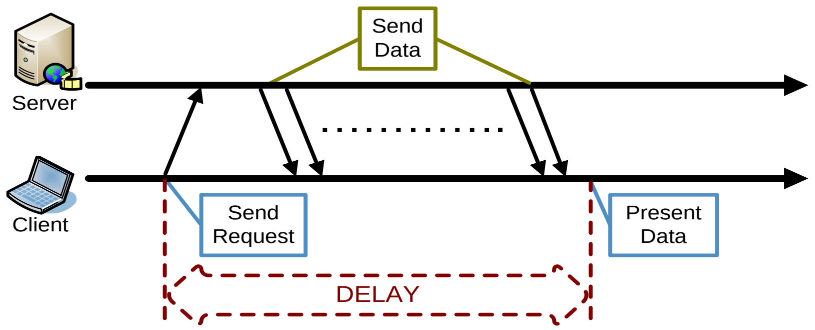

Figure 2.5: Start-up delay in download model.

On the other hand, continuous media is made from a considerably larger chunk of data, as such; the delay caused in transmission of such an object would be much larger. So much so, that it would be intolerable for the provision of service. Let's consider a video file that is two hours long and encoded with MPEG2 standard and average bit rate of 6 Mbps. Such a file produces 5.4 GB (2x3600x6000000/8) of data. Even if one was to transmit this data using high speed broadband internet at 8 Mbps, it will take some one and a half hour (5400000000 × 8) / (8000000 x 3600) for the download to compete and for the multimedia presentation to start. Unacceptably long delay time, waiting for a download to finish, is the main issue that prevents us to use the download model in providing continuous media service.

Although a system cannot present any image or graphic file until and unless it is completely available to that system, continuous media can be decoded and even commence presentation with only partial data at the systems disposal[68][69]. This is mainly because of features such as: a video file is made from a large sequence of video frames and once sufficient number of these video frames, in a sequence, is available; the video file can start its playback and rest of the missing frames from the coming sequences can be downloaded from the server while the playback continues.

Using this feature of continuous multimedia data, a streaming model is one where playback commences while data is still being downloaded [77]. An example of this model is shown in figure 2.6. In streaming model, once the request has been made by the client to the server, the client would only wait for the arrival of first few packets of data in its buffer and once they are available the client system would start with the presentation of the continuous media to the user. While playback of the files initial packets at the users display device, the client would keep on receiving the subsequent data packets from the server machine. Such an approach reduces the time client would have had to wait if it were to download the complete file before playback.

In contrast to the earlier method of download, a streaming model necessitates a couple of requirements. One of these requirements is that it must be possible for the system to fragment the multimedia file into small units that can be decoded and presented individually. The other requirement, continuity requirement, is that the system must make sure a sequential and ordered delivery of the fragments to the client. That is to say, each fragment must reach the client while maintaining the timing integrity of the complete media file [78].

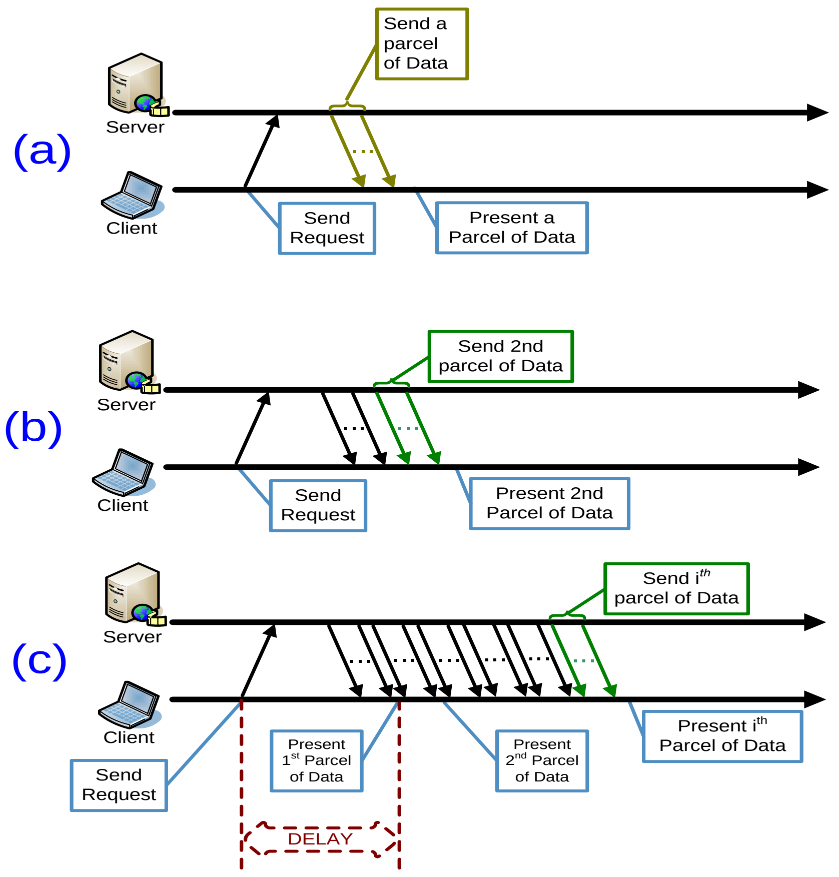

Figure 2.6: Partial media playback in streaming model.

Figure 2.7 shows that the streaming model can easily service many clients at the same time because it receives the media in ordered packets within the timing integrity of the media object [77]. It is worth mentioning that the transmission rate and playback of multimedia objects are fairly similar and as such the startup delay is irrelevant to the total population of clients being served. The only limitation, in terms of size of clients that can be served with same startup delay for the same media file, is that of network and server capacity.

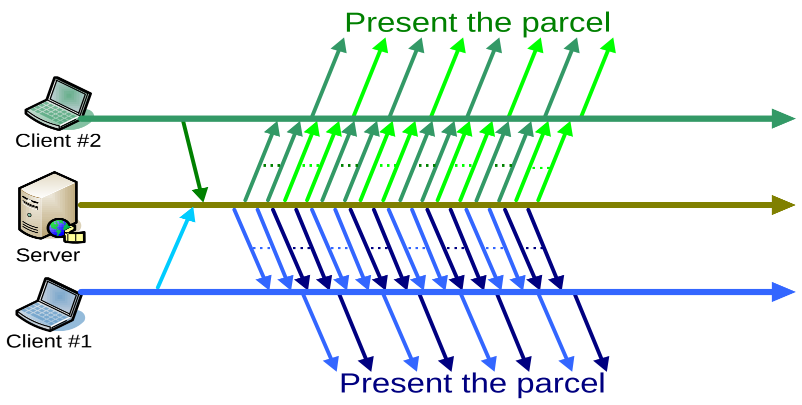

Figure 2.7: Multi-stream pipelining in the streaming model.

2.8 Challenges in Building Continuous Media Streaming Systems

This section discusses the main challenges in the design and implementation of continuous media streaming systems.

2.8.1 Scalable On-Demand Streaming of Non-Linear Media

In the earlier section, it was stated that for the streaming model, we must make sure that the fragments making up the media content are received in correct order and within the integrity constraints of the multimedia presentation being transmitted.

Furthermore, it must be decided when the presentation should commence its playback. Figure 2.8 is used to emphasize the fact that after the media data starts its presentation at the client application layer, the presentation must be played at the same rate throughout the playback. If, for any reason (such as interactive control from the user), the client changes the playback timing or schedule for one fragment, then all subsequent fragment schedules would need to be updated and changed.

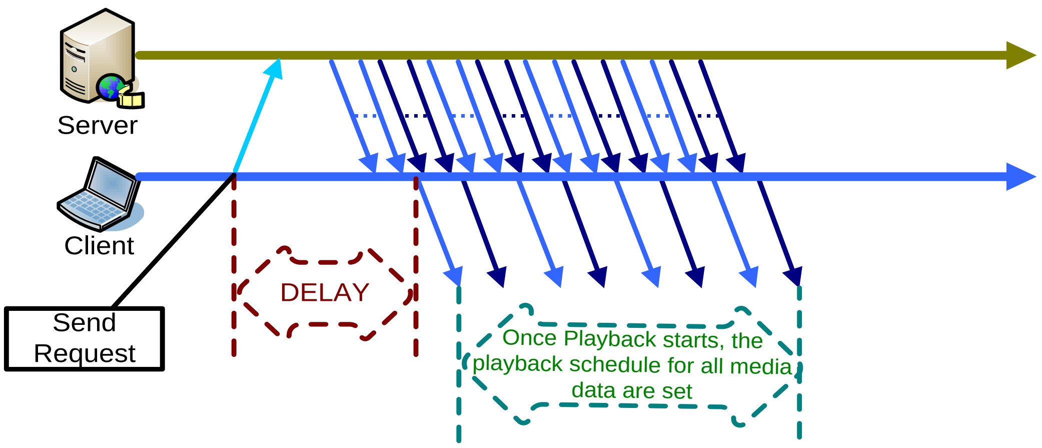
Server
Client
Send Request
DELAY
Once Playback starts, the playback schedule for all media data are set

Figure 2.8: Relation between start-up delay and the playback schedule.

2.8.2 Deviations During Streaming

Like any other large system, continuous media streaming systems are also made up of other subsystems with their own components. These subcomponents are shown in Figure 2.9. This figure shows that multimedia objects are stored in storage devices at the servers. Upon request these objects are fetched from these local memory units and transmitted over a network. On the client end, these objects are received using an interface to the network, saved temporarily in buffers dedicated for system usage and then decoded and displayed at the user screen

There are many different variations that occur during streaming of a multimedia data file. Major variation points of a streaming system are as follows:

Multimedia Data: difference in data rate consumption at the client or transmission of data from the server can cause variations in playback rate.

Multimedia Encoding: Different encoding techniques or compression methods can also be a major reason of data-rate changes of the media file being presented.

Storage Hardware: The quality or type of the storage devices being used for the media files at servers or the buffers at the client are yet another major reason of varying data rate.

Network Capabilities: The resources available to the network and the load on the network can also have a big effect on the transmission and as a result on presentation of the streaming media.

Processing Devices: The processing power and the processing requirements at a given time may also cause huge effect on the streaming process itself.

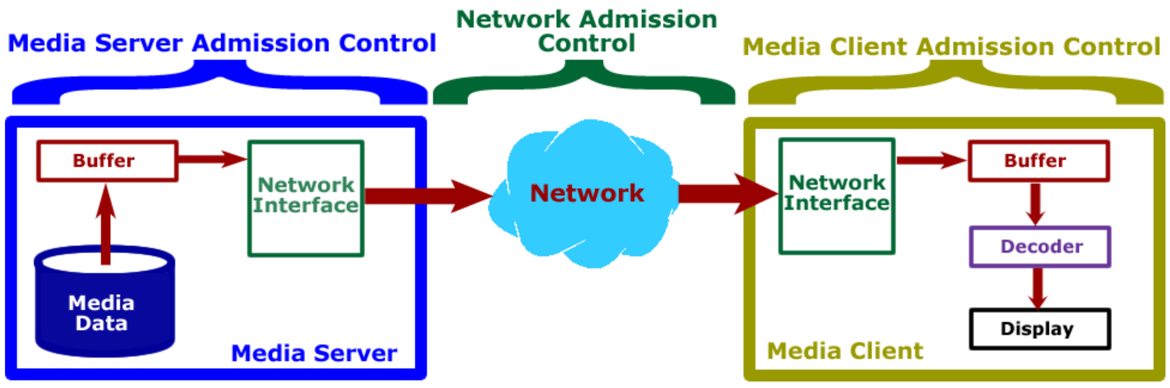

Figure 2.9 End-to-end system and its admission control components

The components making up the streaming system rarely work without a glitch. Normally there are problems, such as delays, that originate from subcomponents of the system and there is never a guarantee as to when and which component would cause a variation in the system behavior. This brings unpredictability in the system.

As an example when one looks at the way multimedia data is created it is noted that, in order to minimize data rate of the multimedia object, compression is required [79][80][81]. Compression of the data itself instigates many variations such as the decoding method and timing required to decompress the individual fragments of the video data etc.

Hard disks are, typically, used to store the media data. The access time as well as disk throughput changes in a system from time to time. When a request is being processed, a server must fetch the requested media objet from its storage and then prepare it to transmit to the client. This fetching time would vary, depending on how busy the storage mechanism is at that instant as well as the scheduler priorities of the system. These parameters cannot be known a priori for a known storage system. Furthermore, there are many types of the storage systems making it, in itself, impossible to predict these values beforehand.

Once the media data is retrieved from the hard disk (storage subsystem) of the server, it starts processing the media object for transmission to the client by making packets and adding control headers. After the packets are ready, the server starts sending them over the network. The available network resources, then decides the time it would be necessary for the media file to go through the network and reach the client. The state of network resources and the path taken by the packets to reach the destination change at different times and cannot be known in advance [82][83][84]. If quality-of-service

(QoS) can be assured for a network, only then the system can predict and as such efficiently manage the delays incurred due to many subcomponents involved in the streaming [83,84,85]. Today's best effort internet is unable to do provide any such guarantees [72,86].

As for the client side, once the media data reaches the requesting client, it saves it in its buffer memory and starts the decoding process. As aforementioned, the decoding time depends on the encoding algorithm complexity as well as the processing power available to the client along with the processing schedule at a given instance of time. The same is true for playback time.

Figure 2.10 shows how all these variations in the streaming process of a multimedia file can affect the service provision to the client. It is clear from the discussion in this section that variable timing is a part of the nature of continuous media streaming systems and it should be treated as a special case scenario.All systems managing continuous multimedia data must take precautions for these variations and changes.

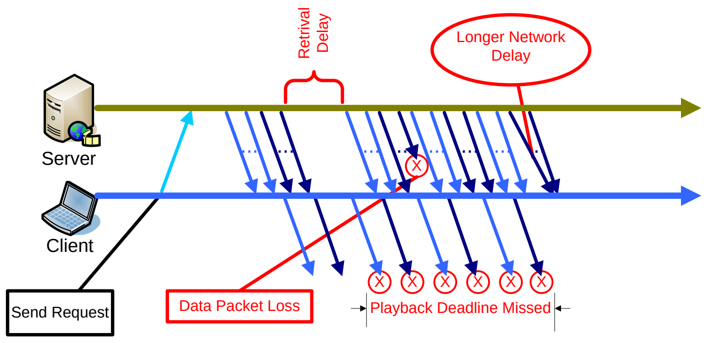

Figure 2.10: Variations in the system can disrupt continuous media playback.

2.8.3 Real-time Applications

Many applications need interactive user participation in real-time – these are called real-time applications, such as interactive teaching, video conferencing or real-time video games. In such cases, live streams are exchanged between the participants of such activities. These live streams are intolerant to delays like startup delay for every live stream [73, 74, 75, 78]. Hence, designing a continuous media steaming system that serves real-time multimedia data has to put up with more stringent requirements in terms of timing, in addition to all the other constraints already present in any multimedia streaming systems. As such, a trade off must be made among these constraints depending on the system goals, in order to balance the system and design efficient yet realizable real-time continuous multimedia streaming system.

2.8.4 System Scalability

An important feature of any system is that whether it is able to increase its capacity of service with the increase in the demand for its services without increasing cost of the system a lot, that is if the system can be scaled up [87]. Figure 2.11a illustrates a system with a single multimedia server. With increasing clients in the system, one server would become insufficient to continue providing service efficiently. This would cause degradation in quality, increased waiting time for the clients, and delays in accessing the storage, to mention a few system overload related issues. A scale up would mean adding a new multimedia server to the existing single server system and to copy the media on existing server to the new server, shown in Figure 2.11b. Such a scale up would increase the capacity of the system two folds at the cost of a single multimedia server.

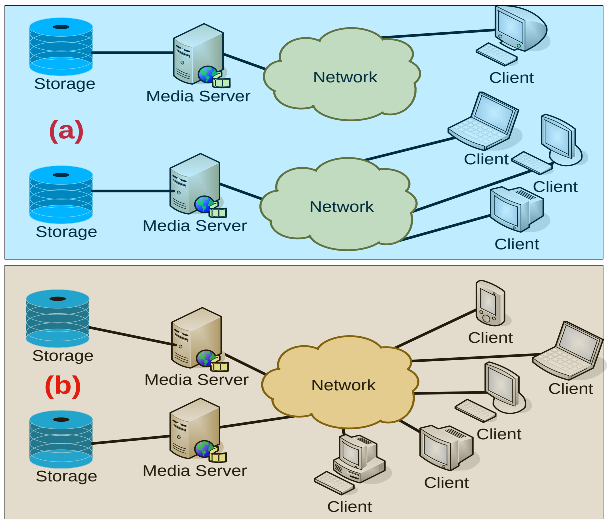

Figure 2.11: Increasing the service capacity of a media streaming system.

2.8.5 System Reliability

System reliability is yet another very important feature and poses a challenge to continuous multimedia services, especially streaming services design and management. If there are failures at any subcomponent of a continuous media service system at any given time, it would naturally disturb the entire service mechanism. The only way to ensure this does not occur is by adding fault tolerant features to the system while designing the system. The failure, may it be disk failure, network failure, peripheral failure or any other failure should not be allowed to hinder the service. To the very least it must not be allowed to render the service inoperable.

To make sure that the system keeps on providing the service it is intended to provide, redundancies might be necessary to be included in the system. Redundancies such as, disk, bandwidth, processors, or servers help in making sure that there are no interruptions in service provision once the system is up and running. This thesis presents a multimedia service provision system for continuous multimedia in a distributed environment using redundant servers arranged in a hierarchical fashion.

2.8.6 System Trade-offs

In this section, different kinds of tradeoffs are examined. Tradeoffs become necessary when different challenges have conflicting requirements or when resources are not enough to fulfill all the challenges. The tradeoffs can be categorized as follows:

- Tradeoff in capacity – This would include but is not limited to the tradeoff in the throughput of the disk and that of I/O as well as the bandwidth at the networks disposal.
- Tradeoff in time – These are tradeoffs that deal with timing constraints such as delay, jitter, and the time it takes to give response to a request.
- Space tradeoff – Space tradeoff targets the space needed to store data, as in storage space (hard disk etc.) as well as the buffer space available to the servers and clients included in the service.
- Tradeoff in Quality – This would involve a tradeoff in the quality parameters and the state of how high or low is the quality of the data being presented to the consuming client.

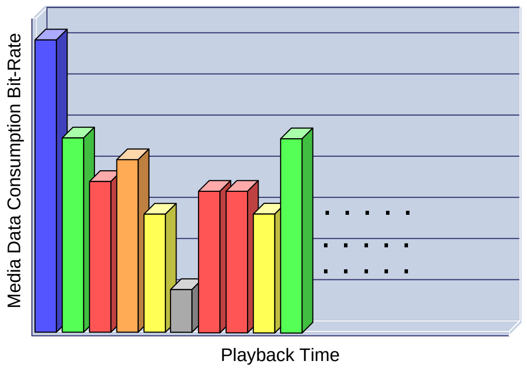

Figure 2.12: A media stream with time-varying playback bit-rates.

In the coming subsections, for the sake of discussion, consider the diagram in Figure 2.12 where a multimedia object is being streamed with variant rates of playback time. Let us further consider that the file being streamed is segmented in time slices of *T* time units, hence making each playback time slice for each segment identical in length. This would make the size of each segment different as the bit rate is different in each identical time slice. In the coming sections, that discusses the tradeoffs using the initial example in Figure 2.12, let us assume that r_i is the playback rate for a segment *i* where *i* is an integer from 0,1… to *n*. We look at a scenario where this multimedia file will be streamed from a continuous media server to a client.

2.8.6.1 Capacity Tradeoff

Let us assume, to keep things simple, that the only traffic in network is the multimedia object being streamed. Furthermore, let us also assume that the network has a fixed bandwidth at its disposal. In this case, in order to serve the system efficiently, one tradeoff would be in bandwidth capacity. In such a tradeoff, the network can allocate a bandwidth capacity $C = \max\{r_i \mid \forall i\}$, where *C* is the highest bit rate required by any segment that makeup the multimedia object. This scenario is depicted in Figure 2.13

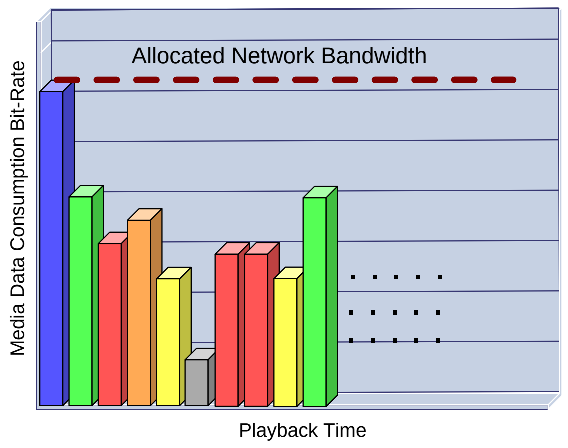

Figure 2.13: Network bandwidth allocation based on peak bit-rate.

It can be noted, with a first glance at the Figure2.13 that this approach suffers from a problem where only one segment (the one with highest bitrate) uses the bandwidth efficiently, while all other segments waste a part of the bandwidth allocated to them. This means although the streaming would go smoothly, this will be at the cost of wasted bandwidth, hence the tradeoff in useable bandwidth capacity

2.8.6.2 Time Trade-off

A different tradeoff can be that in time, where we waste time in order to save network capacity at the network disposal. Continuing the example from Figure 2.12, let us say that the segment with the highest bit rate i.e. the first segment is streamed at a lower bitrate as compared to the required playback rate. This would mean that, the initial segment will require more time than *T* time units. This would necessitate a delay in playback. Figure 2.14 illustrates this scenario. Making a tradeoff in time would save bandwidth, but would require a higher startup delay.

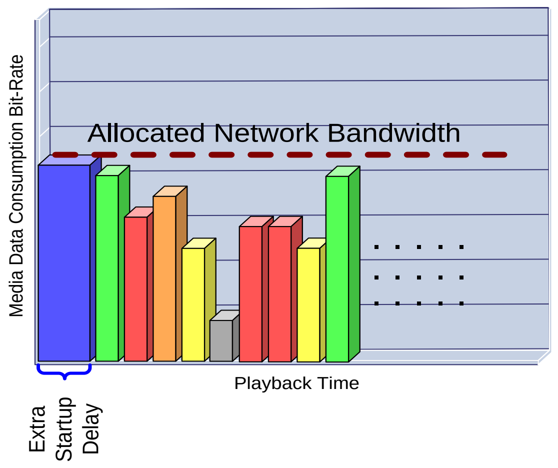

Figure 2.14: Time trade-off incurring delay by reducing playback in initial segment.

2.8.6.3 Trade-off in Space

Continuing with the earlier example and the discussions we note that the crucial network resource, i.e. the whole of the bandwidth remains not effectively utilized and is a part of it remains wasted, especially in cases where high bit-rate and very low bit-rate video segments are being transferred simultaneously. This unused (or wasted bandwidth) can be reclaimed provided the system looks into the future packets and plans in advance for their transmission as depicted in Figure 2.15. The figure is used to bring attention to 5th packet, which is low bit-rate, and the 6th packet that is made up of media that is high bit-rate as compared to rest of the segments.

If the system providing the streaming service, can foresee this difference in bit-rates required, it can initiate the 6th packet as soon as the 5th one is sent. In this way, knowing the extra bandwidth resource is available and the slow bit-rate of 5th segment, the system is certain that the sequence would not be disturbed and 5th packet will reach before the 6th packet segment. This also ensures that, because 6th segment was initiated earlier than normal transmission time, average transmission rate is improved. As in our example, reducing the transmission rate of the packet with highest bit-rate in the media data, inevitably improves the peak bit-rate of the entire transmission.

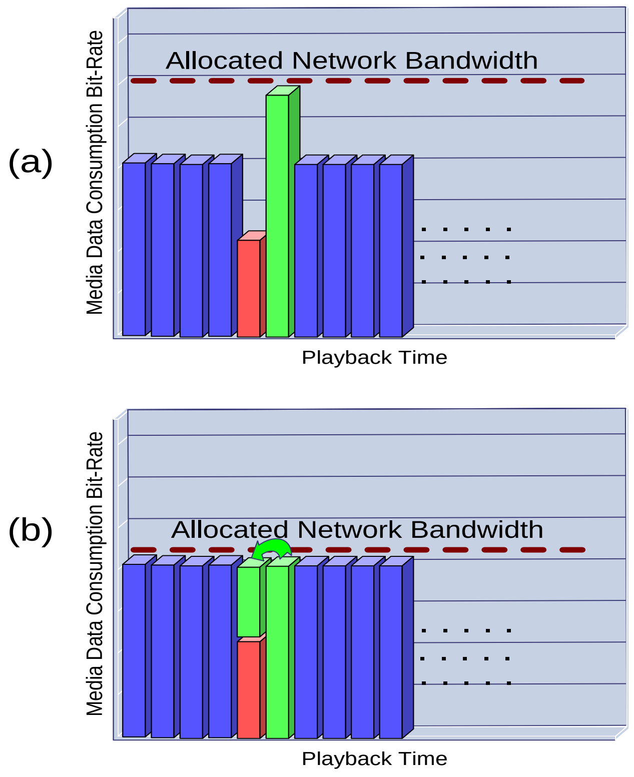

Figure 2.15: Spatial trade off in the buffer of the client.

2.8.6.4 Quality Trade-off

Throughout our discussion thus far, it was a given that all media data must be transferred to the client. A tradeoff in quality of media suggests that some of the media data may not be transmitted to the client at the cost of quality but saving the bandwidth being used and providing a more speedy service as shown in Figure 2.16. The data that is not sent should not, however, render the data being sent useless but, naturally, the media being transferred cannot be used to reconstruct the original media file but a lower quality version of the same.

How much of the quality would be compromised is purely dependent on encoding techniques, the skipping technique, the type of data and its amount that is skipped. Another method used to degrade the quality of a media file is to map a high bit-rate file into a low bit-rate as illustrated in Figure 2.16 where the 6th segment is transformed from a high bit-rate to a low bit-rate using some encoding technique [88], making sure that the new bit-rate falls in the range of available bandwidth resource.

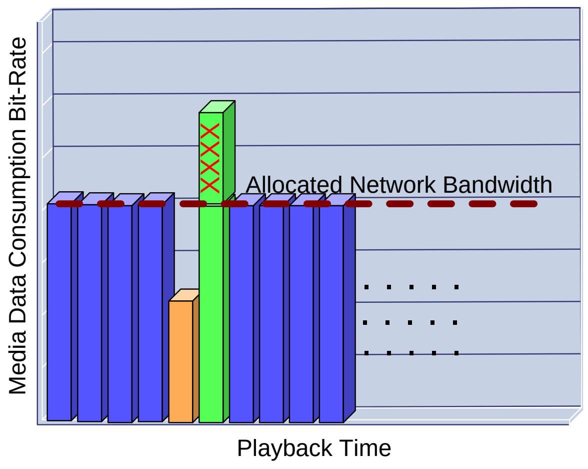

Figure 2.16: Trading off the quality by skipping some media data.

2.9 Performance and its Guarantees

Whatever technique of transmission is deployed and whatever the tradeoff, the ultimate goal in media data streaming system is to make sure that the media files and their segments reach the client in time and in order, so that the client can use that to decode and display the media data content to the client. This means providing performance guarantees. Provision of performance guarantees means that the system must make sure that the worst case scenario is taken into account for any tradeoff or technique being deployed for the streaming system.

When a system can guarantee that performance matrices will be met in all cases, it is called deterministic guarantee [89].Consider cases where the worst case scenario never or rarely happens, it is apparent that resources will be wasted.

Another approach can be probabilistic performance guarantee or statistical guarantees [89][90], where only the most important performance matrices are met and guaranteed for most instances, and at rare times they might be ignored. Yet another way is to provide all the resources at the disposal of the system for the streaming of the media providing no guarantees for any performance matrix. This is called best effort performance guarantee [89][90] and in this way no guarantees are actually provided and that the system does not or cannot control the media data requirements as well as the transmission requirements of the system used for the transmission, such as the internet. It must be noted that best effort does not imply poor service, but that at times service might be poor provided the required resources are not available.

2.10 Admission Control

In cases where the network utilization is very high and as such the network resources are not sufficient to provide the required service, the system would not be able to entertain the requests being received from the clients. This is natural as there is a physical limit to the amount of resources that can be used and at any given time the number of requests being served should not exceed the amount of resources at system disposal (Figure 2.17). Admission control is the part of the system that makes sure that requests being admitted can be fulfilled [91][92].

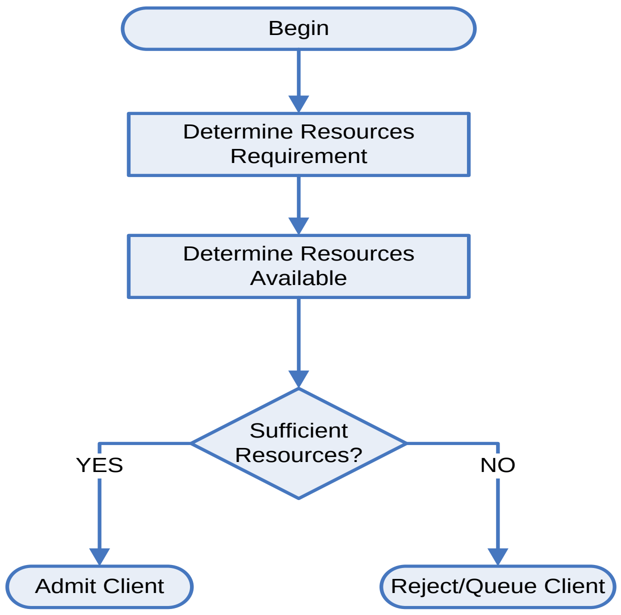

Figure 2.17: Flow chart for a general admission control procedure.

Looking at the multimedia streaming system extensively, like in Figure 2.18, we note that just having an adequate admission control system in place would not be enough to provide performance guarantees. There are many components to a streaming system, as aforementioned, whereas the resources are limited. The components involved in the streaming process are not used to stream single stream but multiple streams, furthermore the client side is also typically busy with more than one application, where each application is contending for client resources such as processor time, input and output resources etc. In order to provide an end-to-end performance guarantee, all these components, resources, load on the systems involved must be considered.

Within a distributed continuous multimedia streaming system, each server, all bandwidths and connections, every storage and transmission unit must be investigated so that a fair admission control system is placed, and a proper resource allocation policy is established so that a seamless streaming system can be designed that can guarantee the system performance.

For each request being considered for admission, the system must make sure that each component involved in the streaming process is capable of providing the service throughout the life of the request. If for any reason a single component is unable to provide or continue its service for a period of time, then that request cannot be met and as such admission should not be granted.

Chapter 3

DISTRIBUTED CONTINUOUS MULTIMEDIA STREAMING ARCHITECTURE

A distributed continuous multimedia streaming system consists of multiple continuous multimedia servers, where each server is capable of providing the necessary resources needed for streaming the media objects. The clients for these objects are, geographically dispersed. When a request is received by the system, the continuous media servers should try to realize the request and serve the client. Normally, DCMS's are designed as a network and many times they are arranged in a hierarchical fashion.

In a hierarchical arrangement, each Centralized Continuous Multimedia Server acts as a node of the hierarchy where the links connecting these nodes are the edges. The CCMS's are, as mentioned before, able to store and stream a limited number of media objects. The links making the edges and providing the communication mechanism between the nodes are assumed to be able to provide performance guarantees that are required for the media object being transferred.

The hierarchical topology, usually employed for DCMS, resembles that of a tree where one end of the hierarchy is the root (top most node) and the other end has leaves (entry nodes). The client requests are handled through the entry nodes, also known as head-ends. Clients are connected to the DCMS system through the head-ends, typically using broadband connection. When a client makes a request to a DCMS for a media object, it is received by the head-end. The head-end node then looks to sees if the media object is stored in its local storage. Provided the media object, requested by the client is available at the storage of the head-end node, it (the head-end) serves the client. Otherwise the head-end node forwards the request to its parents at the next higher level of the hierarchy. If needed, the request will eventually reach the root node, which will then serve the request by streaming the object through the hierarchy via the head-end to the client. Hence, object placement and object location and delivery are at the center of the resource management component of a DCMS. These functions are discussed in detail below.

1. **Object Placement**. The main task of this component of a DCMS is to map the multimedia objects throughout the DCMS nodes. The placement should be such that the costs to the system (such as the communication or storage etc.) are minimal. This issue, also referred to as Media Asset Mapping Problem (MAMP), has been contemplated by several researchers. Some addressed it using analytical models and access patterns [93], others considered resource constraints to get optimal distribution of the media data employing some duplication policies [94, 95, 96, 97].

2. **Object Location and Delivery**. Object Location and delivery component of DCMS deals with finding the locations where the multimedia object is placed within the DCMS and then allocating a delivery route together with all resources needed by the system to satisfy the client request. The main purpose of this component is to locate and then deliver the requested object such that minimal system resources are utilized for a single stream, hence increasing the efficiency. At the same time, another key goal of such a component is to keep all resources

occupied with different streams that are serving different clients, as such increasing the utilization.

3.1 Object Location Scheme

The object location component is, usually a middleware that runs at the application layer of a DCMS. The DCMS is made from multiple, hierarchical CCMS's. These CCMS's are capable to handle multimedia storage as well as the streaming requirements. Making the middleware independent of the network layer and placing it on the application layer makes it possible for the system to be independent of underlying communication network. This enables the system to be able to provide performance guarantees, irrespective of the underlying network. Hence, if the network topology is designed efficiently, then together with resource management middleware, they define the general architecture of a distributed multimedia streaming system. As far as topology is considered, we have chosen to use the Redundant Hierarchy [9] or, briefly, RED-Hi (see Figure 3.1). The RED-Hi topology decreases the rigid restriction of one parent per node for pure hierarchy and allows each node to have two or more parents.

Having more than one parent makes it possible for the system to balance the load on any given path of the topology, not only that it makes it possible for the system to be more fault tolerant and efficient in terms of resources and cost. There is a redundancy involved in such a topology; nevertheless this redundancy is not of extra resources but only of extra links. These links can share the same bandwidth resources and because of that, the aggregate bandwidth that connects a node with its multiple parents would be the same as the bandwidth that is used in a pure hierarchy for a node to connect to its single parent.

It is evident from the earlier discussion that in RED-Hi the system does not require higher bandwidth resources but can still provide a possibility for higher connectivity at the negligible cost of an added link. Looking at some existing hierarchical structures such as Central Offices (CO), Points of Presence (POP) and Internet Service Providers (ISP) that are at the foundations of streaming Internet provision system employed by DCMS, we notice that RED-Hi is highly compatible with them as they too are connected with redundancies [98, 99].

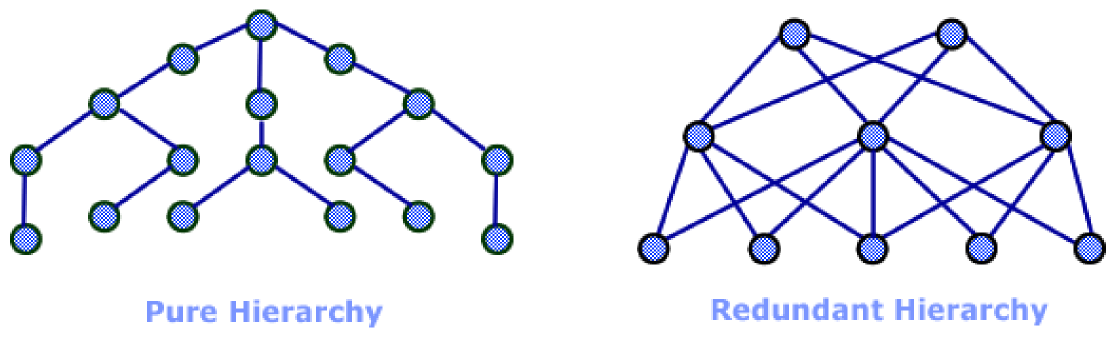

Figure 3.1: Pure and Redundant hierarchy.

The Distributed Continuous Multimedia Streaming system being considered in this study has the following assumptions:

1. The topology used for the system would be based on RED-Hi.
2. Centralized Continuous Multimedia Streaming servers will be used as nodes where each would have its own storage and bandwidth resources.
3. The underlying network would be responsible to provide resources such as bandwidth where control and streaming media streams would be sharing the bandwidth resource.

4. The links would be duplex links where the traffic can go in both directions.
5. Nodes at the top of the hierarchy are called Root nodes, the ones at the bottom are called Entry nodes or Head-ends whereas all intermediate nodes are called Intermediate nodes.

6. In object distribution policy, all multimedia objects will initially be placed in the root nodes.Then, based on demand and popularity, they would be duplicated to the intermediate nodes. With physical limitation approaching in terms of available memory size, the least recently used (LRU) policy [100] would be used to remove the objects from the intermediate and entry nodes.

An application agent is assigned to every CCMS that keeps track of the resources maintained at the node. These agents are responsible for knowing at any instance the number of multimedia objects present in the storage of its CCMS, the total and available bandwidth at the CCMS as well as the number of links free and number of links that are busy at a given time. Furthermore, these agents should keep track of load on each link that connects the node to its parent nodes and child nodes.

These agents assist the CCMS in locating the requested media object in the entire system, i.e. all the nodes that belong to the DCMS. Furthermore, these agents help in selecting the node most suitable to be the source for the request satisfaction and to designate the most appropriate path that can be used from the source CCMS to the client requesting the media object. This decision is made while considering the load along the path of the request transmission ensuring the highest utilization of the available streaming resources and providing an optimal balance in the load at the nodes involved in streaming. Finally, the agents allocate all the necessary resources needed for the streaming of the object from its CCMS. Collectively the agents would make allocation throughout the DCMS network for all necessary resources required for the seamless flow of the streamed packets from the source node to the requesting client.

3.2 Object Location Algorithm

One of the main contributions of this study is development of the object location algorithm that handles the task of locating the media content in the distributed server system. The proposed algorithm performs this task using two modules. One module in each node of the system is dedicated to receive query messages from its child node, in case of a head-end, this query is received from the client through the network. The second process is devoted to the processing of the query in the node itself and passing the query to its parent node incase the media required is not available locally.

3.2.1 Receive Query Process

This process is responsible for receiving all query messages in a node. The process can be considered as a thread that is active as long as the server is up and running. It receives the request and performs the preliminary processes on it.

```
While (lifetime>0)
{
  RecieveQuery(new_query);
   if (requested media available locally)
       SendResponse(ok_message,Child);
   Else
   {    Send_Query_and_wait_Process();     }
}
```

3.2.2 Send Query and Wait for Response Process

This process is responsible to send the query messages received by a node to its parents, wait for a response from both of the parents and finally to inform its child process of its findings from the inquiry it made to its parents. This process takes into consideration the timeout and sends an Error message in case it is required.

```
Send_Query_and_wait_process();
{  SendQuery(new_query,Parent1);
  SendQuery(new_query,Parent2);  //set timeout for both parents
  while (Timeout>0)  {
   if (RecieveResponse(Status1) !received) and (ReceiveResponse(Status2) !received)
         Wait for the response from both parents
    else
        if ((Respons(Status1)=OK) or (Response(Status2)=OK)
          {  if ((Response(Status1=OK) and (Response(Status2)=OK)
              if (timestampe of Status2> timestamp of Status1)
                   select Parent1
              else  select Parent2
           else  select the only parent with OK status for the file.
          SendResponse(ok_message,Child);   }
      else   SendResponse(negative_message,Child);
     }
    if (Timeout=0)
         SendResponse(Error_message,Child); }
```

3.3 Object Location

Flooding is a famous technique [101] used in routing algorithms. The reasons it is used frequently is because it is highly robust, and it explores all paths at the same time. Furthermore, it is able to have parallel communication among every node involved in the routing system, a highly desired roperty in distributed applications. Finally, it gives a minimal delay in terms of delivery because we are certain after exploring all paths that the shortest path is actually the shortest possible path among a set of nodes. There is an extra cost [102] of time of flooding that comes into consideration, but for multimedia data streaming, this delay or extra overhead is negligibly small compared to the overall service time involved in the streaming process. As such, our object location algorithm uses flooding techniques.

Flooding makes use of different propagation policies depending on the propagation area that should be covered in the routing or object location problem at hand. The possible policies are:

1. Parent-only policy: In this policy, nodes forward the requests that they cannot themselves serve as a source server, only to their parents. As such, the area covered by the propagation of the flooding while locating the object would limit only to the ancestors of the head-end that receives the request from a client.
2. Another policy would be Inclusive policy where a nonselective flooding is performed. Non-selective flooding would mean that any node that cannot service a request itself would propagate the request to all the nodes they are connected to, except the one node that had sent the request to them in the first place. In this policy, all the duplicates of the requested media would be located in the entire network.

3. A third policy that can be used is the Sibling policy. This policy has an addition to the parent-only policy where a node that receives a request for an object from one of its child nodes would forwards it to the siblings of the child from which the query is received and 2) its own parents. Whereas when a node gets a request for an object from its parent, it never forwards it.

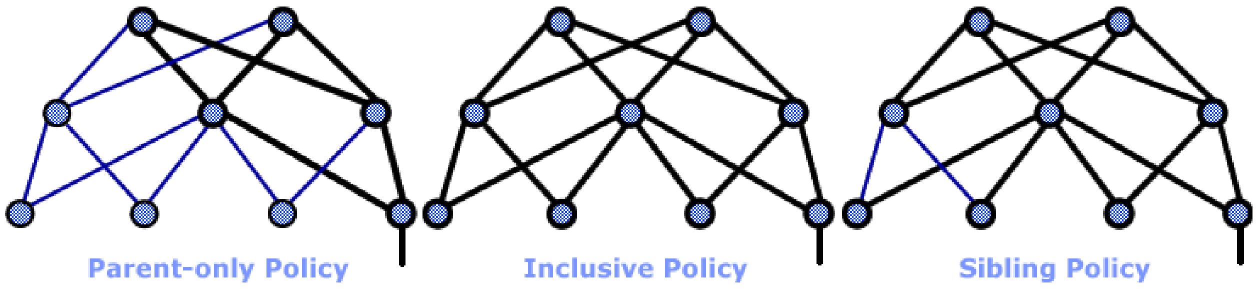

Figure 3.2: Coverage of the propagation policies.

3.3.1 Object Location Algorithm

We propose using four message types in the process of locating the media file in a distributed continuous media server system:

1. **Query message** from a node, asking if the node receiving this message contains the media required.
2. **Negative message** by a node, stating that it does not hold the inquired media data and the status of the query message.
3. **Ok** message by a node confirming that it contains the media Queried.
4. **Error** message by a node informing a timeout situation at a node.

3.3.1.1 Query Message

We propose that this message be propagated in the system using the following approach. The head-end node receives a request from the network for a specific media file. It checks its index of contents in order to find out if it is aware of the availability of the required media file or not. If it contains the media locally then no query message is required and the media file is conveyed to the network. If it finds out that the content is available at a node in its parent's chain, then it tries to confirm this with the specific node, and upon receiving this confirmation, it connects the network with this node. In case the head-end finds out after checking its index of contents that it does not have any knowledge of the file required, then it prepares a message stating "Query: Media File Name" and adds a timestamp and the node identification number to the query message. The query message is then sent to the parents of the head-end node and the head-end node sets a timeout for a reply by the parents. There will be two timeouts for two parents.

3.3.1.2 Negative Message

The Negative message is sent from a node to the child that had sent the Query message to this node. This node will check its index of contents after receiving a Query message and in case it finds that it does not contain the media data requested it sends the same query message to its parents. This process is repeated by all the nodes until either the root nodes are reached or the media is found in some node after checking its index. If the root node is reached and the media asked is not found in any node, then the root node initiates the Negative message saying "Media File not found". The child process upon receiving the negative message from the root node waits for either the Ok message or a Negative message from the other root node. If any of the roots pass an Ok message then the negative message by the root that did not find the required file is ignored and an Ok message is passed down. In case both the roots send a negative message, the child node prepares a negative message for its child and in this manner the tree is traversed. The head-end, in case of receiving negative messages from all nodes in the hierarchy, conveys the message to the network that from this head-end there is no path that leads to the required media file.

3.3.1.3 Ok Message

This message corresponds to finding of the media file by a node in our proposed distributed continuous media server system. When any node receives a Query message from a child node, it looks the media file up in its index of contents. In case it finds the required file in its index, it sends an Ok message to the corresponding child process. This message contains the statement ‘File available’ along with the node identification number of the node containing the media file and the current time stamp. After doing this, it places the identification number of child node that received the Ok message in its index of content, meaning that this child is aware of the media contents’ availability at this node. This is done because when this media content has to be moved or deleted, for one reason or the other, the child process should be informed of this change.

Upon receiving the Ok message, the child process checks the timestamp of the received message and compares it with other timestamps of the received Ok messages (if any). It discards all messages except the one with the smallest timestamp on it. It conveys this information in an Ok message of its own stating ‘File available’ along with the node identification number of the node containing the media file and the current timestamp. After doing this it adds the name and the node number of the node containing the media data in its index of content. Furthermore, it places the identification number of child node that was sent the Ok message in this index of content.

In this way, information is passed down the hierarchy from parent to child nodes until the head-end is reached. The head-end then conveys the message of availability of the content to the network and connects the node with the smallest timestamp with the network.

3.3.1.4 Timeout and Error Message

When a client sends a request and it is received by an entry node the network for a media file that it does not have listed in the index of contents then the head-end sends a Query message to the parents of the head-end node. After sending this Query message, the head-end has to wait for a reply from the parents informing it of the status of the information required. This wait cannot be infinite as the head-end as well as the client cannot wait forever in case of a lost connection between a node and its parent in the hierarchy. The waiting on part of the node sending the Query message has to be sufficiently long for the parent node to receive the message sent by the node, check its index and send a reply message to the node waiting for the reply. Let us call this timeout 'n' second. As each node has two parents, it will have two separate timeouts n_1 and n_2 corresponding to the parents. The main tasks for the nodes can be divided into three categories:

1. Sending/receiving message between nodes involved
2. Checking of the index of contents by the parent node
3. Comparing and comprehending the messages.

Any timeout mechanism should take these three aspects into consideration. After determining the timeout the nodes are to wait for the response messages from both its parents. If none of the parents send a response message before the expiration of the timeout then the child process transmits an Error message to its child processes and assumes the position of a root node itself.

3.4 Request Propagation and Provision

Whenever a client initiates a request, it is received in the system by a head-end node. The head-end node is a CCMS with agents associated to it and these agents create a query message to be propagated into the DCMS network. The coverage of the network propagation is decided based on the flooding policy selected.

Each query message will go through different paths and analyze the costs incurred over the path traversed incrementally. Eventually, some nodes will be traversed that hold the requested media objects in their local storage. These nodes will then create and transmit positive response messages back to the head-end nodes using the same path that was used by the query message to reach them. The acknowledgement message – positive response message – would entail the resources and their load. The load and cost information is readily available at the agents of each CCMS. These agents use the information included in the positive response message to determine the best path from their node to the source node. In this way, incrementally, an overall best path would be found. This path would then be allocated and reserved by the head-end node for the streaming of the media object from the source to the client requesting the object from the DCMS system. Once the best node is selected and reserved, the agents would be periodically used to maintain a list of alternate routes to other replicas of the streamed object in the system.

Chapter 4

PETRI-NET MODEL DEVELOPED FOR THE SYSTEM

Petri-net [102] is a tool used for modeling a directional graphic depicting flow of events. A Petri-net is made from arcs, places and transitions. Input arcs start at a place and connect it with a transition whereas an output arc connects a transition with a place with arc originating at the transition. A Petri-net model [102] is designed for the system developed in this study, in order to help with the simulation. The model is based on three levels in hierarchy of the CCMS nodes of the DCMS network. A total of 9 servers are considered for the Petri-net model.

Tokens are used in Petri-nets to represent the current state of the system being modeled. Places may contain tokens and these tokens can have a marking and a type. A transition is fired when a token is moved from a place to another through a transition and this firing of transition depicts a change in the state of the system. When a transition is fired, a token is removed from an input place to an output place. There might be preconditions attached to firing of transition and a token can only move from place to place if all the preconditions of a transition firing are met.

In our model, we assume that all servers are geographically dispersed and are connected to each other through the Internet. The clients are connected only to Level 1 servers through the Internet. The requests made by clients are accepted only through Level 1 servers, called head-ends. The general structure of the system is depicted in Figure 4.1. All the connections between the clients and servers are duplex as well as the connections between the servers with each other. Level 3 servers are at the root of the system and have no parent node or higher level of hierarchy.

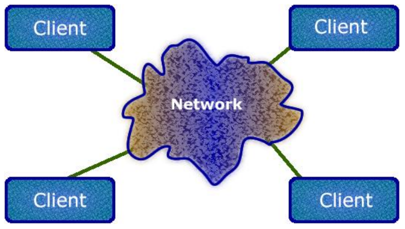

Figure 4.1: Clients connected to the network.

The servers at Level 1 are responsible for all the communication with the clients. In this model, it is assumed that there is no failure at a node (server) and timeout does not happen. As mentioned before, the clients communicate with the servers at level 1 through the network, forwarding their requests and waiting for replies. This is shown in Figure 4.1.

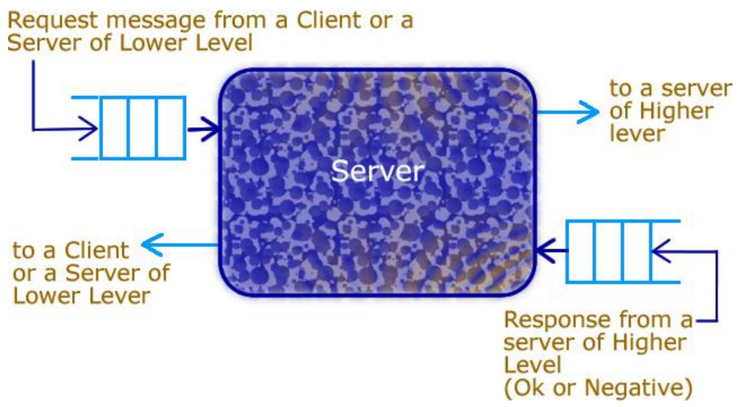
Request message from a Client or a Server of Lower Level
to a server of Higher lever
Server
to a Client or a Server of Lower Lever
Response from a server of Higher Level (Ok or Negative)

Figure 4.2: General input/output mechanism for a server.

The server on the other hand take inputs both form the client (child node) and from the server at higher level of hierarchy, except for the root node. The general input/output mechanism of the servers is shown in Figure 4.2.

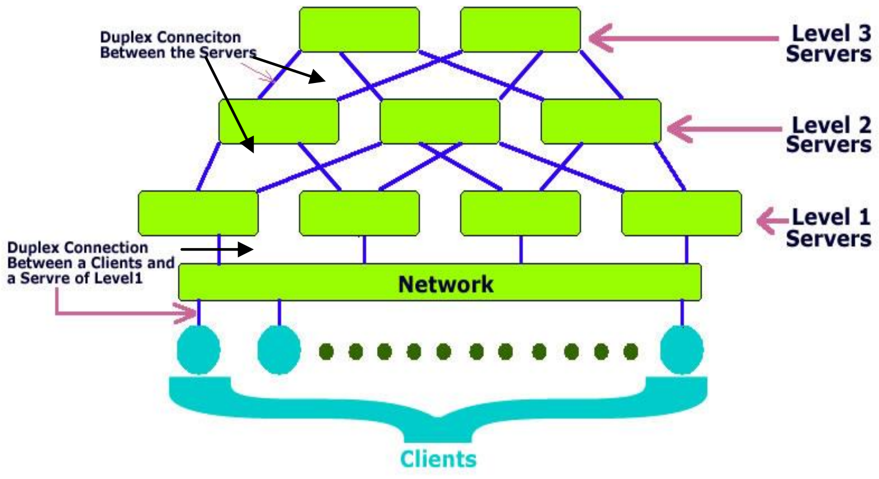

Figure 4.3: General structure of the system.

4.1 Assumptions for the Petri-net Model

The following assumptions are made while developing the Petri-net model for this system:

1. Each client, with equal probability, sends a request to a server of the first level and waits for the response. Only after processing of once request a client may generate a new request.
2. Requests from clients arrive to servers of the first level through the Internet.

3. Servers of the first level communicate with the servers of the second level through the Internet as well. This is valid for all communication between all servers as they are geographically dispersed.
4. Transmission in the Internet is modeled as a random delay, distributed exponentially with some specific mean time ‘T_N’
5. Format of messages is as given in Figure 4.4.

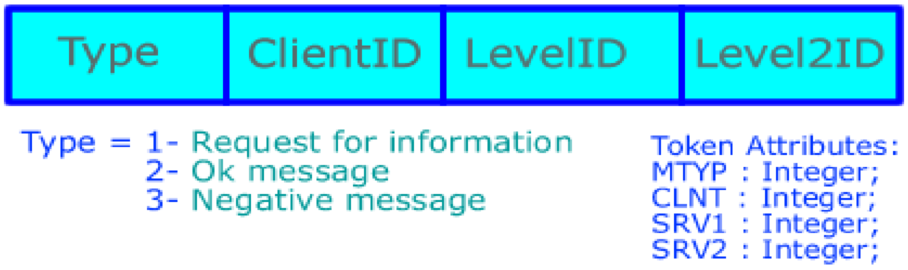

Figure 4.4 Message format for the model.

4.2 Model for Clients

The client sends a request to the head-ends, initiating the search for the media content. The Petri-net graph representing the functioning of a client is shown in Figure 4.5. In this and subsequent Petri-net graphs, ‘n’ is the client number where n=1,2,… N.

The Petri-net graph of Figure 4.5 consists of transitions represented by ‘T’,’X’ and ‘Y’, places represented by ‘S’, token by a dot, and the inputs/outputs of the places by arrows. T10n is a transition that simulates the (random) time of generation of a request, exponentially distributed, with given mean T_R. Transition T50n to T80n simulates random time of the transition of the request in the internet, exponentially distributed with some given mean T_N. Transition X10n simulates the random choice of a server of the first level (head-end) by the client, with equal probability. S10n place must contain the token at the start of simulation (as shown in Figure 4.5), and attribute CLNT must be assigned ID of the client initiating the specific request. S30n contains the average response time for a client. The transition X10n must place 1 in the attribute MTYP (indicating that this is a request message) and id of the client in the attribute SORC.

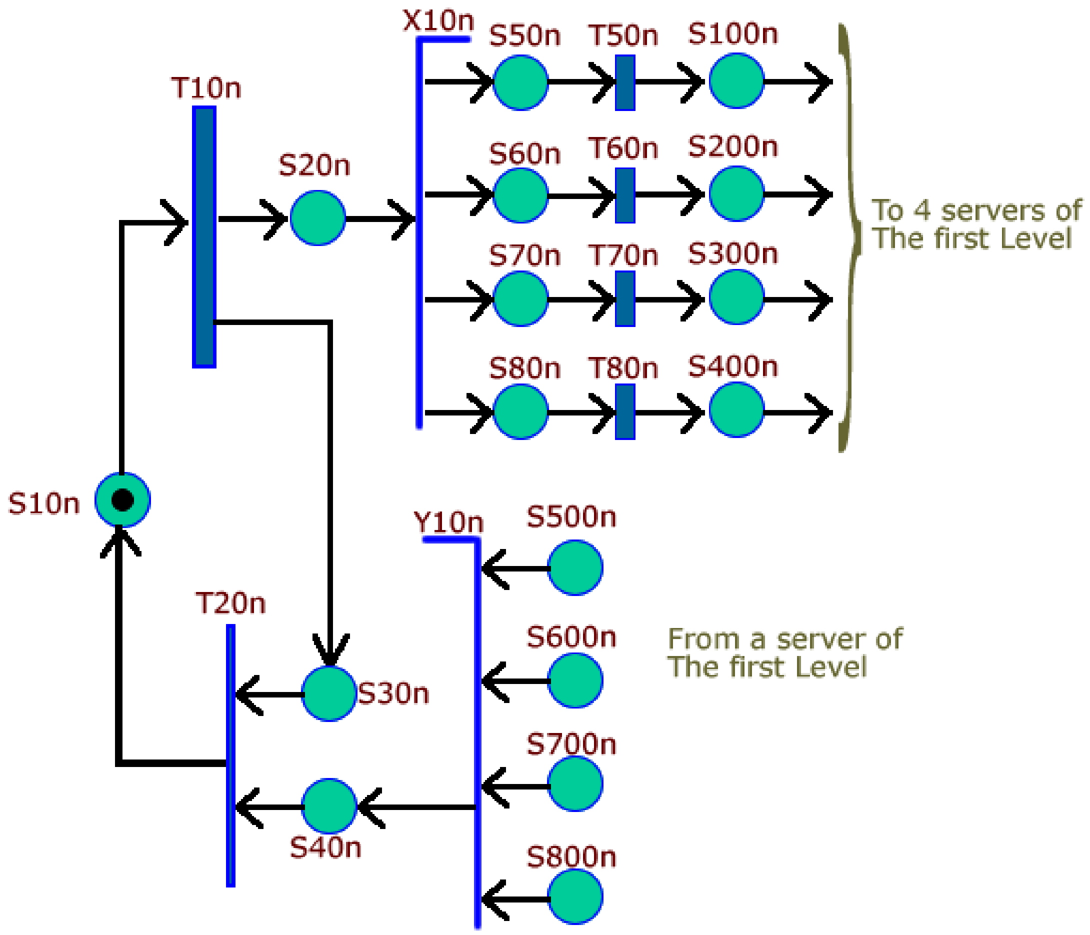

Figure 4.5 Petri-net Graph for clients.

The servers of the first level of the hierarchy (head-end) take the token from the S places identified by S100n - S400n. The presence of a token in places S500n-S800n and a the transition in Y10n shows that a message is received from the head-end servers (in the specific Petri-net graph there are 4 head-end servers). The presence of a token in S place S40n means that the client has received a message and as soon as a token is available in place S40n that transition, T20n is fired, meaning that the request has been received and the client can continue its work or make a new request. In this simulation model, we assume, for simplicity, that there are 3 clients and three levels of hierarchy in the distributed media server although there can be potentially thousands of clients and many levels of hierarchy.

4.3 Model for Intermediate Level Servers

The intermediate level servers are the servers of level 1 and level 2. They are unique from the root level servers because they have both inputs from and outputs to lower level of the hierarchy, may it be a server or a client making the request, and inputs to and outputs from higher level of hierarchy in the distributed continuous media server system under simulation.

The Petri-net model for the intermediate level servers can be described in the following manner using Figure 4.6.

The presence of a token in any of the places S1000n-S3000n would mean that a message is received from a client (in case the receiving server is a head-end) or a server of level 1 (in case the server receiving the request is a level 2 server).

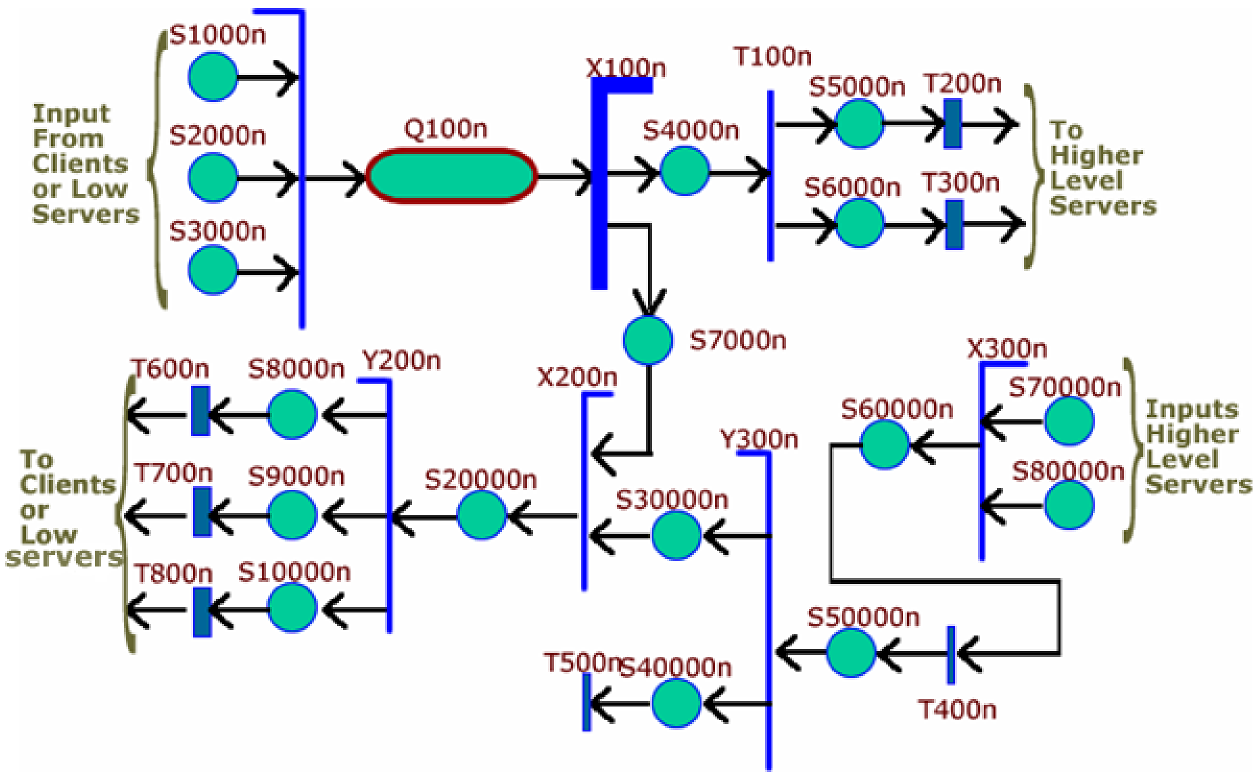

Figure 4.6: Petri-net graph for Intermediate Level Servers.

This would immediately cause the transition Y100n to fire. The fired transition from Y100n would be cause addition of a token in queue place Q100n. The queue place is necessary because potentially different clients/server can make a request at the same time or a new request from a different client/server may be received before a previous request is served by the specific server of the system. The server takes a request from the queue, represented by removal of a token from the queue, and starts checking its index for the possible presence of the information required. This is shown by firing of transition X100. It can be seen from Figure 4.6 that X100 is shown using thick lines. The thickness is for normally distributed random delay with mean value T_S necessary to represent the time consumed by the server in searching its index. There are two possible outcomes of this search, either the content being searched is found or not.

In case the content is found locally by the current server, an Ok message is sent to the client. This process is depicted by presence of a token in S7000n place and firing of transition X200n. When transition X200n is fired, a token is placed in S2000n place meaning that the request made by a client is responded and the transition Y200n can fire. This transition (Y200n) places the token into place S8000n, S9000n or S10000n after checking the attributes of the request message to find out which client made the request. The presence of a token in S8000n, S9000 n or 10000n would cause the corresponding transition T600n-T800n to fire. These transitions are responsible to show that the output has been sent to the network for the specific low level server or client to receive. Transitions T600n-T800n also manage the simulation of network transition time delay according to an exponentially distributed random factor based on some mean value $T_{N.}$

If the content is not available in the server locally then the request message is sent to servers of higher level for a further search by them in their content base. This process is illustrated by presence of a token in place S4000n causing the firing of transition T100n. This transition creates an entry in an array of the request being sent to both its parent servers and it increments the counter to 2. After making the entry of the request being sent to the servers in higher level of hierarchy tokens are placed in S places S5000n and S6000n, this causes the transition T200n and T300n to fire after an exponentially distributed random delay with a mean value of T_N.

The presence of a token in either of the places S70000 n or S80000n represents receipt of a reply from the servers in higher level of the hierarchy. This causes the transition X300n to fire and places a token in S place S60000n. whenever a token becomes available in S60000n, the transition T400n fires. This transition is responsible to find the entry of the corresponding request message in the array of requests sent by this server to the servers in higher level of hierarchy. If this reply is positive, meaning that the content is available in the server sending this reply, then a status flag is set to 1 and the counter held by this server is decremented by 1. If the negative message is received then only the counter is decremented by 1 and the status flag is left unchanged. The status flag is used to find out if this is the first negative message received or the second. This information will be useful later on. Firing of the T400 transition places a token in S50000n place which causes transition Y300n to become ready to fire. Transition Y300n determines if this is the first reply. If that is the case then it further determines if this is a positive reply or negative reply. If it is a negative reply then it is directed to S40000n place which is actually a sinking place, meaning that this token is wasted by firing of transition T500n. If transition Y300 determines that this is a second negative reply or a first or second positive reply then it is directed to S30000n place and an entry is made in the array kept by this server for maintaining the request status stating that a reply has been directed to the server of lower level of hierarchy. A token in S30000n place causes transition X200n to fire which sends the response to the server of lower level (or the client if this is a head-end server) through the Internet as explained above in this section. The array maintained by the intermediate level servers is schematically illustrated in Figure 4.7.

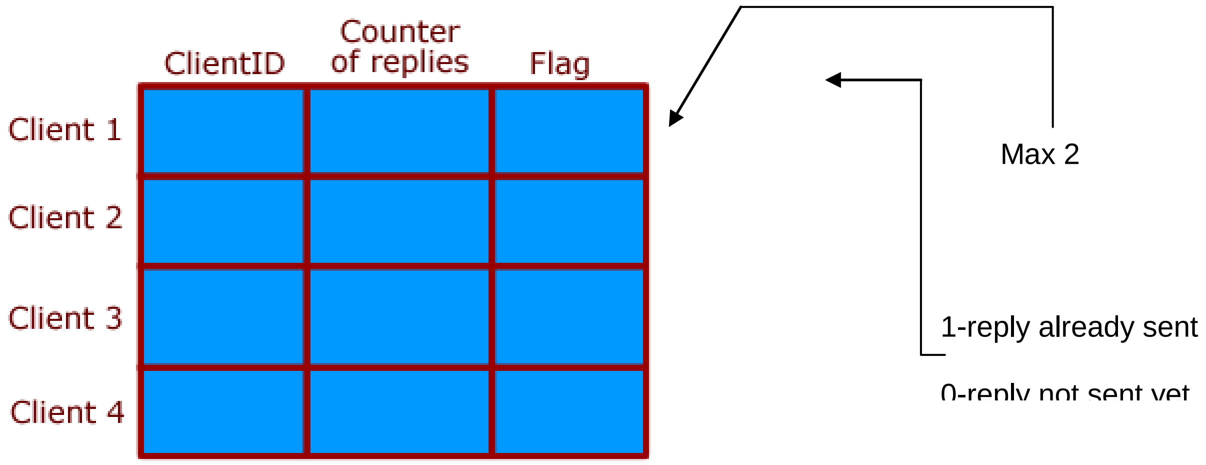

Figure 4.7: Status Array kept by intermediate level servers.

4.3 Model for Root Level Servers

The fundamental concept behind the model for the root level servers is the same as that of the intermediate level servers except they do not have any input from higher servers. Thus, they make no inquiry about the required content from other servers. The change in the design of model of root level servers is in the transition X1000n. If upon looking up the index of its content, the root level server finds that it does not contain the required file, it sends a negative message to the server that had sent the query message. This is done by placing of a token in place S500000n with attribute TYP set to 3 meaning that the content is not available in this server. Furthermore, this server does not maintain any array of status as it does not need any such mechanism. The Petri-net graph representing the working of root level servers is shown in Figure 4.8.

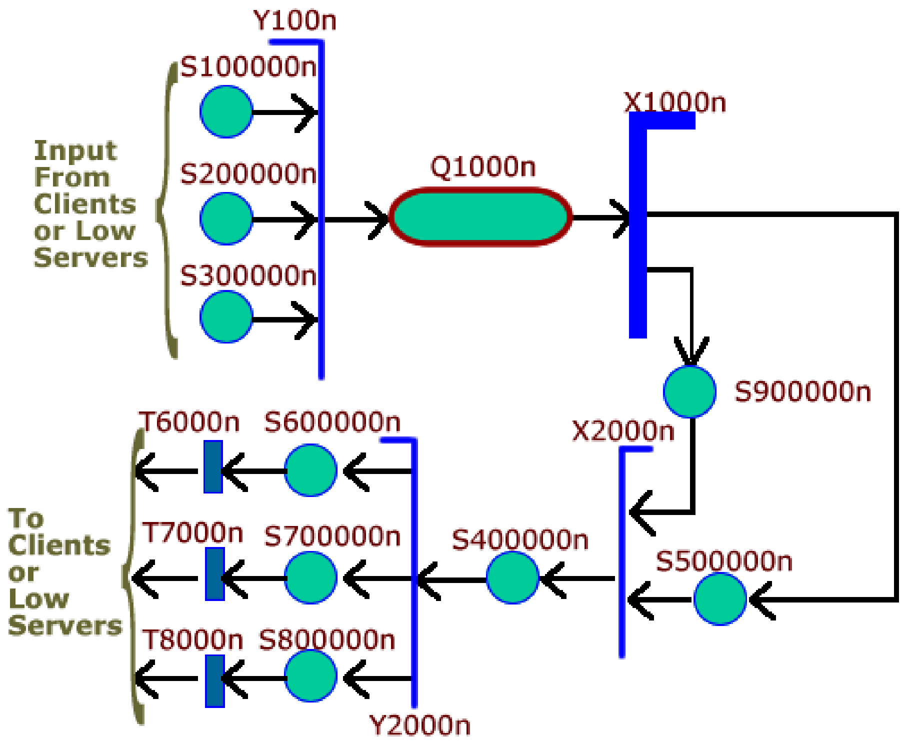

Figure 4.8: Petri-net Graph for a root level server.

4.4 Cost Functions

To balance the load, one can choose the least loaded path among all the candidate paths. There are many ways to quantify the load of a path. One method is to take the summation of the loads of all the participating nodes and links in the path. In this case, longer paths with more links are more likely to have higher loads. With this approach, longer paths are penalized.

An alternative method would be to take the average of the loads of all the participating nodes and links in the path. This method does not provide shorter paths with any advantage. However, if a path has a number of lightly loaded links and nodes but a single nearly saturated node, this method would choose this path over an averagely loaded path. This may saturate the heavily loaded node and prevent it from serving other requests while the nodes in the other path are less loaded. Therefore, we have decided to choose the cost of the most loaded component in the path as the cost of the path. That is, the node or the link which is most loaded determines the load of its corresponding path.

We propose three cost functions corresponding to our three heuristics to measure the load of a node or a link in order to select the least loaded path. The first cost function *FreeBW* uses the available bandwidth *B* of servers or links as the load indicator. The disadvantage of *FreeWB* is that it tends to select nodes with higher bandwidth in the higher levels of the hierarchy over nodes in the lower levels of the hierarchy with less bandwidth. Therefore, a large node operating at fifty percent of its maximum bandwidth is going to be selected over a smaller node, lower in hierarchy, with same load percentage. This can unfairly saturate the higher nodes and might yield a higher communication cost since most of the objects will be retrieved from the higher levels of the hierarchy.

To overcome this drawback, a second cost function *RatioBW* is used. Instead of using available bandwidth as load indicator, RatioBW uses the available bandwidth ratio ($RatioBW = B / Max(B)$) to measure the load.

The third cost function *UserBW* employs *UserBW* employs an alternative method to eliminate the disadvantage of *FreeBW*. *UserBW* divides the available bandwidth of a node by the number of users served collectively by the node and its children. Similarly the *UserBW* of a link is the available bandwidth of a link divided by the number of user served collectively by a node and its children.

Chapter 5

RED-Hi BASED LOAD MANAGEMENT POLICY

5.1 System Architecture

A layered approach is used for placing the multimedia servers in the network because with geographically distributed servers, the connections as well as the communications between the servers (nodes) can be handled with relative ease using a layered approach. Furthermore the resources, such as bandwidth and memory can be better allocated based on the functionality of the layer that a multimedia server belongs to, as each layer has a predefined functionality in the system.

We assume three categories of layers, namely, Entry, Intermediate and Root level layers. There are single Entry Level and Root level layers in the system whereas there may be many Intermediate level layers. A simple schematic representation of the system is depicted in Figure 3.1. Redundant Hierarchy (RED-Hi) [9] is used to connect the servers in each layer to its parents in the adjacent layer.

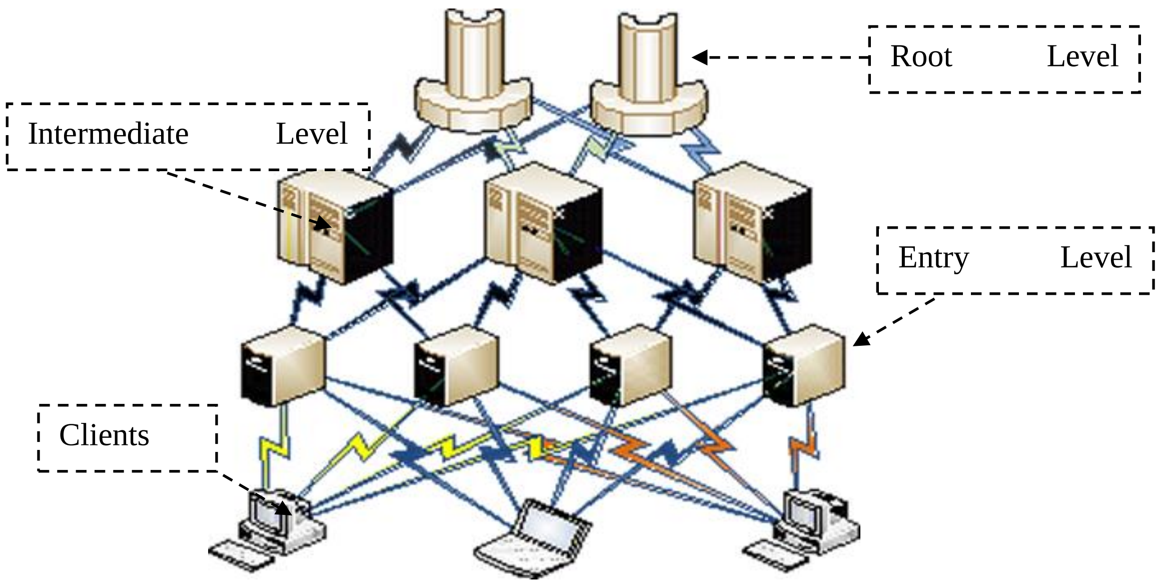

Figure 5.1: Network Architecture of the Streaming Servers in the system.

5.1.1 Assumptions

It is assumed that the following are provided for the policy to work.

a) All nodes have exactly two parent nodes based on RED-Hi [9].

b) Both the parent nodes belong to the same immediate parent layer of the node.

c) Root nodes have no parent nodes.

d) Each node can store multimedia data.

e) Each node is a multimedia server in itself with multimedia processing capabilities.

f) Only entry level servers, also known as head-end servers, communicate with the clients.

5.1.2 Entry Level Layer

This layer is responsible for all direct communication with the clients to the system. The servers in this layer act as head-ends to the system and receive all requests from the clients. Once the requested object is located, these head-end servers deliver the requested stream from the system of servers to the requesting client.

As shown in Table 5.1, it is assumed that the head-end servers have relatively low storage capacity but a high bandwidth capacity. The storage capacity of these servers is low because they only need to keep the current streaming object and a few (popular) previously streamed objects in their memory. The bandwidth capacity allocated to these servers is high as they are responsible for streaming to a possibly large set of clients. The servers in this layer have designated parent servers in the adjacent intermediate layer. Each server in this layer has exactly two parent servers in the adjacent intermediate layer that can help it in realizing the streaming of a requested multimedia object.

Table 5.1 Resource distribution in the system.

Layer	**Capacity**	**Bandwidth**
Root Level Layer	High	Low
Intermediate Level Layer	Medium	Medium
Entry Level Layer	Low	High

5.1.3 Intermediate Level Layer

The servers in this layer act as intermediaries to the system. When a request is received from a lower level server and the requested object is not available in the memory of intermediate level servers, it forwards the request to the layer above and waits for the response from that higher level layer. Once the requested object is located, the corresponding intermediate level server informs the lower level layer about the location and resources required, along with the current load on the servers involved in the path of the requested object.

Once the streaming commences, a server of this layer involved in streaming of a multimedia object forwards the packets available locally in its memory or retrieved from a higher layer to the requesting server in the adjacent lower level layer.

Each server in this layer has a small set of child nodes from the adjacent lower layer that depend on it to locate and deliver multimedia objects. At the same time, each server in this layer has exactly two servers in the adjacent higher layer that can help it in realizing the streaming of a requested multimedia object.

The servers of the intermediate level are given medium bandwidth and storage capacities as they tend to the needs of a small set of the system requests and as such do not have a very high bandwidth requirement. Furthermore, they keep the currently streamed objects as well as recently streamed objects in their repository and hence need a higher storage capacity as shown in Table 5.1.

In a large system containing multiple Intermediate level layers, as you go higher in the hierarchy, the bandwidth requirements are reduced, on the other hand the storage requirements increase for each increased level in the hierarchy.

5.1.4 Root Level Layer

This layer is the main repository of all streams available to the system. If a requested object is not found in the servers of the root level then that object is not available in any server of the system. The servers in the root level layer are called root servers. Each of these root servers is responsible for requests coming from a subset of servers from the lower level layer (intermediate layer). These servers need high storage capacity as they keep record of all multimedia objects in the system. The root servers should not be actively participating in most of the streaming activities in the system. Hence, low bandwidth capacity is sufficient for these servers.

5.1.5 Server Connections

Physical communication connections exist between the servers of different layers. All connections are duplex and hence can be used for bidirectional communication. There are no connections between the servers of the same layer. The network architecture used is based on Redundant Hierarchy (RED-Hi) [9].

In RED-Hi a node is connected to exactly two nodes of the higher layer in the hierarchy, the redundancy in RED-Hi is that of links and not bandwidth.

This means that instead of doubling the bandwidth to increase the link, the available bandwidth is divided among the two links. This gives a higher degree of connectivity and it is a solution to the bottleneck problem. This also provides continued service provision in face of node/link failures [9].

5.2 Request Life Cycle

A request for a multimedia object is created by a client. The client sends this request to the system. The system receives the request through an entry level server (head-end). The head-end checks its memory for the requested object, as depicted in

Figure 5.2. If the requested object is found, it is streamed to the requesting client. If the requested object is not found in the memory of the head-end, the head-end generates a query packet and forwards a copy of that query packet to both intermediate level servers who are its parents.

These parent intermediate level servers, upon receiving the query packet, check their own repository for the queried item. If the object is available, a positive acknowledgement is sent to the server querying for the object. Otherwise, the query packet is forwarded to the two servers in the higher layer of the hierarchy that are parents of the current server.

In this way, the query packet will eventually reach a server that contains the requested media or a negative acknowledgement will be generated at the root server.

In case the requested media is located at an intermediate or root server, a positive acknowledgement is passed down the hierarchy to the head-end responsible for initializing this query packet. The head-end may receive multiple acknowledgements. In that case it would select the path with higher resources. The resources available to the server and all intermediate servers will be mentioned in the positive acknowledgement packet.

Once the path with highest resources is selected and reserved for streaming, the head-end propagates a packet, through the reserved path, to inform the server containing the multimedia packet to start streaming. Upon receiving this packet, the server starts streaming the multimedia object down the hierarchy.

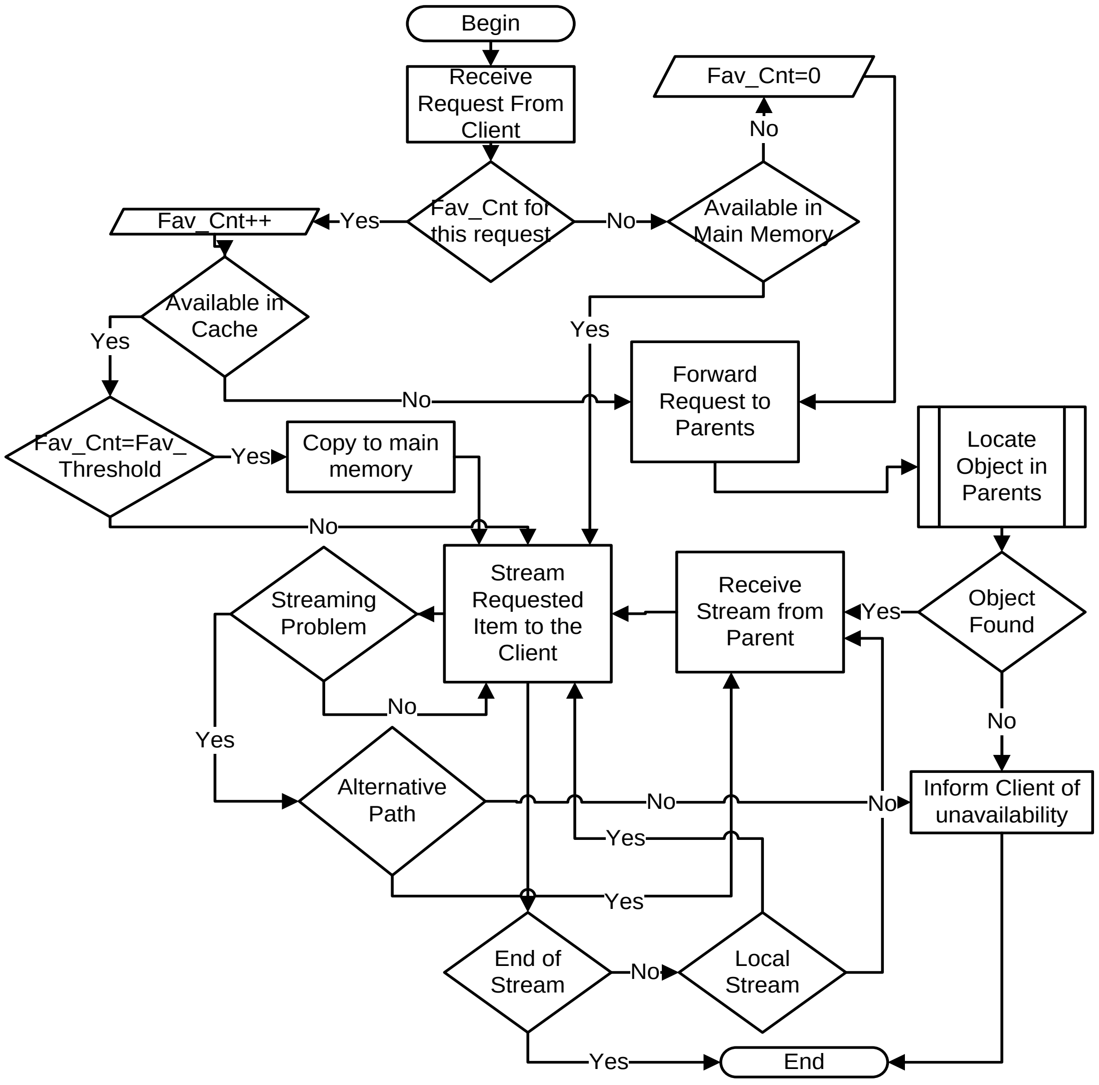

Figure 5.2: Basic flowchart representing the functioning of a node.

5.3 Dynamic Object Placement

Objects are placed dynamically using the ‘popularity’ counter of each streamed object. Each server in the system maintains its own counter of popularity for each object streamed through it.

Each server that receives a streaming packet from its parent server, saves that packet in its local cache memory and then forwards it down the stream. This saving is done to minimize the number of servers involved to manage a possible packet-loss event at the lower layers.

The server receiving the stream packet increments the popularity counter of the currently streamed object by one whenever a new request/query message for this particular object is received by this server. Once the popularity counter reaches a predefined popularity threshold, that popular object is moved from the cache memory into the main memory of that server as shown in Figure 5.3.

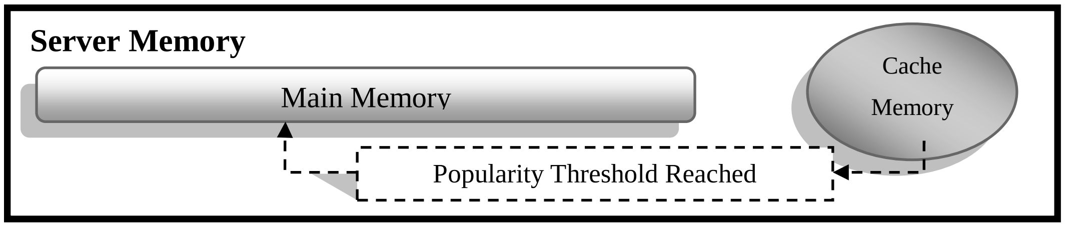

Figure 5.3: Moving an Object from Cache to Main Memory of a Server.

The popularity counter helps in determining the popularity of objects among a set of servers and dynamically places those objects in close vicinity of the requesting clients. Dynamic object placement makes it possible to populate the multimedia objects based on their demand zone and hence optimizes the servers' usage of memory and bandwidth resources.

Once the Main memory gets low on storage space, the least recently used media object is removed from the node's memory. Similarly the least recently used video object is removed in case the Cache memory becomes full. No object is removed from the root nodes.

5.4 Fault Tolerance

A general architecture of client-server system is presented in Figure 5.4. This figure depicts a generic end-to-end media delivery components involved in a DCMS system. In the storage subsystem, the server holds compressed and encoded multimedia objects that are, upon request, retrieved from the storage devices and sent over the network from the server to the client.

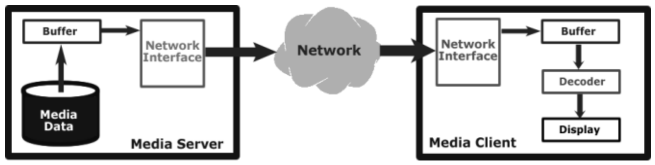

Figure 5.4: Basic building blocks of a multimedia system.

Protocols, such as the application protocol is used by the server to handle the requests retrieval and transmission, whereas the transport protocol is used to transfer the multimedia object to the requesting client. The client, upon receiving the requested object packets — buffers and decodes them before presentation can be given to end user.

If a situation arises in any of these subcomponents, like the processors involved the storage or network hindering the flow of the media data, it can degrade the performance of the whole system. It is imperative that any end-to-end multimedia system provides tolerance to a failure that occurs in as many of these components as possible.

In the object location phase, all possible paths to the requested object are saved at all included nodes. The best path is selected for streaming initially. However, whenever a node feels that problems in its currently streaming parent node are inhibiting its performance, it changes the streaming path using the other parent node, if possible.

As all available multimedia objects are kept in both root servers, it is always possible for a head-end to find at least two different paths for a given multimedia object, provided the root servers are not the ones with the fault.

For popular objects, multiple copies at multiple nodes on multiple layers are available making them more accessible. Even in case of node/link failures in the streaming path, the worst case scenario would be the delay caused by a change in the nodes involved in the streaming. This delay is of negligible time as it would not be required to locate the object from the beginning.

Any failure that occurs during object delivery can be managed by changing the nodes included in the streaming path other than a physical failure in the head-end.

In case a head-end component fails, the client has to redirect its request to some other head-end after a timeout. The redirect process will include fresh new object location and object delivery phases.

Figure 5.5 gives the flowchart for the node that detects a problem in the streaming process. Upon realizing that the streaming process is suffering due to some server component in the higher layers, the node changes the path from itself up to the server that commences the streaming process.

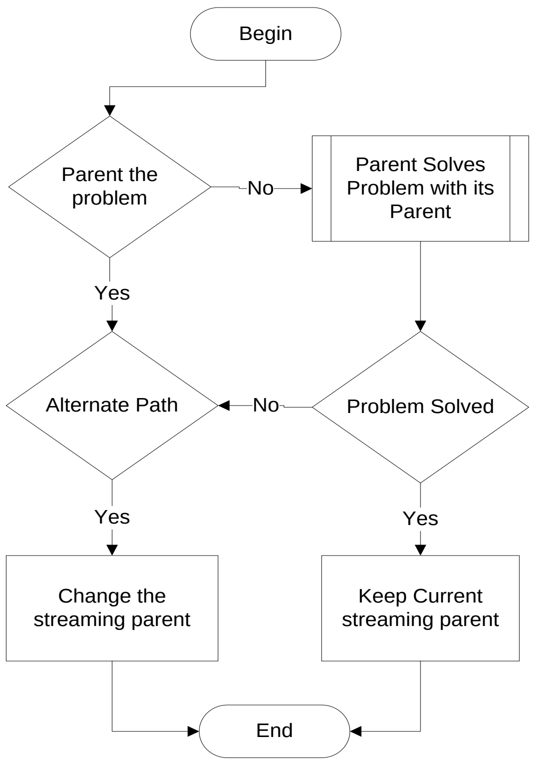
Begin
Parent the problem
No
Parent Solves Problem with its Parent
Yes
Alternate Path
No
Problem Solved
Yes
Yes
Change the streaming parent
Keep Current streaming parent
End

Figure 5.5: Path change flowchart at a node.

Chapter 6

SIMULATION FRAMEWORK

This chapter details the simulation framework. This framework includes the simulation tool, its model and parameters. The chapter is organized as follows: section 6.1 details the tool used for simulation, 6.2 gives the architecture of the RED-Hi and NonRED-Hi network models. Section 6.3 gives definitions of the parameters of the simulation itself.Section 6.4 provides an overview on performance measures.

6.1 Simulation tool

In this study, all simulation studies were developed using the GPSS World simulation tool. GPSS World is based on the seminal language of computer simulation, GPSS, which stands for General Purpose Simulation System.

6.2 Network Architectures of RED-Hi and NonRED-Hi Models

We simulated the behavior of RED-Hi and NonRED-Hi models as depicted in Figures 6.1 and 6.2, respectively. NonRED-Hi environment is one in which the servers are connected using pure hierarchy, i.e. where each server in the system has exactly one parent [103]. Both RED-Hi and NonRED-Hi architectures consist of an Entry Level Layer with five nodes, Intermediate Level Layer 1 with four nodes, Intermediate Level Layer 2 with three nodes, a Root Level Layer with two and one nodes, correspondingly.

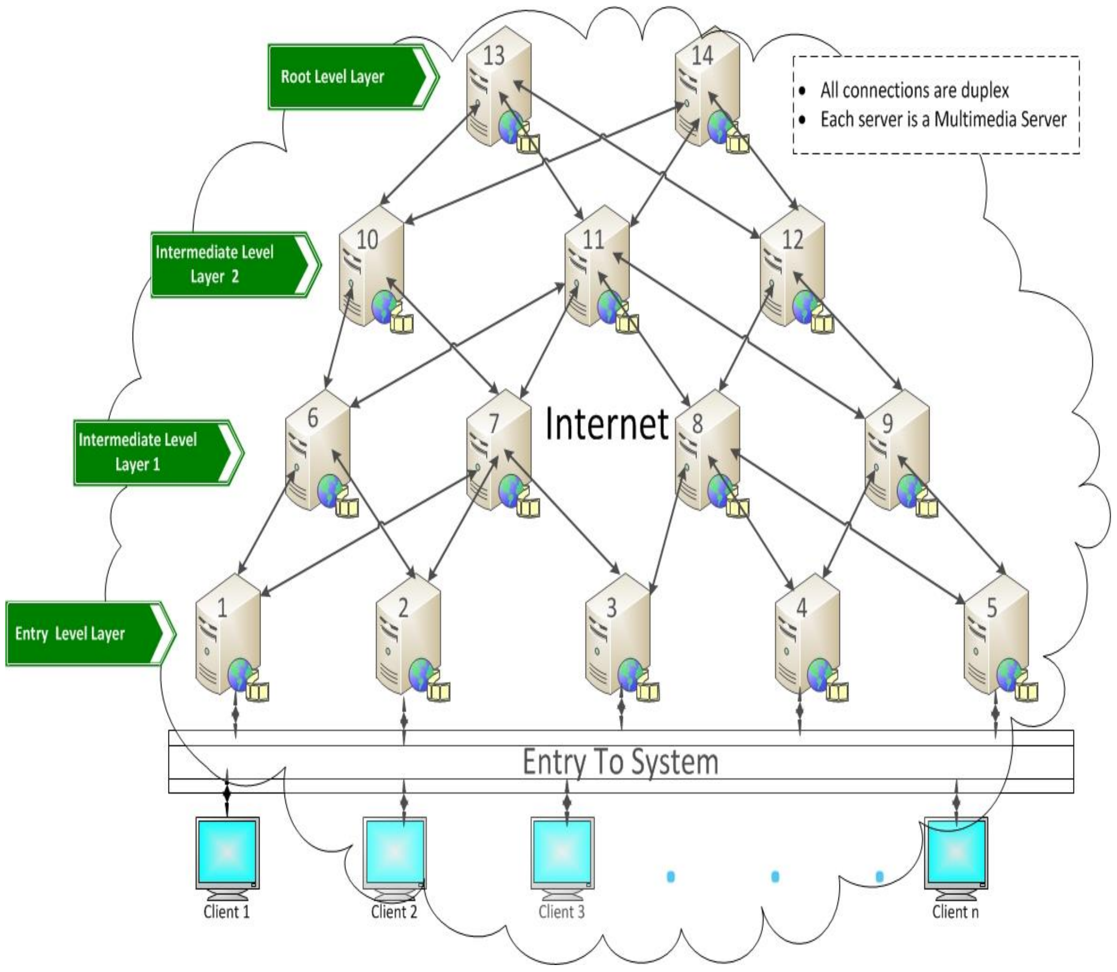
Root Level Layer
Intermediate Level Layer 2
Intermediate Level Layer 1
Entry Level Layer
13
14
All connections are duplex
Each server is a Multimedia Server
10
11
12
Internet
6
7
8
9
1
2
3
4
5
Entry To System
Client 1
Client 2
Client 3
Client n

Figure 6.1: Network architecture of the RED-Hi model.

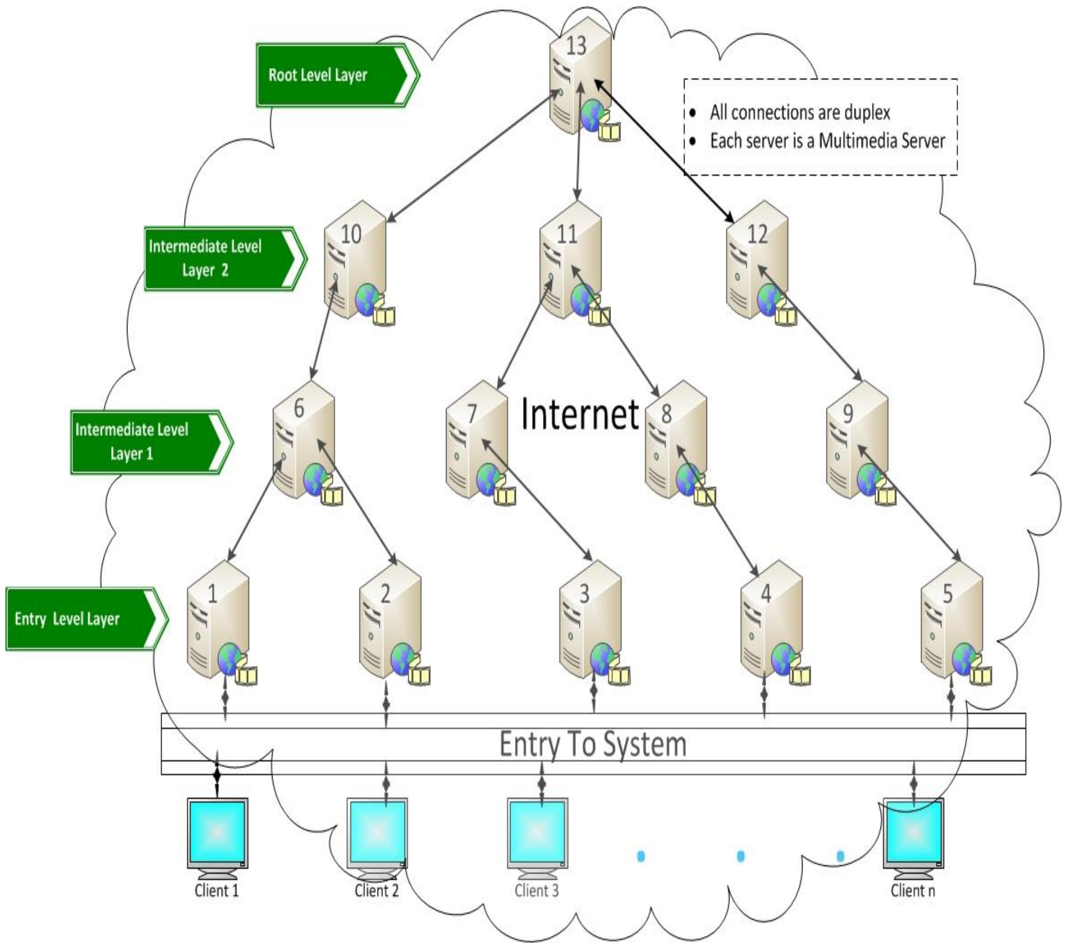
Root Level Layer
13
All connections are duplex
Each server is a Multimedia Server
Intermediate Level Layer 2
10
11
12
Intermediate Level Layer 1
6
7
Internet
8
9
Entry Level Layer
1
2
3
4
5
Entry To System
Client 1
Client 2
Client 3
Client n

Figure 6.2: Network architecture of the NonRED-Hi model.

6.3 Simulation Parameters

Table 6.1 provides a summary of the simulation parameters used.

Table 6.1: Simulation parameters.

Parameter
Request arrival rate
Popularity Threshold
Storage size of an object
Bandwidth requirement of an object
Total number of objects

6.4 Performance Measures

This thesis evaluates the performance of RED-Hi based DCMS system in terms of Average Transmission Delay, Average Communication Delay of Successful Requests, Average Number of Control Messages of Successful Requests, Average Number of Traversed Nodes of Successful Requests, Average Number of Hops of Successful Requests, Blocking Ratio and Load Distribution, while the RED-Hi and NonRED-Hi models were compared based on Blocking Ratio and Load Distributions.

Average Transmission Delay (ATD) is defined as follows:

$$\mathbf{ATD = D_s / N_s} \tag{6.1}$$

where D_s is the total delay originating from successful requests and N_s is total number of successful requests.

Average Communication Delay of Successful Requests (ACDSR) is defined as follows:

$$\mathbf{ACDSR = D_c / N_s} \qquad (6.2)$$

where D_c is the total communication delay of successful requests and N_s is the total number of successful requests.

Average Number of Control Messages of Successful Requests (ANCMSR) is defined as follows:

$$\mathbf{ANCMSR = M_c / N_s} \qquad (6.3)$$

where M_c is the total number of control messages of successful requests and N_s is the total number of successful requests.

Average Number of Traversed Nodes of Successful Requests (ANTNSR) is defined as follows:

$$\mathbf{ANTNSR = T_t / N_s} \qquad (6.4)$$

where T_t is the total number of traversed nodes by successful requests and N_s is the total number of successful requests.

Average Number of Hops of Successful Requests (ANHSR) is defined as follows:

$$\mathbf{ANHSR = H_t / N_s} \qquad (6.5)$$

where H_t is the total number of hops by successful requests and N_s is the total number of successful requests.

Blocking Ratio (BR) is defined as follows:

$$\mathbf{BR = N_b / N_t} \qquad (6.6)$$

N_b is the total number of blocked (rejected) requests and N_t is the total number of requests.

Chapter 7

SIMULATION RESULTS

This chapter examines the simulation results gathered for both RED-Hi and NonRED-Hi (Pure Hierarchy). The performance of the RED-Hi scheme was evaluated in terms of Average Transmission Delay, Average Communication Delay of Successful Requests, Average Number of Control Messages of Successful Requests, Average Number of Traversed Nodes of Successful Requests, Average Number of Hops of Successful Requests, Blocking Ratio and Load Distribution, while the RED-Hi and NonRED-Hi models were compared based on Blocking Ratio and Load Distributions

7.1 Values of the Simulation Parameters

We assume that the propagation delay of links follows a normal distribution with mean of 0.07 sec and standard deviation 0.014. The flow of all the client requests in both of the models is modeled as a Poisson process whose rate is λ. The simulation results were obtained for arrival rates of 0.05, 01, 0.15 and 0.2.

In each simulation 1,000,000 requests were generated and each simulation was run 20 times to ensure that the obtained results are in the 95% confidence interval level [104]. Table 7.1 contains the values of the simulation parameters.

Table 7.1: Values of the simulation parameters.

Parameter	Values
Request arrival rate	0.05, 0.1, 0.15, 0.2 s^{-1}
Popularity Threshold	5, 25, 50
Storage size of an object	0.5 GB
Bandwidth requirement of an object	1 Mb/s
Total number of objects	200

7.2 Performance Analysis of RED-Hi

In this section, simulation results of the RED-Hi scheme are discussed. We investigate the effect variation in the popularity threshold and arrival rate has on the performance of the RED-Hi model.

7.2.1 Average Transmission Delay (ATD)

We start with ATD versus arrival rate with popularity threshold values of 5, 25 and 50 depicted in Figure 7.1. The graph clearly displays that there is not a significant increase in the transmission delay caused by increasing the number of servers or with substantial increase in the arrival rate of the requests.

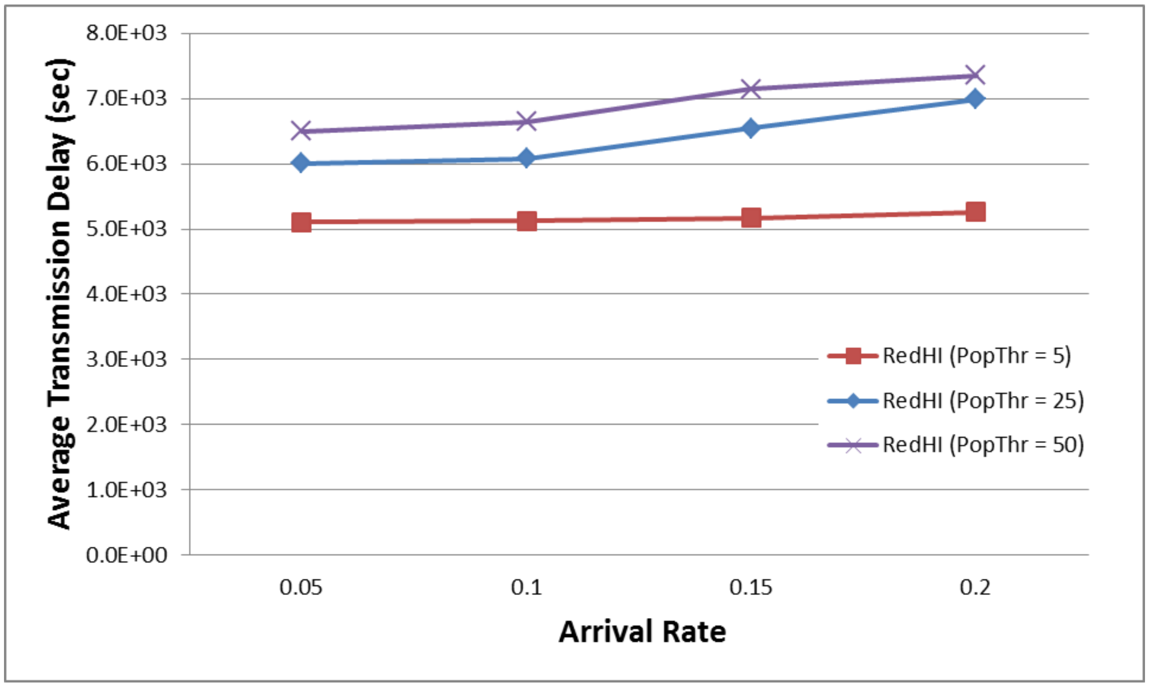

Figure 7.1: ATD versus interarrival time for RedHI scheme with Popularity Threshold values of 5, 25 and 50.

7.2.2 Average Communication Delay and Average Number of Control Messages of Successful Requests

We continue by examining how ACDSR and ANCMSR are affected by the varying arrival rate and popularity threshold. It is seen in Figures 7.2 and 7.3 that with increase in the arrival rate the communication delay, the number of control messages in the system remain fairly constant. This gives the system scalability in case a large number of requests must be handled.

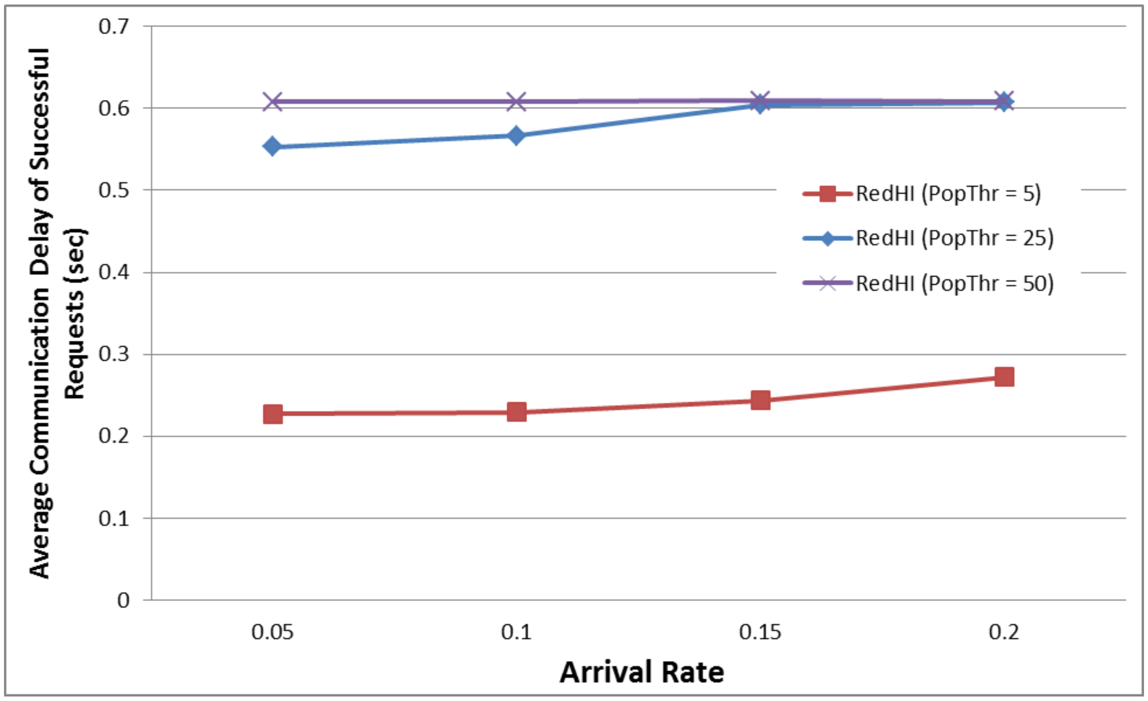

Figure 7.2: ACDSR versus interarrival time for RED-Hi scheme with Popularity Threshold values of 5, 25 and 50.

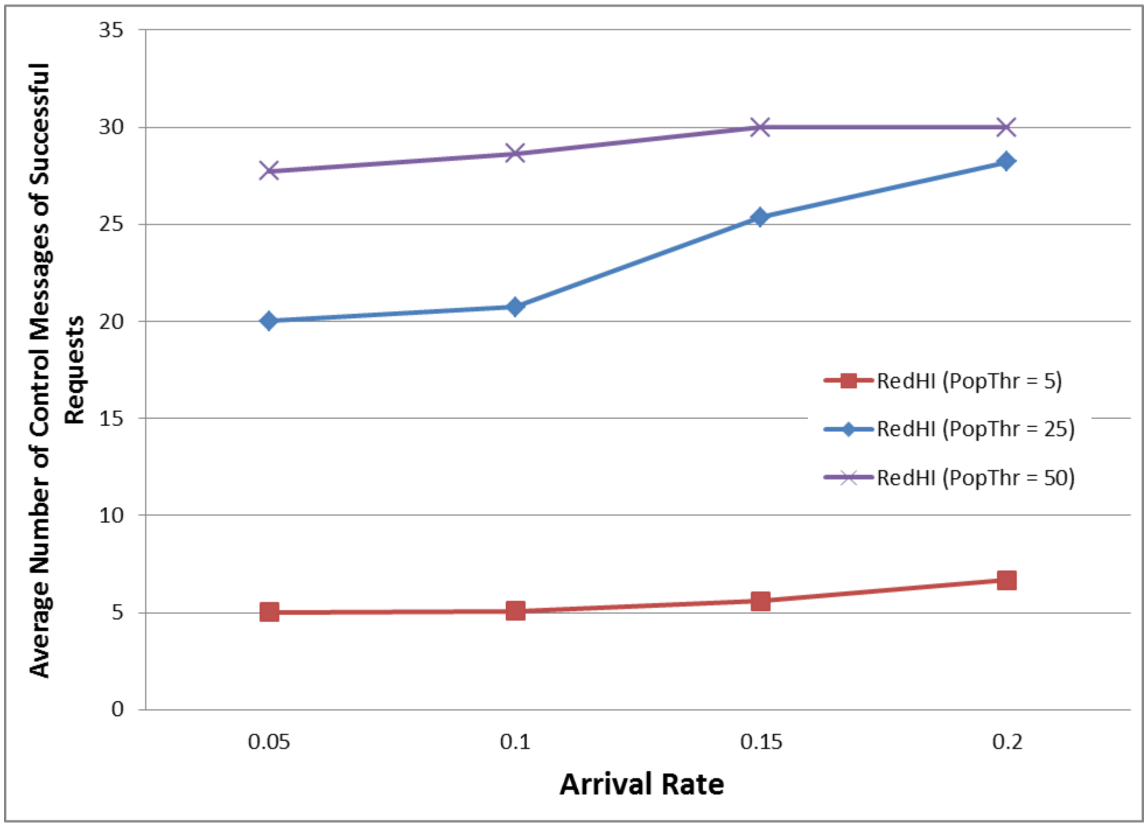

Figure 7.3: ANCMSR versus interarrival time for RED-Hi scheme with Popularity Threshold values of 5, 25 and 50.

7.2.3 Average Number of Traversed Nodes of Successful Requests and Average Number of Hops of Successful Requests

Next, we consider how RED-Hi performs in terms of Average Number of Traversed Nodes (ANTNSR) and Average Number of Hops of Successful Requests (ANHSR). It is obvious that in a redundant hierarchy, the number of nodes traversed and number of hops for a higher popularity threshold would be higher as is depicted in results graphed in Figures 7.4 and 7.5. With a popularity threshold of 50 requests for the same object, the system has to stream from the root level much more then for a popularity level of 5 requests making an object popular.

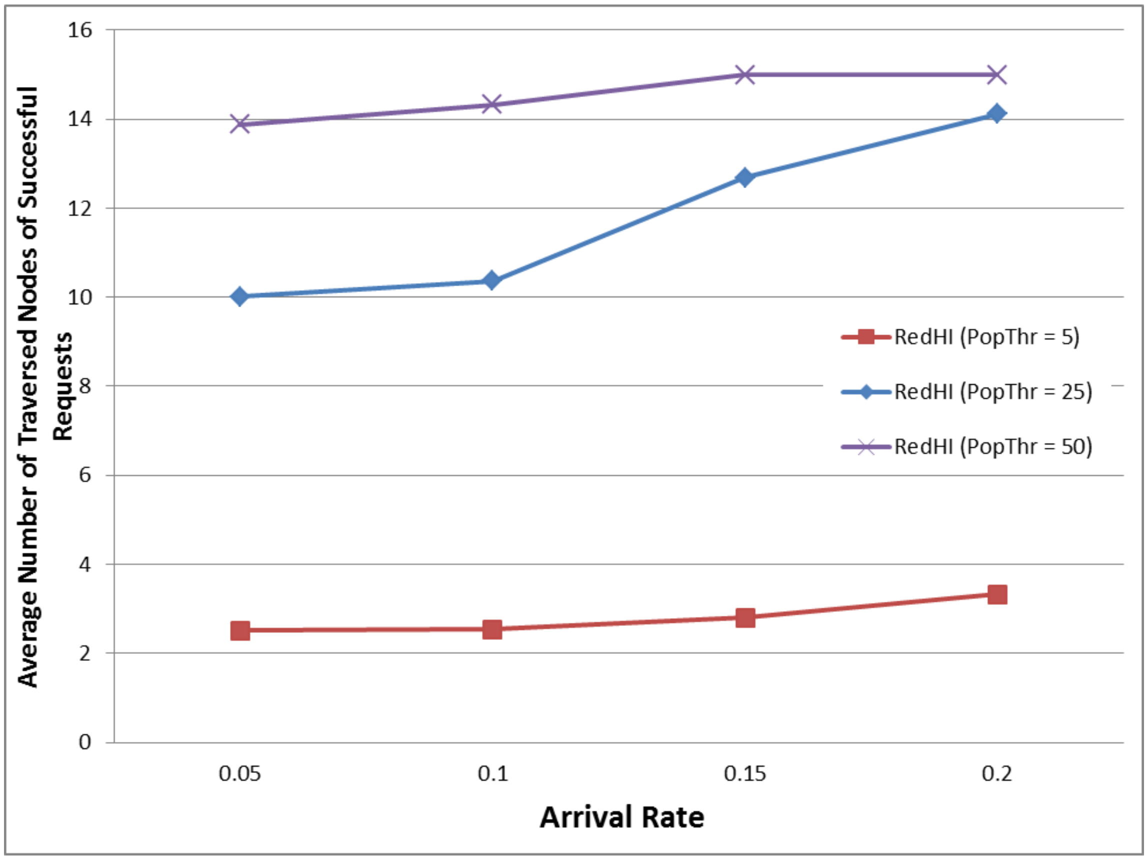

Figure 7.4: ANTNSR versus interarrival time for RedHI scheme with Popularity Threshold values of 5, 25 and 50.

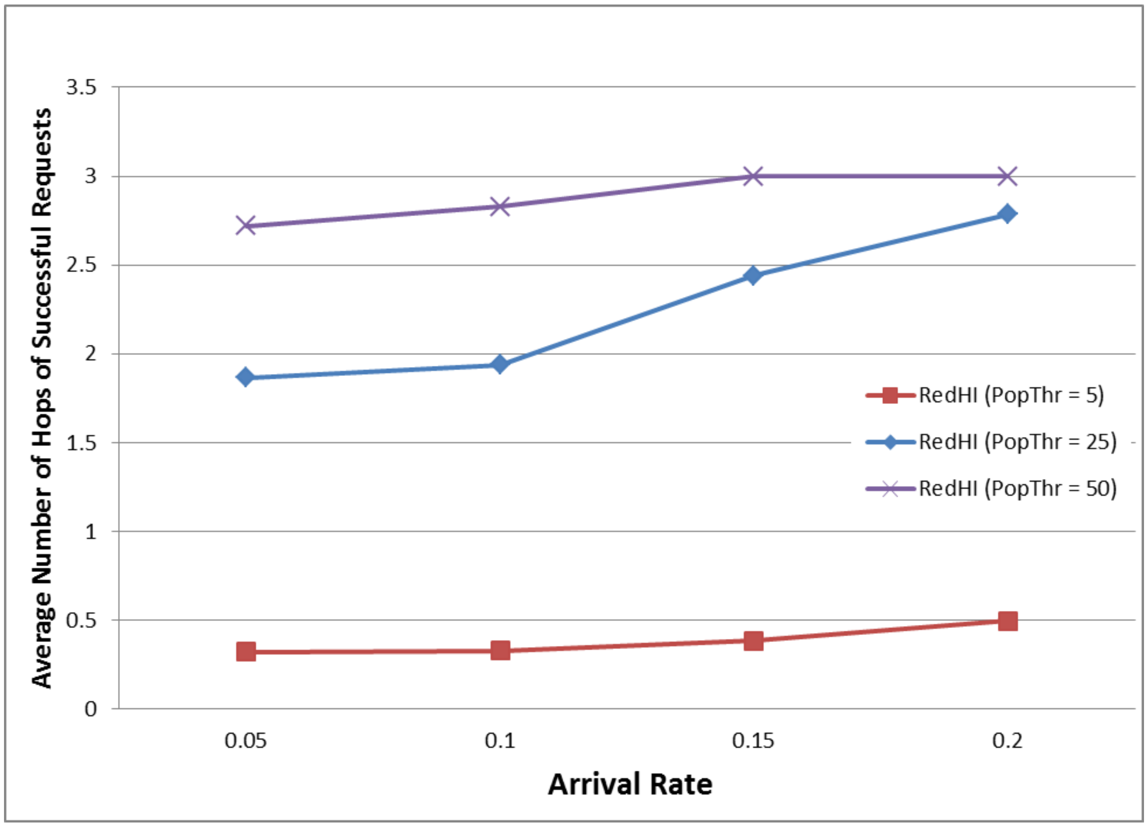

Figure 7.5: ANHSR versus interarrival time for RED-Hi scheme with Popularity Threshold values of 5, 25 and 50.

7.2.4 Blocking Ratio

Now we look at the effects of varying arrival rate and popularity threshold on the Blocking Ratio (BR). Figure 7.6 shows that BR values are increasing as the popularity threshold increases. RED-Hi model with popularity threshold 5 achieves the lowest BR. When the popularity threshold is fairly low it causes the content to reach the head-ends faster as such the clients get service directly from the head-ends without the need to get service from the intermediate or root nodes. This naturally reduces the blocking ratio in load balanced RED-Hi.

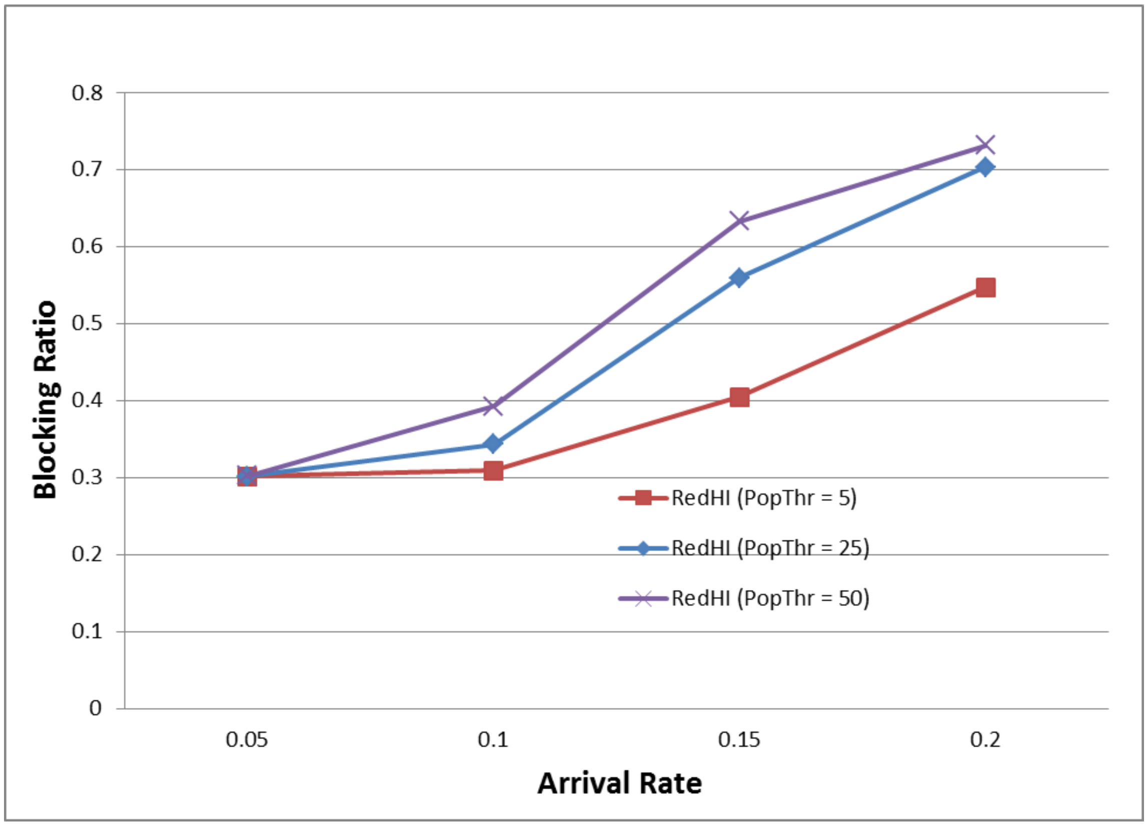

Figure 7.6: BR versus interarrival time for RED-Hi scheme with Popularity Threshold values of 5, 25 and 50.

7.2.5 Load Distribution

We finalize the performance evaluation of the RED-Hi scheme by inspecting its performance in terms of load distribution. The following figures depict load distribution of RED-Hi under various arrival rates.

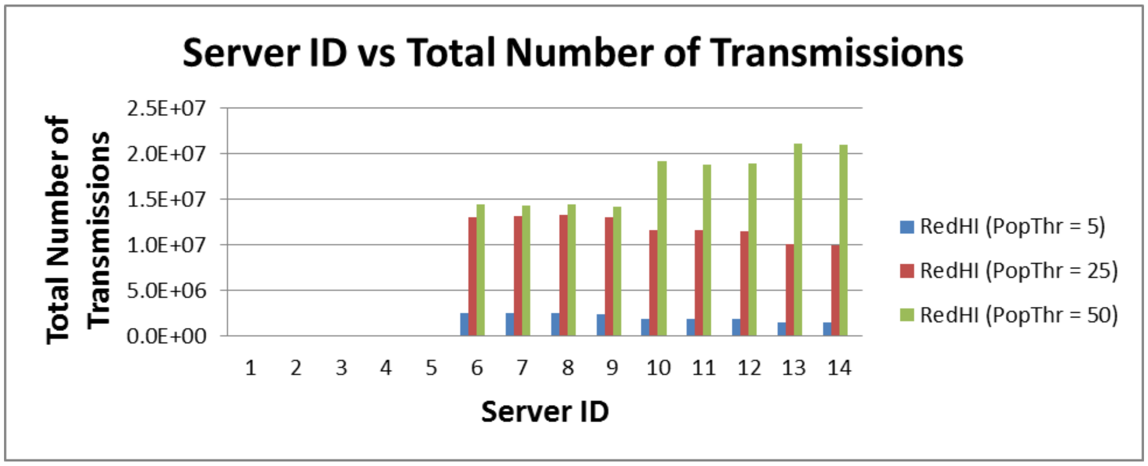

Figure 7.7: Server ID versus Total Number of Transmissions with arrival rate 0.05 and Popularity Threshold values of 5, 25 and 50.

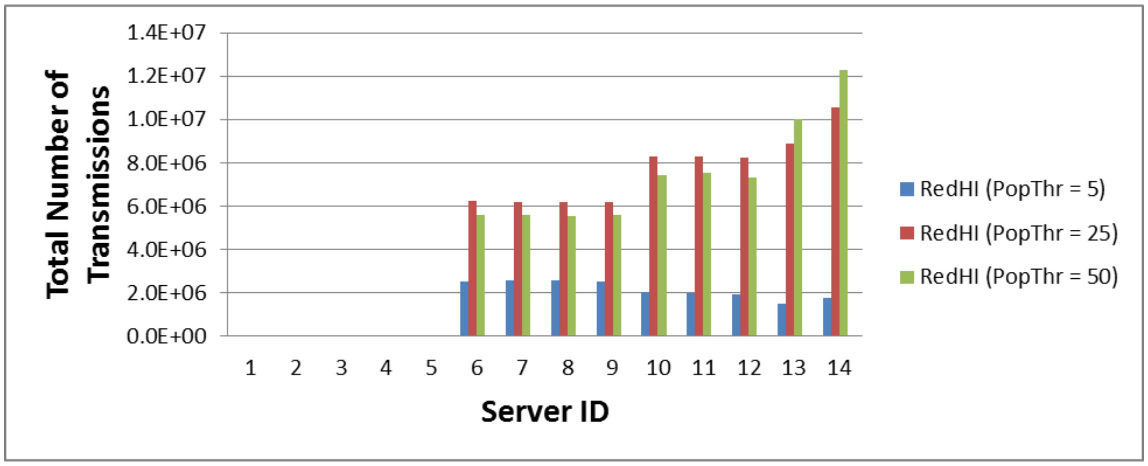

Figure 7.8: Server ID versus Total Number of Transmissions with arrival rate 0.1 and Popularity Threshold values of 5, 25 and 50.

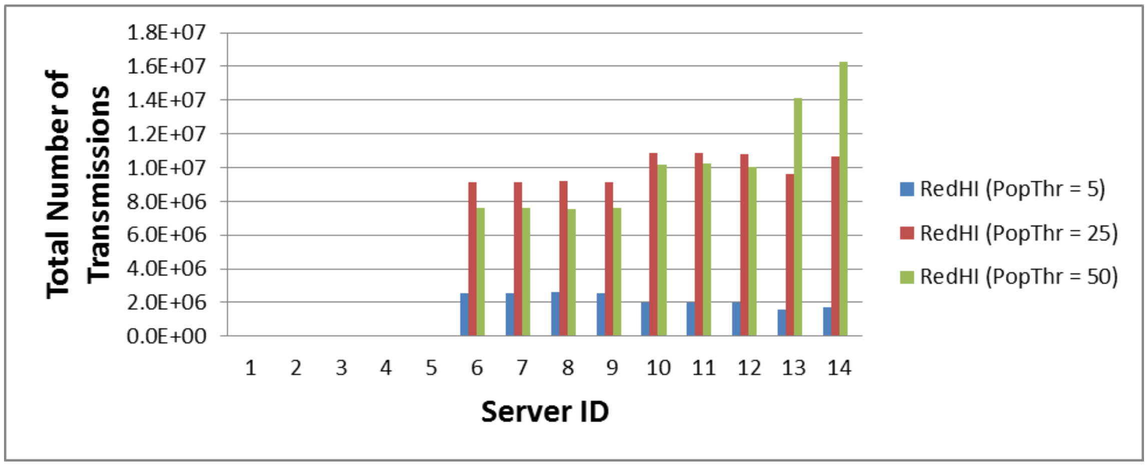

Figure 7.9: Server ID versus Total Number of Transmissions with arrival rate 0.15 and Popularity Threshold values of 5, 25 and 50.

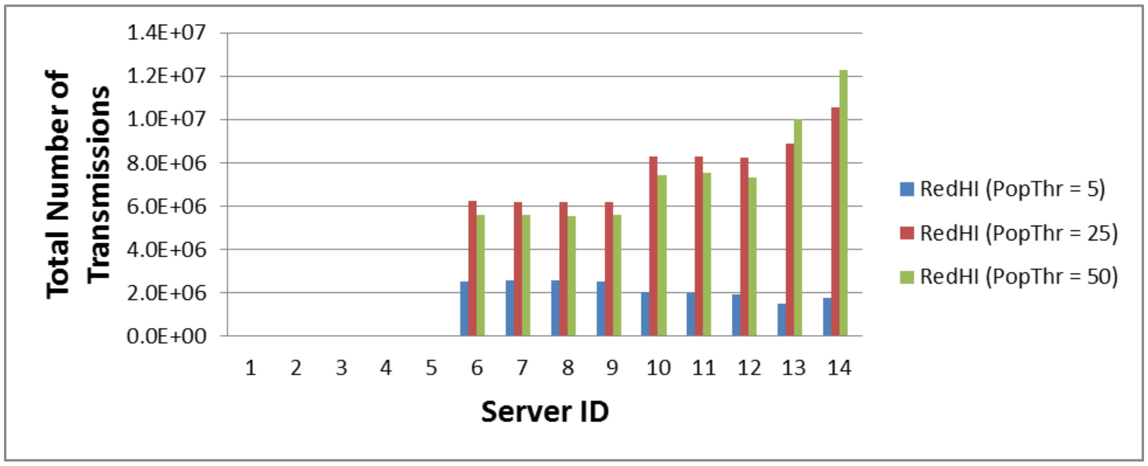

Figure 7.10: Server ID versus Total Number of Transmissions with arrival rate 0.2 and Popularity Threshold values of 5, 25 and 50.

Looking at Figures 7.7 to 7.10, with increasing arrival rate, the system remains very scalable and the load on the serves remains balanced. These results show that the system achieves its purpose of load balancing under very high arrival rate of requests for streaming.

7.2.6 Overview of the RED-Hi performance

To sum up, Figures 7.1 to 7.6 illustrate that RED-Hi with Popularity Threshold 5 outperforms RED-Hi with Popularity Thresholds of 25 and 50 in terms of Average Transmission Delay, Average Communication Delay of Successful Requests, Average Number of Control Messages of Successful Requests, Average Number of Traversed Nodes of Successful Requests, Average Number of Hops of Successful Requests and Blocking Ratio. Moreover, RED-Hi with Popularity Thresholds 50 demonstrates the worst performance. However, Figures 7.7 to 7.10 show that RED-Hi with Popularity

Threshold 25 attains better Load Distribution then RED-Hi with Popularity Thresholds of 5 and 50.

7.3 Comparison of RED-Hi and Pure Hierarchy

In this section, we compare the performances of RED-Hi and Non RED-Hi (Pure Hierarchy) in terms of Blocking Ratio and Load Distribution.

7.3.1 Blocking Ratio

In earlier graphs, we saw how different levels of popularity provided load balancing and scalability at different arrival rates. Figure 7.11 depicts a comparison of RED-Hi and Pure Hierarchy in terms of Blocking Ratio.

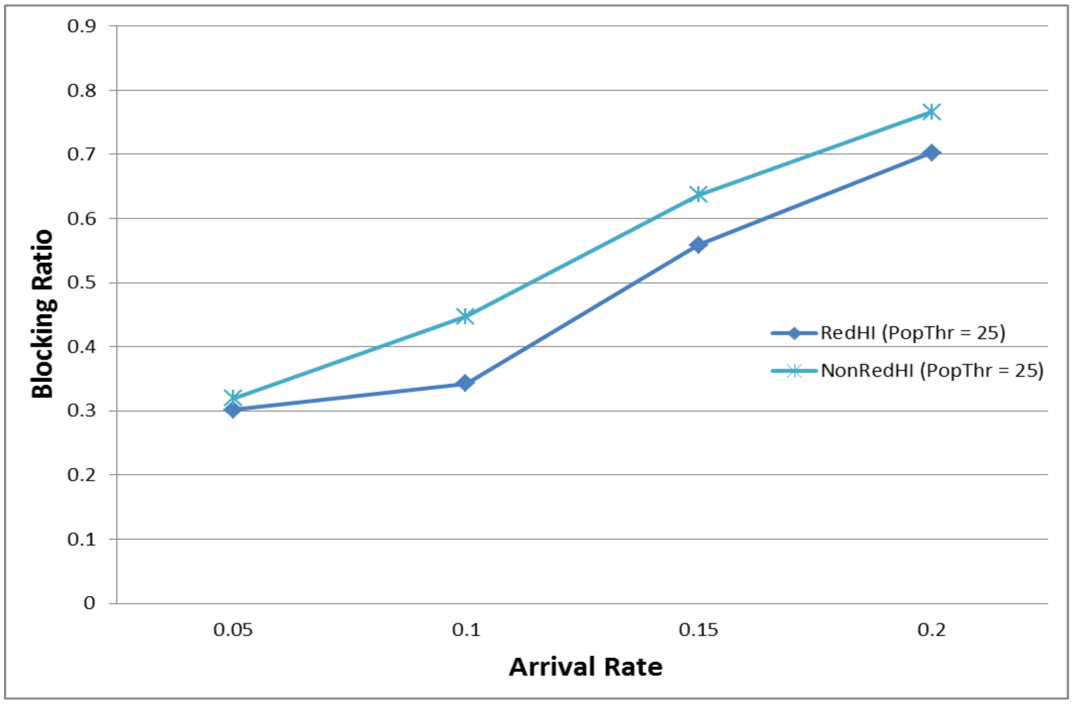

Figure 7.11: BR versus interarrival time for RED-Hi and Pure Hierarchy with Popularity Threshold of 25.

In Figure 7.11 we see that compared to a Pure Hierarchy, our proposed system works better under all arrival rates. This is achieved by the availability of the multiple paths to the same multimedia object, i.e. if one path fails object transmission will continue through an alternate path. Pure Hierarchy, however, lacks the same flexibility, hence it achieves significantly higher Blocking Ratio comparing to our proposed system.

7.3.2 Load Distribution

Figures 7.12 to 7.15 show that RED-Hi achieves better Load Distribution then Pure Hierarchy under various arrival rates.

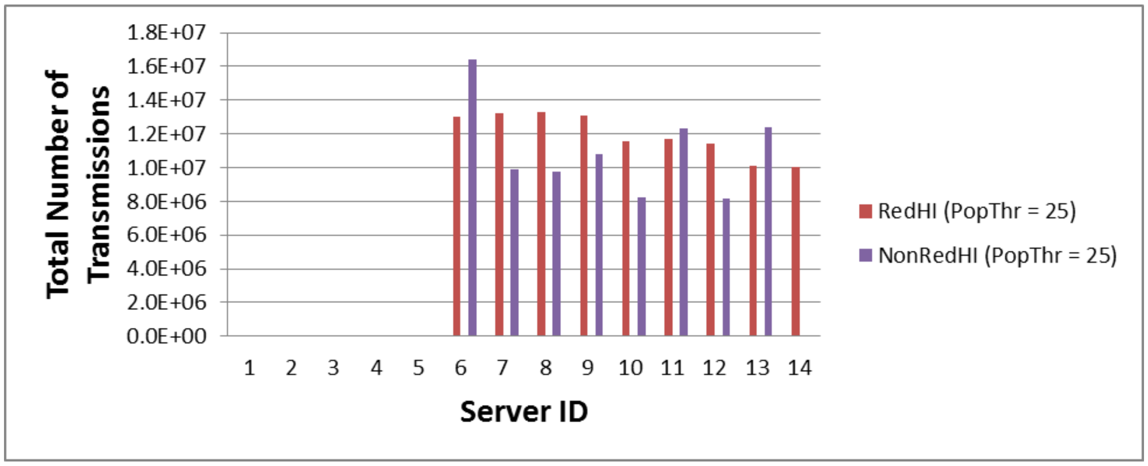

Figure 7.12: Server ID versus Total Number of Transmissions for RED-Hi and Pure Hierarchy with arrival rate 0.05 and Popularity Threshold of 25.

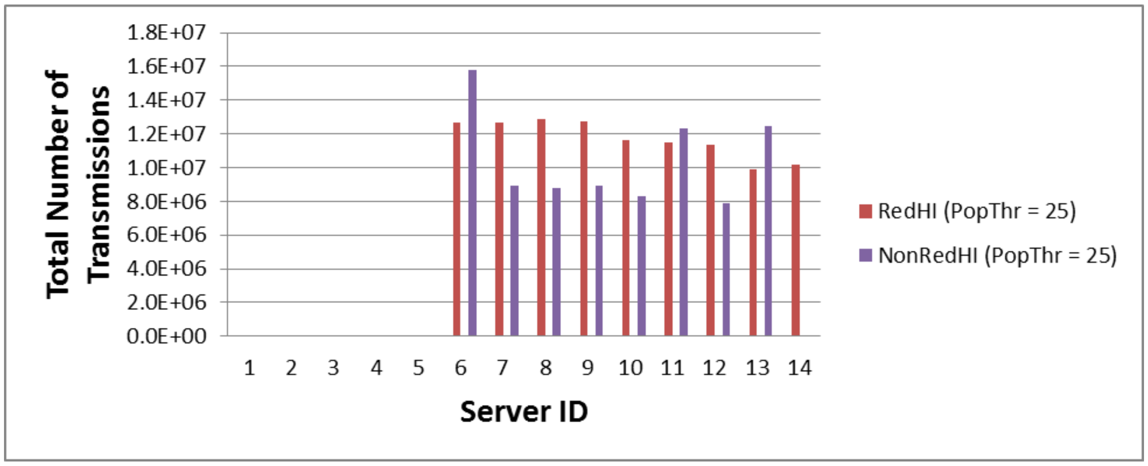

Figure 7.13: Server ID versus Total Number of Transmissions for RED-Hi and Pure Hierarchy with arrival rate 0.1 and Popularity Threshold of 25.

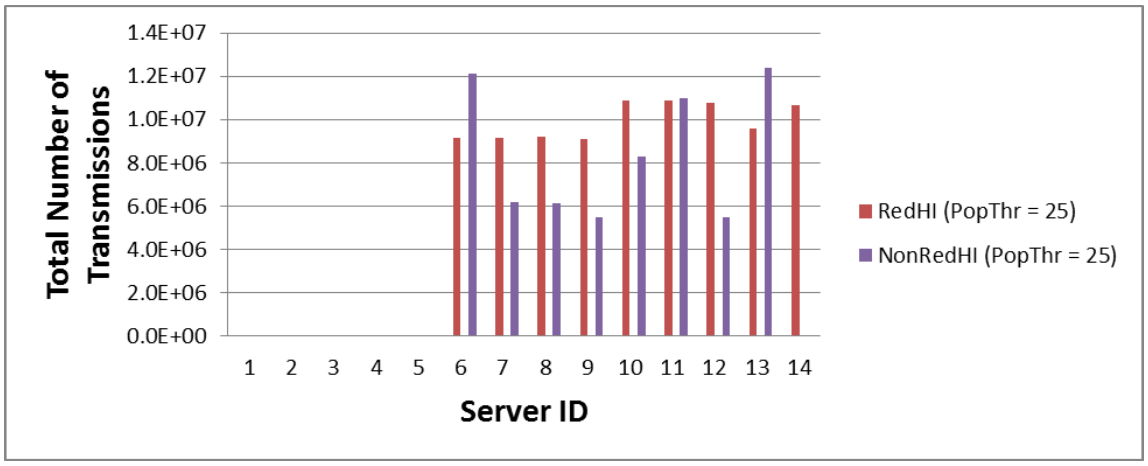

Figure 7.14: Server ID versus Total Number of Transmissions for RED-Hi and Pure Hierarchy with arrival rate 0.15 and Popularity Threshold of 25.

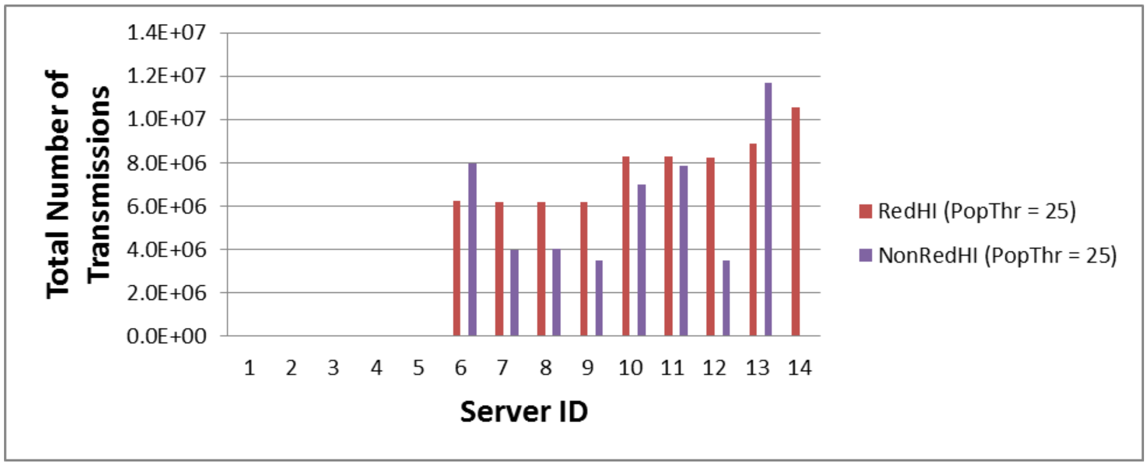

Figure 7.15: Server ID versus Total Number of Transmissions for RED-Hi and Pure Hierarchy with arrival rate 0.2 and Popularity Threshold of 25.

By examining the figures 7.12 to 7.15, it can be seen that for RED-Hi the Total Number of Transmissions are fairy distributed among the servers of the same level. For example, servers 6, 7, 8 and 9 have similar number of Total Number of Transmissions; the situation is same for the servers of all the layers in the system. However, Pure Hierarchy does not provide fair distribution of Total Number of Transmissions for the servers of the same level. For instance, servers 6, 7, 8 and 9 do not achieve just distribution of Total Number of Transmissions; the situation is same for the servers of all the layers in the system. We can conclude that RED-Hi achieves better Load Distribution then Pure Hierarchy under various arrival rates. It should be noted that in these graphs there is no node to compare with node 14 of Pure Hierarchy as Pure Hierarchy is single rooted and has highest node 13 in these simulations.

7.4 Cost Functions

Cost functions are important parts of the object locating and retrieval heuristics. The three cost functions i.e *RatioBW*, *UserBW* and *FreeBW* mentioned earlier in this thesis are compared. The results graphed in Figure 7.16 show that *RatioBW* and *UserBW* consistently outperform *FreeBW*. For example, when load = 200%, *RatioBW* rejected 21% and 80% less requests compared to *UserBW* and *FreeBW* respectively. As mentioned earlier, *FreeBW* has a tendency to retrieve objects from the higher levels of the hierarchy. Therefore, more links become occupied to retrieve these objects resulting in bad load balancing.

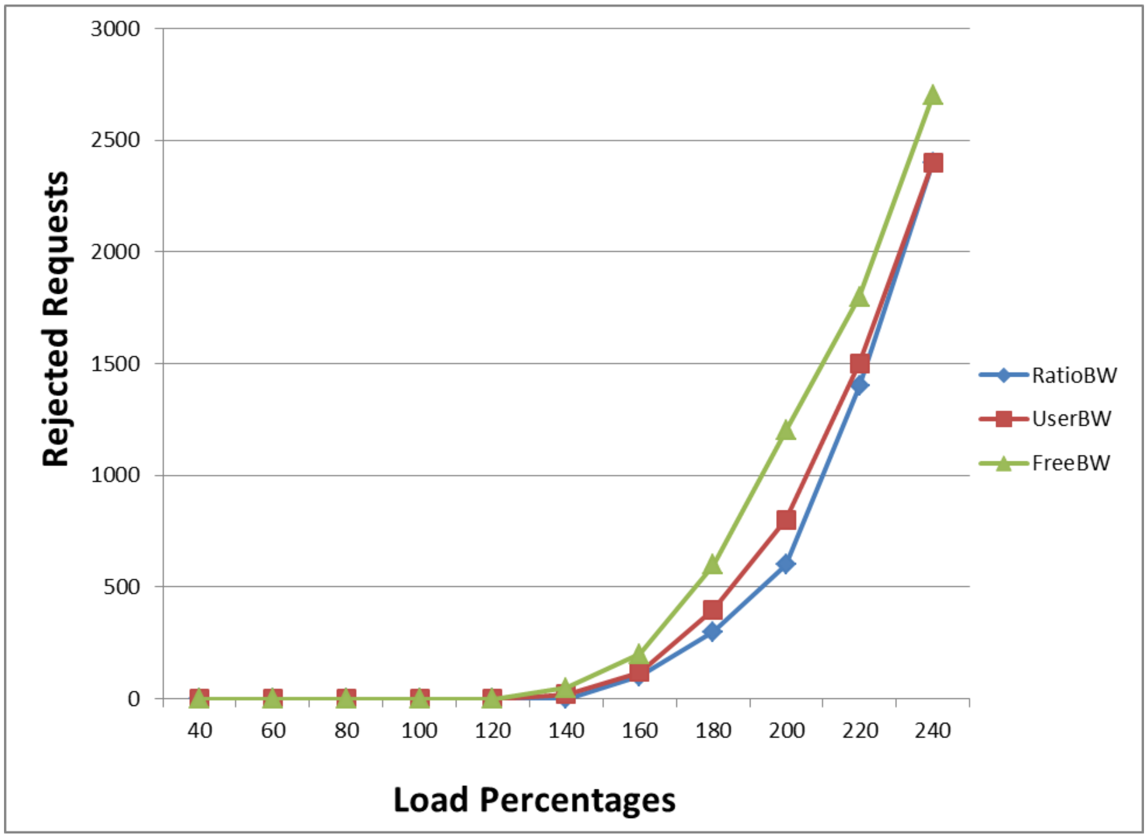

Fig. 7.16 The three cost functions *FreeBW*, *UserBW* and *RatioBW* compared

Chapter 8

CONCLUSION

In this thesis, a load balancing and inherently fault tolerant policy for streaming multimedia objects in a distributed environment is used that proposes a popularity threshold based dynamic content placement and delivery system. The proposed system uses the RED-Hi topology with a two-parent policy per node as its main property.

The main contribution of our work is introducing a popularity threshold based dynamic object placement for a distributed continuous media system using a hierarchical topology. The popularity threshold dynamically reduces the hops between the client and the source of a popular media content requested. Along with dynamic placement of the media content, another task sustained by the proposed system is balancing the load on the content provision system while making sure the content is delivered in a fault tolerant way. The proposed architecture keeps track of multiple routes to a requested object throughout the streaming process. This is done to make sure an alternative route is available in case of a link or node failure. Having an alternate route, using alternate links and nodes, results in seamless service provision from the clients prospective whenever system faces a node/link failure. It is shown that this system dynamically places multimedia objects in the distributed multimedia server system based on the popularity of that particular multimedia object. When storage space demands removal of some multimedia objects on a server, the least popular object is removed. The removed object is always available at a higher level in the hierarchy. Flexibility is achieved as there are multiple paths to a multimedia object. This not only facilitates near access to a resource but also gives a chance to flexibly select the most suitable node for a streaming session.

The hierarchical topology used in this thesis is so-called RED-Hi topology. The authors [9] of RED-Hi have extensively deliberated on the fault tolerant nature of RED-Hi architecture when used for service provision in distributed continuous media systems. They proposed RED-Hi for distributed continuous media servers and showed that RED-Hi has a better fault tolerant capability. They prove that the task of locating an object and retrieving it, in a distributed system, can be performed in a fault tolerant manner using RED-Hi by eliminating the possibility of bottlenecks in the system as compared to that for a pure hierarchy. To the best of our knowledge this [9] is the only other research work that uses RED-Hi in any academic research up till this writing.

The popularity threshold based dynamic content management and delivery system proposed in this thesis is at least as scalable as any other pure hierarchy based distributed content multimedia streaming server system. The rationale behind this is: the only difference in the two systems is that of redundancy in links, not in bandwidth. This has been discussed in detail in earlier sections. The cost analysis section shows that our system uses the bandwidth while improving the resource utilization and because of this, the extra link is an advantage of this system.

The performance of the system is shown in the Simulation results section of this thesis. It is evident from the results that the system is highly scalable, resiliently reliable and fault tolerant, furthermore it provides a superior delivery mechanism compared to the traditional pure hierarchy approach used in most DMS systems simply by better allocating the resources available. The results use a maximum of 0.2 arrival rate. This means that a new client arrives in our system every five seconds. The simulation results show that with an arrival rate of 0.2 sustained over a long period of time generates more than two million clients in the system that are served at a given time. This makes the system saturated. Having a higher arrival rate would cause same saturation when the number of clients in the system reaches the saturation point of two million requests. As such, the arrival rate alone does not affect the simulation results. It is the total number of requests being served at a particular instance of time that affects the results generated. It is shown in our results that our proposed system services up to two million clients in a fault tolerant, load balancing manner with only 14 servers in the system.

In the future, this work can be extended by providing an analytical model for this system. Along with an analytical model, a detailed study on the utilization of all the resources along the path can be researched looking at the costs incurred due to the redundancy in the system. This research provides an admission control mechanism that simply looks at the available resources for a node and the load on the links before granting admission to a request, future work can also look at different admission control protocols and their effect on the system performance.

REFERENCES

[1] Hsu, Y. H. T. J. K., & Huang, Y. E. W. W. F. (2011). Distributed Multimedia Content Processing in ONVIF Surveillance System.

[2] Gramatikov, S., Jaureguizar, F., Cabrera, J., & García, N. (2011, June). Content delivery system for optimal vod streaming. In *Telecommunications (ConTEL), Proceedings of the 2011 11th International Conference on* (pp. 487-494). IEEE.

[3] Jin, X. (2012, January). A Scalable Distributed Multimedia Service Management Architecture Using XMPP. In *Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science* (pp. 139-145). Springer Berlin Heidelberg.

[4] Song, F. F., Gao, W. L., Zhang, G. H., Gao, D. W., & Jiang, H. B. (2012). A P2P Based Video on Demand System for Embedded Linux. *Procedia Engineering*,*29*, 3070-3074.

[5] Tan, B., & Massoulié, L. (2010, July). Brief announcement: adaptive content placement for peer-to-peer video-on-demand systems. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*(pp. 293-294). ACM.

[6] Applegate D, Archer A, Gopalakrishnan V, Lee S, Ramakrishnan KK. Optimal content placement for a large-scale VoD system. In: Proc. of the 6th International ACMConference on emerging Networking EXperiments and Technologies (CoNEXT), December 2010; Philadelphia, USA.

[7] Borst, S., Gupta, V., & Walid, A. (2010, March). Distributed caching algorithms for content distribution networks. In *INFOCOM, 2010 Proceedings IEEE* (pp. 1-9). IEEE.

[8] Zhang, S., Shao, Z., & Chen, M. (2010, October). Optimal distributed p2p streaming under node degree bounds. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on* (pp. 253-262). IEEE.

[9] Shahabi, C., Alshayeji, M. H., & Wang, S. (1997, January). A redundant hierarchical structure for a distributed continuous media server. In *Interactive Distributed Multimedia Systems and Telecommunication Services* (pp. 51-64). Springer Berlin Heidelberg.

[10] Bhattacharyya, S., Diot, C., Giuliano, L., Rockell, R., Meylor, J., Meyer, D., ... & Haberman, B. (2003). *An overview of source-specific multicast (SSM)*. RFC 3569, July.

[11] Chawathe, Y., McCanne, S., & Brewer, E. (2000). An architecture for internet content distribution as an infrastructure service. *Unpublished work, February*. http://yatin.chawathe.com/~yatin/papers/scattercast.ps.

[12] Chu, Y. H., Rao, S. G., Seshan, S., & Zhang, H. (2002). A case for end system multicast. *Selected Areas in Communications, IEEE Journal on*, *20*(8), 1456-1471.

[13] Jannotti, J., Gifford, D. K., Johnson, K. L., & Kaashoek, M. F. (2000, October). Overcast: reliable multicasting with on overlay network. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4* (pp. 14-14). USENIX Association.

[14] Pendarakis, D. E., Shi, S., Verma, D. C., & Waldvogel, M. (2001, March). ALMI: An Application Level Multicast Infrastructure. In *USITS* (Vol. 1, pp. 5-5).

[15] Zhuang, S. Q., Zhao, B. Y., Joseph, A. D., Katz, R. H., & Kubiatowicz, J. D. (2001, January). Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video* (pp. 11-20). ACM.

[16] Ratnasamy, S., Handley, M., Karp, R., & Shenker, S. (2002). Topologically-aware overlay construction and server selection. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 3, pp. 1190-1199). IEEE.

[17] Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., & Khuller, S. (2003, March). Construction of an efficient overlay multicast infrastructure for real-time applications. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies* (Vol. 2, pp. 1521-1531). IEEE.

[18] Young, A., Chen, J., Ma, Z., Krishnamurthy, A., Peterson, L., & Wang, R. Y. (2004, March). Overlay mesh construction using interleaved spanning trees. In*INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (Vol. 1). IEEE.

[19] Microsoft Windows Media 9 series, July 2004. http://www.microsoft.com/windows/windowsmedia/default.aspx.

[20] Bernier, P. (2003, January). Your video wish is our command. *Xchange Magazine*, http://www.xchangemag.com/articles/311coverstory.html.

[21] Shaw Video-on-demand. (2004, July). https://secure.shaw.ca/sod/home.asp.

[22] Sen, S., Rexford, J., & Towsley, D. (1999, March). Proxy prefix caching for multimedia streams. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 3, pp. 1310-1319). IEEE.

[23] Almeida, J. M., Eager, D. L., Ferris, M., & Vernon, M. K. (2002). Provisioning content distribution networks for streaming media. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 3, pp. 1746-1755). IEEE.

[24] Akamai. (2004, July). http://www.akamai.com.

[25] Digital Island. (2004, July). http://www.sandpiper.net.

[26] Qiu, L., Padmanabhan, V. N., & Voelker, G. M. (2001). On the placement of web server replicas. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 3, pp. 1587-1596). IEEE.

[27] Radoslavov, P., Govindan, R., & Estrin, D. (2002). Topology-informed internet replica placement. *Computer Communications*, *25*(4), 384-392.

[28] Kangasharju, J., Roberts, J., & Ross, K. W. (2002). Object replication strategies in content distribution networks. *Computer Communications*, *25*(4), 376-383.

[29] Karlsson, M., & Mahalingam, M. (2002, August). Do we need replica placement algorithms in content delivery networks. In *7th international workshop on web content caching and distribution (WCW)*.

[30] Wang, L., Pai, V., & Peterson, L. (2002). The effectiveness of request redirection on CDN robustness. *ACM SIGOPS Operating Systems Review,36*(SI), 345-360.

[31] Ranjan, S., Karrer, R., & Knightly, E. (2004, March). Wide area redirection of dynamic content by Internet data centers. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*(Vol. 2, pp. 816-826). IEEE.

[32] Tang, X., & Xu, J. (2004, March). On replica placement for QoS-aware content distribution. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (Vol. 2, pp. 806-815). IEEE.

[33] Golubchik, L., Lui, J., & Muntz, R. (1995). *Reducing I/O demand in video-on-demand storage servers* (Vol. 23, No. 1, pp. 25-36). ACM.

[34] Aggarwal, C., Wolf, J., & Yu, P. S. (1996). *On optimal piggyback merging policies for video-on-demand systems* (Vol. 24, No. 1, pp. 200-209). ACM.

[35] Carter, S. W., & Long, D. D. (1997, September). Improving video-on-demand server efficiency through stream tapping. In *Computer Communications and Networks, 1997. Proceedings., Sixth International Conference on* (pp. 200-207). IEEE.

[36] Hua, K. A., Cai, Y., & Sheu, S. (1998, September). Patching: a multicast technique for true video-on-demand services. In *Proceedings of the sixth ACM international conference on Multimedia* (pp. 191-200). ACM.

[37] Gai, Ying. "Optimizing patching performance." (1999).

[38] Gao, L., & Towsley, D. (1999, July). Supplying instantaneous video-on-demand services using controlled multicast. In *Multimedia Computing and Systems, 1999. IEEE International Conference on* (Vol. 2, pp. 117-121). IEEE.

[39] Sen, S., Gao, L., Rexford, J., & Towsley, D. (1999, June). Optimal patching schemes for efficient multimedia streaming. In *Proceedings of 9th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99)*.

[40] Eager, D., Vernon, M., & Zahorjan, J. (1999, October). Optimal and efficient merging schedules for video-on-demand servers. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)* (pp. 199-202). ACM.

[41] Eager, D. L., Vernon, M. K., & Zahorjan, J. (2000, January). Bandwidth skimming: A technique for cost-effective video-ondemand. In *Proc. IS&T/SPIE Conf. On Multimedia Computing and Networking 2000 (MMCN 2000)* (pp. 206-215).

[42] Bar-Noy, A., & Ladner, R. E. (2004). Efficient algorithms for optimal stream merging for media-on-demand. *SIAM Journal on Computing*, *33*(5), 1011-1034.

[43] Eager, D., Vernon, M., & Zahorjan, J. (2001). Minimizing bandwidth requirements for on-demand data delivery. *Knowledge and Data Engineering, IEEE Transactions on*, *13*(5), 742-757.

[44] Ke, W., Basu, P., & Little, T. D. (2001, December). Time-domain modeling of batching under user interaction and dynamic adaptive piggybacking schemes. In *Electronic Imaging 2002* (pp. 130-141). International Society for Optics and Photonics.

[45] Coffman Jr, E. G., Jelenković, P., & Momčilović, P. (2002). The dyadic stream merging algorithm. *Journal of Algorithms*, *43*(1), 120-137.

[46] Bar-Noy, A., Goshi, J., Ladner, R. E., & Tam, K. (2004). Comparison of stream merging algorithms for media-on-demand. *Multimedia Systems*, *9*(5), 411-423.

[47] Aggarwal, C. C., Wolf, J. L., & Yu, P. S. (1996, June). A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *Multimedia Computing and Systems, 1996., Proceedings of the Third IEEE International Conference on* (pp. 118-126). IEEE.

[48] Viswanathan, S., & Imielinski, T. (1996). Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia systems*, *4*(4), 197-208.

[49] Juhn, L. S., & Tseng, L. M. (1997). Harmonic broadcasting for video-on-demand service. *Broadcasting, IEEE Transactions on*, *43*(3), 268-271.

[50] Hua, K. A., & Sheu, S. (1997, October). Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. In *ACM SIGCOMM Computer Communication Review* (Vol. 27, No. 4, pp. 89-100). ACM.

[51] Gao, L., Kurose, J., & Towsley, D. (2002). Efficient schemes for broadcasting popular videos. *Multimedia Systems*, *8*(4), 284-294.

[52] Pâris, J. F., Carter, S. W., & Long, D. D. (1998, July). Efficient broadcasting protocols for video on demand. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1998. Proceedings. Sixth International Symposium on* (pp. 127-132). IEEE.

[53] Eager, D. L., & Vernon, M. K. (1998). Dynamic skyscraper broadcasts for video-on-demand. In *Advances in Multimedia Information Systems* (pp. 18-32). Springer Berlin Heidelberg.

[54] Pâris, J. F., Carter, S. W., & Long, D. E. (1998, October). A low bandwidth broadcasting protocol for video on demand. In *Computer Communications and*

Networks, 1998. Proceedings. 7th International Conference on (pp. 690-697). IEEE.

[55] Pâris, J. F., Carter, S. W., & Long, D. D. (1999, January). A hybrid broadcasting protocol for video on demand. In *Proc. 1999 Multimedia Computing and Networking Conference* (pp. 317-326).

[56] Hu, A., Nikolaidis, I., & Van Beek, P. (1999). On the design of efficient video-on-demand broadcast schedules. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999. Proceedings. 7th International Symposium on* (pp. 262-269). IEEE.

[57] Hu, A. (2001). Video-on-demand broadcasting protocols: A comprehensive study. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 1, pp. 508-517). IEEE.

[58] Mahanti, A., Eager, D. L., Vernon, M. K., & Sundaram-Stukel, D. (2001, August). Scalable on-demand media streaming with packet loss recovery. In*ACM SIGCOMM Computer Communication Review* (Vol. 31, No. 4, pp. 97-108). ACM.

[59] Dan, A., Sitaram, D., & Shahabuddin, P. (1996). Dynamic batching policies for an on-demand video server. *Multimedia systems*, *4*(3), 112-121.

[60] Makaroff, D., Neufeld, G., & Hutchinson, N. (2001). Design and implementation of a VBR continuous media file server. *Software Engineering, IEEE Transactions on*, *27*(1), 13-28.

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

[62] Wang, Y., Zhang, Z. L., Du, D. H. C., & Su, D. (1998, April). A network-conscious approach to end-to-end video delivery over wide area networks using proxy servers. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 2, pp. 660-667). IEEE.

[63] McManus, J. M., & Ross, K. W. (1996). Video-on-demand over ATM: Constant-rate transmission and transport. *Selected Areas in Communications, IEEE Journal on*, *14*(6), 1087-1098.

[64] Feng, W. C. (1996). *Video-on-demand services: Efficient transportation and decompression of variable bit rate video* (Doctoral dissertation, The University of Michigan).

[65] Wong, W. M. R., & Muntz, R. R. (2001). Providing guaranteed quality of service for interactive visualization applications. *PERFORMANCE EVALUATION REVIEW*, *28*(1; SPI), 104-105.

[66] Gao, L., Kurose, J., & Towsley, D. (2002). Efficient schemes for broadcasting popular videos. *Multimedia Systems*, *8*(4), 284-294.

[67] Birk, Y., & Mondri, R. (1999, July). Tailored transmissions for efficient near-video-on-demand service. In *Multimedia Computing and Systems, 1999. IEEE International Conference on* (Vol. 1, pp. 226-231). IEEE.

[68] Sen, S., Gao, L., & Towsley, D. (2001, April). Frame-based periodic broadcast and fundamental resource tradeoffs. In *Performance, Computing, and Communications, 2001. IEEE International Conference on.* (pp. 77-83). IEEE.

[69] Bulterman, D., Grassel, G., Jansen, J., Koivisto, A., Layaïda, N., Michel, T., ... & Zucker, D. (2005). Synchronized multimedia integration language (smil 2.1).*W3C Recommendation, 13*.

[70] Sato, J., Hashimoto, K., Katsumoto, M., & Shibata, Y. (1999). Performance evaluation of media synchronization for multimedia presentation. In *Parallel Processing, 1999. Proceedings. 1999 International Workshops on* (pp. 608-613). IEEE.

[71] http://en.wikipedia.org/wiki/PAL (last visited 10-12-2013)

[72] http://en.wikipedia.org/wiki/NTSC (last visited 10-12-2013)

[73] Claypool, M., & Tanner, J. (1999, October). The effects of jitter on the perceptual quality of video. In *Proceedings of the seventh ACM international conference on Multimedia (Part 2)* (pp. 115-118). ACM.

[74] Ghiasi, S., Huang, P. K., & Jafari, R. (2006). Probabilistic delay budget assignment for synthesis of soft real-time applications. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, *14*(8), 843-853.

[75] Santos, R. M., Santos, J., & Orozco, J. (2000). Scheduling heterogeneous multimedia servers: different QoS for hard, soft and non real-time clients. In*Real-Time Systems, 2000. Euromicro RTS 2000. 12th Euromicro Conference on* (pp. 247-253). IEEE.

[76] Goode, B. (2002). Voice over internet protocol (VoIP). *Proceedings of the IEEE*, *90*(9), 1495-1517.

[77] Janssen, J., De Vleeschauwer, D., & Petit, G. H. (2000, April). Delay and distortion bounds for packetized voice calls of traditional PSTN quality. In*Proceedings of the 1st IP Telephony workshop (IPTEL 2000)* (pp. 105-110).

[78] Krasic, C., Li, K., & Walpole, J. (2001). The case for streaming multimedia with TCP. In *Interactive Distributed Multimedia Systems* (pp. 213-218). Springer Berlin Heidelberg.

[79] Pittet, A. (1996, June). Performance Issues in CD-ROM based Storage Systems for Multimedia. In *Proceedings of the 1996 International Conference on Multimedia Computing and Systems* (p. 0259). IEEE Computer Society.

[80] Ebrahimi, T., & Kunt, M. (1998). Visual data compression for multimedia applications. *Proceedings of the IEEE*, *86*(6), 1109-1125.

[81] Reusens, E., Ebrahimi, T., Le Buhan, C., Castagno, R., Vaerman, V., Piron, L., & Kunt, M. (1997). Dynamic approach to visual data compression.*Circuits and Systems for Video Technology, IEEE Transactions on*, *7*(1), 197-211.

[82] Milward, M., Nunez, J. L., & Mulvaney, D. (2004). Design and implementation of a lossless parallel high-speed data compression system. *Parallel and Distributed Systems, IEEE Transactions on*, *15*(6), 481-490.

[83] Ozden, B., Rastogi, R., & Silberschatz, A. (1995, May). A framework for the storage and retrieval of continuous media data. In *Multimedia Computing and Systems, 1995., Proceedings of the International Conference on* (pp. 2-13). IEEE.

[84] Kurose, J. (1993). Open issues and challenges in providing quality of service guarantees in high-speed networks. *ACM SIGCOMM Computer Communication Review*, *23*(1), 6-15.

[85] Chen, S., & Nahrstedt, K. (1998). An overview of quality of service routing for next-generation high-speed networks: problems and solutions. *Network, IEEE*,*12*(6), 64-79.

[86] Nagarajan, R., & Kurose, J. F. (1992, May). On defining, computing and guaranteeing quality-of-service in high-speed networks. In *INFOCOM'92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE* (pp. 2016-2025). IEEE.

[87] Ghosh, D., Sarangan, V., & Acharya, R. (2001). Quality-of-service routing in IP networks. *Multimedia, IEEE Transactions on*, *3*(2), 200-208.

[88] Ghandeharizadeh, S., Zimmermann, R., Shi, W., Rejaie, R., Ierardi, D., & Li, T. W. (1997). Mitra: A scalable continuous media server. *Multimedia Tools and Applications*, *5*(1), 79-108.

[89] Li, K., & Shen, H. (2005). Coordinated enroute multimedia object caching in transcoding proxies for tree networks. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, *1*(3), 289-314.

[90] Ng, J. K. Y., Song, S., & Tang, B. (2002). A Computation Method for Providing Statistical Performance Guarantee to an ATM Switch. *Real-Time Systems*,*23*(3), 297-317.

[91] Wen, J., & Lu, X. (2002). The design of QoS guarantee network subsystem.*ACM SIGOPS Operating Systems Review*, *36*(1), 81-87.

[92] Menth, M., Milbrandt, J., & Kopf, S. (2004, June). Impact of routing and traffic distribution on the performance of network admission control. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on* (Vol. 2, pp. 883-890). IEEE.

[93] Xia, Z., Yen, I. L., & Li, P. (2003, December). A distributed admission control model for large-scale continuous media services. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE* (Vol. 7, pp. 4001-4005). IEEE.

[94] Papadimitriou, C. H., Ramanathan, S., & Rangan, P. V. (1994, May). Information caching for delivery of personalized video programs on home entertainment channels. In *Multimedia Computing and Systems, 1994., Proceedings of the International Conference on* (pp. 214-223). IEEE.

[95] Ramarao, R., & Ramamoorthy, V. (1991, June). Architectural design of on-demand video delivery systems: the spatio-temporal storage allocation problem. In *Communications, 1991. ICC'91, Conference Record. IEEE International Conference on* (pp. 506-510). IEEE.

[96] Ryoo, J. D., & Panwar, S. S. (1999). Algorithms for determining file distribution in networks with multimedia servers. In *Communications, 1999. ICC'99. 1999 IEEE International Conference on* (Vol. 2, pp. 875-879). IEEE.

[97] Lüling, R. (1999, May). Static and dynamic mapping of media assets on a network of distributed multimedia information servers. In *INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS* (Vol. 19, pp. 253-261). IEEE Computer Society Press.

[98] Kalpakis, K., Dasgupta, K., & Wolfson, O. (2001). Optimal placement of replicas in trees with read, write, and storage costs. *Parallel and Distributed Systems, IEEE Transactions on*, *12*(6), 628-637.

[99] Burch, H., & Cheswick, B. (1999). Mapping the internet. *Computer*, *32*(4), 97-98.

[100] Broido, A. (2001, July). Internet topology: Connectivity of IP graphs. In *ITCom 2001: International Symposium on the Convergence of IT and Communications*(pp. 172-187). International Society for Optics and Photonics.

[101] Smith, A. J. (1978). Bibliography on paging and related topics. *ACM SIGOPS Operating Systems Review*, *12*(4), 39-56.

[102] Shahabi, C., & Banaei-Kashani, F. (2002). Decentralized resource management for a distributed continuous media server. *Parallel and Distributed Systems, IEEE Transactions on*, *13*(7), 710-727.

[103] Kostin, A. E., & Savchenko, L. V. (1988). Modified E-nets of distributed information processing system performance analysis. *Automatic Control and Computer Sciences*, *22*(6), 26-33.

[104] Aybay, I., & Shah M. A., Load management in a distributed multimedia streaming environment using a fault tolerant hierarchical system. *Turkish Journal of Electrical Engineering & Computer Science*, Paper accepted waiting for publication.