Date: 04/11/2007

I, Bo Zhang,

hereby submit this work as part of the requirements for the degree of:

Master of Science

in:

Computer Engineering

It is entitled:

A Large Scale P2P Based Architecture for Video on Demand Services

This work and its defense approved by:

Chair:
Yiming Hu

Dharma Agrawal

Carla Purdy
A Large-Scale P2P Based Architecture for Video on Demand Service

A thesis submitted to the
Division of Research and Advanced Studies
Of the University of Cincinnati

in partial fulfillment of the
Requirements for the degree of

MASTER OF SCIENCE

in the Department of
Electrical and Computer Engineering and Computer Science
of the College of Engineering
2007

by

Bo Zhang

Committee Chair: Dr Yiming Hu
Abstract

Network video on demand service (VOD) is gaining vast popularity nowadays. The VOD service allows users to select and watch video contents over a network and provides VCR-like functionalities which give users the controls over the playback of media sessions. The VOD services are commonly provided in conjunction with a live streaming system. Some conventional architectures for VOD service are server-based or proxy-based which all users are connected to dedicated servers or server cluster in one hop distance. The video contents are delivered to end users directly from server in unicast pattern. However, such centralized communication pattern has obvious shortcomings. For instance, poor scalability due to the limited server’s bandwidth and computing capacities, high vulnerability to malicious attacks or system failures. In this thesis, we’re going to present a peer-to-peer based architecture for the video on demand service. The proposed VOD system can be implemented as part of a famous IPTV/P2PTV system, the PPLive. Our system provides benefits in the following aspects: (1) past video programs can be replayed on demand of the users. (2) VCR-like functions are supported like pause and rewind. (3) the peer to peer communication pattern makes the system highly scalable. (4) the system supports high throughput rate and has low service response time which guarantee high quality of service.
Contents

Introduction ............................................................................................................................. 8
  1.1 Overview and organization of the on demand streaming system .......... 9
  1.2 Variety of streaming services ................................................................. 10
  1.3 Motivation and overview ................................................................. 12
  1.4 Thesis organization ............................................................................ 14

Internet Media Distribution ................................................................................................. 15
  2.1 P2P multicast system ........................................................................ 15
    2.2.1 Tree-based multicast (TBM) ............................................................. 17
    2.2.2 Mesh-based multicast (MBM) ......................................................... 18
  2.3 Providing play-as-you-download streaming service ......................... 20
  2.4 TBM versus MBM ............................................................................... 22

Design Issues ....................................................................................................................... 25
  3.1 Dynamics in P2P network .................................................................. 25
    3.1.1 Behavior analysis on BitTorrent ...................................................... 26
    3.1.2 The impact of dynamics on P2P network ...................................... 27
  3.2 Scalability .............................................................................................. 30
  3.3 Data management in distributed streaming system ......................... 31

Chapter 4 ............................................................................................................................ 33

System Design ....................................................................................................................... 33
4.1 Overview

4.2 System design

4.2.1 Content distribution

4.2.2 Content download

4.2.5 Adding interactivity

4.2.6 Media playback

4.2.7 Content management

4.2.8 Other Design Issues

Simulation and Results

5.1 Introduction to GPS: a general peer to peer simulator

5.2 Simulation setup

5.2.1 Simulation parameters

5.3 Simulation design

5.3.1 Content distribution infrastructure

5.3.2 Distributed video on demand

5.3.3 Churn generator

5.4 Simulation results

5.4.1 Downloading rate

5.4.2 End-to-end traffic

5.4.3 Index table multicast

Conclusion and Future Work
List of Figures

2.1 Tree-Based Multicast Architecture.......................................................20
2.2 Structure of PPLive IPTV System ..........................................................22
3.1 A Schematic Daily Access Model for a VOD System.................................29
4.1 File Saving..................................................................................................38
4.2 Search in Chord.........................................................................................39
4.3 Search in Gnutella Network.......................................................................40
4.4 Content Location and Download...............................................................42
5.1 GPS Architecture......................................................................................51
5.2 Procedures of Single Piece Download.......................................................56
5.3(a) Download rate (KB/s) as a function of peer lifetime (sec) using
       tracker-supported
       search.....................................................................................................59
      (b) Download rate (KB/s) as a function of peer lifetime (sec) using
          random
          search..................................................................................................60
5.4(a) Variation of download rate (KB/s) over time using tracker-supported search
      scheme....................................................................................................60
     (b) Variation of download rate (KB/s) over time using random search
       scheme..................................................................................................61
List of Tables

2.1 Basic Interactive Multimedia Services..............................................12
4.1 Sample of Index Table......................................................................41
4.2 Storage Quota Calculation....................................................................48
5.1 Simulation setups..............................................................................53
5.2 The percentage of traffic contributed by server and the number of partners a VOD user has, with various peer lifetime (min).............................................62
5.3 Configuration for the IPTV system....................................................62
Chapter 1

Introduction

The video on demand service is a network service that allows users to select and watch video contents over a network. Nowadays, people don’t have to rent cassettes or disks from the video groceries. Instead, by connecting their TVs to the set-top-box with broad-band cable plugged, the ordering of media products become as easier as download and play at your homes or offices.

The amazing convenience and satisfactory service quality had encouraged a fast growth of the on demand streaming system both technically and commercially. A market report made by www.rogertowne.com (an independent consulting company) supplements this on a statistical basis. According to their work, the number of subscribers of “In Demand Network” (a VOD service provider) had grown from 8.9 millions to 14 millions in the year 2004, and this number will continue to grow to 31 millions in 2008. Moreover, by the second quarter of 2004, roughly 75% of the network cable systems in this country has already been installed some forms of VOD services reported by US cable operators, nearly 12 millions digital cable households have the abilities to order some forms of video contents provided by VOD service. In addition to the increasing service quality, the dramatic drop in the cost of deployment also contributes to its vast popularity.
Survey made by www.rogertowne.com also shows, in the early 1990s, the per-stream cost was upwards of $10,000. In 2004, this number had fallen to $100-130 or even less. This cost will continue to come down with the advances in technologies.

1.1 Overview and organization of the on demand streaming system

In the past few years, Technical advances in the network and multimedia fields had given birth to a whole new category of network service, the media streaming over the Internet. The main objective is to stream media contents to thousands of end users across the Internet without the need of pre-downloading.

Based on the freshness of contents, the media streaming service is further divided into two types. The streaming service is a live streaming if the contents are sourced by ongoing events, like the daily news broadcast or festival gala. Otherwise, if the contents already exist for users to order on demand, then it is an on demand streaming service. In this thesis, we mainly focus at the on demand streaming services. Several terminologies are called interchangeably in the following sections. 1) video on demand: on demand streaming. 2) video contents: media contents.

A typical on demand streaming system consists of three major parts: 1) the
media server provides media sources, manages media contents, and coordinate with end users when they’re participating in the media sessions. 2) a large amount of end users join the streaming system and receive media contents. The users can initiate controlling commands to the server on the media sessions they’re watching. The server will make interactive reactions to the commands. The responsiveness to the controlling commands is crucial to the users’ degrees of satisfactions. 3) the media contents are delivered from the server to end users via the content distribution network. The throughput rate and efficiency of resource utilization which the network can achieve have significant effects on the service quality observed by the end users.

As mentioned, the on demand streaming system provides a set of interactive functionalities, like pause, rewind and fast-forward. The users are capable of controlling the playback of the media sessions by initiating commands to the server. Upon receiving commands from users, the server makes instant response to each of them. The responsiveness of the media server to the requests is critical to the user's service satisfaction.

1.2 Variety of streaming services

We have classified the streaming service into live streaming and on demand streaming, while there are some intermediate service types. So the streaming service can be also distinguished by the levels of interactivity they provide.
(1) Broadcast: It provides a regular cable TV-like service. The users are simply passive participants receiving the pre-loaded programs from the media server. There is no interactions existed between the end user and server in the duration of the playback sessions.

(2) Pay-per-view: This is like the existing CATV PPV services that the users pay for some specific programs and download from the server.

(3) Quasi video on demand: The users are grouped based on their interests. Different types of programs are served in different groups. The users have some extents of control over the programs they wish to watch (by switching from one group to another), but the control is still limited.

(4) Near video on demand: It provides some extents of interactive functions like pause, rewind and fast-forwarding by transitioning across discrete time intervals which represent different playing positions. This is implemented by serving the same program in multiple virtual channels skewed in time.

(5) True video on demand: It provides the users a complete set of interactive functions, just like they’re using VCR machine.

For the broadcast mode, the users passively receive video contents from server without client-side control. The server broadcasts pre-loaded TV programs, but cares little about the users’ interests. In addition, since no interactive function is supported, broadcast mode might be the easiest to implement. The multicast mode is a variation of the broadcast mode which multiple channels (servers) are
established with each channel serving programs with one same theme. The multicast streaming improves the broadcast mode in allowing users to choose the channels they like. The pay-per-view mode and the Quasi VOD mode are suitable for the users whose requirements for interaction is limited. The near VOD divides one single channel into several virtual channels and put one video program multiple times on those virtual channels skewed in discrete time intervals, users perform pause, rewind and fast-forward by switching between different virtual channels. The true VOD requires a real-time, mutual communications between the user and server. However, interactivity is expensive since the server must be powerful enough to make instant reaction to each user’s request.

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies-on-Demand</td>
<td>Customers can select and play movies with full VCR capabilities.</td>
</tr>
<tr>
<td>Interactive Video Games</td>
<td>Customers can play downloadable computer games without having to buy a physical copy of the game.</td>
</tr>
<tr>
<td>Interactive News Television</td>
<td>Newscasts tailored to customer tastes with the ability to see more detail on selected stories. Interactive selection and retrieval.</td>
</tr>
<tr>
<td>Catalog Browsing</td>
<td>Customers examine and purchase commercial products.</td>
</tr>
<tr>
<td>Distance Learning</td>
<td>Customers subscribe to courses being taught at remote sites. Students tailor courses to individual preferences and time constraints.</td>
</tr>
<tr>
<td>Interactive Advertising</td>
<td>Customers respond to advertiser surveys and are rewarded with free services and samples.</td>
</tr>
</tbody>
</table>

Table 2.1: Basic Interactive Multimedia Services

1.3 Motivation and overview

The traditional implementations of on demand streaming service are mostly
server-based or proxy-based. The server-based system is easy to implement and manage. However, the centralized architecture makes them poorly scalable, less reliable, error-prone and locality-oblivious. The proxy-based system consists of a network of powerful server and proxies. The end machines download contents from closer proxies with the lowest overheads. The utilization of system resource increases. However, the proxy-based VOD system is still constrained in their capability in serving a large number of users simultaneously. The emergence of peer to peer networks provides solutions to the problems of scalability and reliability. The possibility of downloading from multiple sources concurrently significantly increases the data throughput and frees up the servers from being heavily loaded. This thesis proposes a prototype on demand streaming system based on the P2P network.

Our system automatically backup TV programs after their live streaming for future on demand ordering. The content source is provided by a BitTorrent-based IPTV/P2PTV system called PPLive. By doing so, the future contents access demands can be served from both the servers and peers. The programs are saved in the format of video file, and can be replayed to users on demand. The video files are stored and well-organized in a distributed file system formed by harvesting spare storage space from a large number of end systems. The server creates and maintains meta-files to keep the contents organized as they’re in local storage. Fast and efficient content download is supported.
Our system has the advantages in:

(1) All the operations regard to the contents distribution and download are performed in decentralized way. The users download contents from multiple sources in the same time. No dedicated server is needed for contents distribution which makes the system scales well.

(2) The system supports interactive functionalities to control the playback sessions, such as pause and rewind. These are done by requesting and downloading the part of a program corresponding to a specified playing point.

(3) The content location and downloading scheme used significantly reduces the search overheads which results in a high data throughput rate, while substantially reduce server’s workloads.

1.4 Thesis organization

The thesis is organized as follow. In chapter 2, we put some literatures on introducing the current state-of-the-art of Internet media distribution. In chapter 3, we review the characteristics of the P2P network and several design issues of our system, such as the system scalability, dynamics of the distributed system, resource availability, bandwidth constrain and meta-data management. In chapter 4, we describe the detailed design and implementation. The chapter 5 evaluates the performance of the system using simulation-based analysis. In the last chapter, we summarize the thesis and give a general view of the future VOD system.
Chapter 2

Internet Media Distribution

The media streaming system delivers media contents to users over a network. We consider two types of peer-to-peer based content delivery systems, namely, the tree-based and mesh-based multicast.

2.1 P2P multicast system

Both tree-based and mesh-based multicast belongs to the category of P2P overlay multicast. The overlay network is an extra network layer built over an existing network layer. It differs from the physical network in that the links of the network are virtually constructed. In another word, two nodes that are neighbors to each other in the overlay might not be physically connected. The peer-to-peer network is one of such overlay built upon the physic network.

The overlay multicast system consists of the central media server, a group of end systems, and a set of connections/edges between the server and end systems or end systems themselves. In a multicast system, the interconnected nodes and the edges between them form an overlay network.

We classify P2P multicast system into two basic types, the \textit{structured} and
unstructured. The terminology “structured” means the networked system is organized in fixed geometric structure such as chain or tree. Each node is mapped onto a specific position and can be located by traversing the structure. In an unstructured overlay system, there is no assumption about the placement of nodes.

The structured P2P multicast system has the following characteristics, 1) the structure of the system must be constantly maintained, poor connectivity of the system leads to the partitioned network and will cause interruption of the service. 2) data flow in the form of media streams which goes through a fixed path along a set of edges, from the source to each destination. 3) the role of data sender or receiver is designated to each node based on their positions. The upstream nodes which are close to the source are senders, while the downstream nodes which locate farther to the source are receivers.

Unlike the structured multicast system, nodes in unstructured network are loosely organized. 1) there is no information maintained regarding the system structure. Nodes are only aware of their immediate neighbors. 2) data flows with no fixed path, but is determined by local decisions on the fly. 3) the nodes can choose to be data sender or receiver based on their needs. 4) only the variations of global network condition or events, like massive network disconnections, or server failure can result in noticeable performance degradation. A good example of the unstructured P2P content deliver system is the BitTorrent.

In the following subsections, we’ll describe two paradigms for the P2P
multicast system, the tree-based multicast, and mesh-based multicast.

2.2.1 Tree-based multicast (TBM)

In the tree-based multicast, the client nodes are grouped based on their interests. The nodes in a group are organized in a structure of tree. The server which is the content source is placed on the root of the tree, and the end hosts which are contents receivers are set to be leaf or inner/non-leaf nodes. The number of children a node has is determined by its capabilities. Usually, the nodes which are powerful in bandwidth and computing resources are assigned with more children. After a tree is built, the flow of contents goes along the edges of the tree from the root to each lower-level node.

One of the famous tree-based multicasting systems is the Scribe, proposed in [7]. The Scribe is a large-scale event notification service implemented upon Pastry, a scalable, self-organizing peer to peer location and routing substrate. The Scribe system consists of a network of Pastry nodes installed with Scribe software. The nodes behave as defined by Pastry. Each node in the system is allowed to create its own event group with a topic and becomes the leader of the group. A tree is built with the topic owner as the root and client hosts as the non-leaf nodes. Other nodes which are interested in that topic can subscribe to the leader and become a member of the group (registration). After registration, any node which has events to publish sends a message containing the event to the root where the
message is to be publicly announced to all the registered members. The nodes can join a tree by sending a *SUBSCRIBE* request to the root. The request is routed through the underlying overlay network towards the root. Along the path which a request is routed through, the first node met on the way which is already in the tree accepts the node as its child. Thereafter, the new node is considered as a registered member of the group. It has the ability to exchange news with other members.

The Scribe is well known as an event notification infrastructure. It can be also applied to media contents distribution. We can construct a multicast tree with the content source as root, and the subscribers as regular nodes. The source server distributes media contents packet by packet along paths to all subscribing nodes.

### 2.2.2 Mesh-based multicast (MBM)

We’ve introduced the tree-based P2P multicast system. In this subsection, we’ll introduce the mesh-based multicast system constructed on the unstructured P2P network. We specify the architecture by giving a commercial example, *PPLive* [16], a well-known BitTorrent-based live streaming system which is currently one of the most popular streaming applications in China.
First of all, we form an overlay substrate consists of the source servers, track servers, and end users who are interested in the service. End systems join in the substrate by sending `SUBSCRIBE` commands. In the substrate, the source server plays a role of seed node as in the BitTorrent which owns complete sets of original files. They serve contents to users and can be viewed as virtual channel stations. A node joins a channel by registering to any of the track servers (subscribing). The tracker sends to the node a set of IDs of peers which currently stay in the channel. The node contacts the peers in the set and sends queries to download pieces which it doesn’t have. Also it uploads pieces to other nodes that are requested. The media contents are decomposed and transferred in the form of data segments. During a video session, the peer sets are updated periodically by track server to assure
there’re always pieces for download. Unlike the Bittorrent, received segments are kept in a temporary playing buffer instead of saved locally in permanent storage. After be played, the pieces are discarded and the buffer will be refilled with new segments. In the duration of a video session, all the segments in the file from the beginning to the end are downloaded and played to users.

Unlike the sequential content delivery in streaming using tree-based structure, the media contents are downloaded un-sequentially in mesh-based structure and needed to be sorted before playing. Obviously, for the time-sensitive streaming services, random content download is not suitable since it might yield a low throughput rate. We’ll put more literatures on solutions of this problem in the subsequent section.

The PPLive system is currently serving millions of users in China. To our knowledge, it is one of the most popular streaming services in China providing online TV programs lively to people who do not have access to Cable TV services.

2.3 Providing play-as-you-download streaming service

As mentioned, video on demand service must provide users with play-as-you-download experience which the programs can be played shortly after requested. In another word, the playback can be started without need to download the entire video file. However, the data segments are downloaded un-sequentially
in MBM, it is possible that the segments in the beginning part of a file are downloaded at the last, while segments in later part are downloaded in the beginning. The useful segments downloaded for currently part are far less than sufficient for the playback to be started, though the download rate is high.

A solution to this problem was proposed in [4]. In their work, files are divided into chunks. Each chunks contains a set of segments in sequential order. The server publishes the chunks one after another in sequence, while the segments within a chunk are allowed to be downloaded in random order which can achieves a high throughput rate. The playback of a program can be started only after a complete set of segments of the first chunk have been downloaded. This gives a download warm-up interval approximately equal to the time needed for
downloading one complete chunk. We’re using this approach in our simulation.

**2.4 TBM versus MBM**

The tree-based multicast and mesh-based multicast are both suitable for content delivery in the streaming applications. In this section, we make comparison of two and choose one to base our VOD system.

The tree structure has the following advantages: 1) it has a fixed structure for data dissemination, the nodes in each level push data packets they received from the upper level to the lower level nodes, the routing decision is simple and predictable. This is beneficial for minimizing both delay and jitter. 2) video files are disseminated in the form of stream along a set of edges from the root to child nodes. The source node release data packets in sequential order. Also, all member nodes always receive and forward packets sequentially as well. This ensures high throughput rate, and short playback warm-up time at the side of the users. This is because there is no need to wait for the collection of packets, and then sort them in right order. 3) The tree-based multicast adopts a “push-based” data dissemination pattern. The nodes stand on the upper level always push packets down to the child nodes. The downstream nodes do not need to send “REQ” message to request a packet which are overhead-effective.

The tree-based multicast system is organized in some fixed structures, so the content delivery and resource management are easier and more efficient. However,
it still has plenty of shortcomings. First, there is no guarantee that all the packets from the upper level could finally reach the lower level due to packet loss. If packet loss happens frequently in the levels close to the root, the nodes in lower levels would experience unacceptable playback quality. Also, duplicated packets might be generated while transferring because of the “push-based” scheme used. Second, the cost for structure maintenance is high. When an inner node leaves the tree, all its downstream nodes are disconnected causing the interruptions of service at those nodes. The multicast tree must be repaired shortly in order to make users oblivious to the interruptions. Third, for a tree with high nodes’ degree, the number of leaf nodes occupies a dominant portion of the total number of nodes. Basically, the leaf nodes make no contribution to data forwarding, while the inner nodes which are the minority take most of the forwarding loads. The unfair load distribution leads to an unreliable multicast infrastructure. The failures of a large amount of inner nodes close to the root destroy the multicast tree which is a disaster to the system.

Based on the above considerations and application requirements, we decide to use mesh-based P2P network as the content distribution platform in our application. The reasons are summarized as follows. First, since the nodes in mesh-based multicast structure positively request packets from their partners. If a packet gets lost on the way, a simple Re-request should be enough to grab the lost packet. Also, since packets are only transferred upon requests of other nodes,
duplicate packets are eliminated. Second, the flat organization of nodes removes the “hot spots” in the network, the data forwarding loads are evenly distributed among all the nodes which makes the system robust and scalable. Third, the unstructured topology eliminates the need for structure repairing. Other than the above considerations, we’ll see in section 4 that the mesh-based multicast infrastructure enhances the content search efficiency.
Chapter 3

Design Issues

Basically, the peer to peer applications are implemented on top of the overlay networks. The extra network layer is built upon the underlying backbone network to leverage the available resources owned by a large number of non-dedicated computers. The re-organized network is capable to support more complex works ran on top of it, while it no longer depends on the expensive centralized servers. However, since the composing units of a P2P application are distributed over wide area network, its performance is significantly influenced by the characteristics of the underlying network. In this chapter, we’ll make in-depth analysis on the top design issues of the P2P network, evaluate their impacts on our system and give solutions to the problems.

3.1 Dynamics in P2P network

The large-scale, distributed systems like peer to peer network are characterized as highly dynamic and instable. The nodes are given large extents of autonomy. They can join and leave the network at any time, request service at their will, or even behave maliciously. From a continuous, global view, the structure of the overlay
keeps changing over time, also the workload intensity fluctuates constantly.

The network dynamics adds great challenges to the object locating in P2P network. The frequent status change of nodes (active/inactive) makes the network topology varies all the time. There is no static “topology map” for guiding the network traffics flow. For example, in file sharing system, some of the search queries are routed through failed nodes which re-routings are needed, while other normal nodes are not taken advantage of. Additionally, higher nodes join/depart rate implies shorter duration of node’s application sessions. The consequence is the fluctuation of resource availability. Experimental studies on this phenomenon have been made on the well-known P2P-based file-sharing application, BitTorrent, and their analysis gives us an in-depth view over this problem.

3.1.1 Behavior analysis on BitTorrent

We choose the BitTorrent as example for analyzing dynamics in peer to peer network. The BitTorrent is a well-known file sharing application based on P2P network. For its vast popularity, we assume that the readers already had clear understanding of its structure and working mechanism.

In [9], the authors give a quantitative analysis of the dynamics and instability in BitTorrent network. The components in BitTorrent system are classified into three main parts. The HTTP server provides download of the “.torrent” file. The tracker server provides support to the users in peer discovery. The regular peers exchange
data with each other. The authors made observations to the behavior patterns of these three types of components over a multi-month period. The degree of dynamics is represented by the average duration of components’ up-time. They assumed a relation that the longer the up-time is, the higher dynamics showed in the network and the lower availability is. According to their experiment, the HTTP servers showed a relative low up-time, only half of the HTTP servers stood online for more than 2.1 days. The track servers showed a better availability, half of the track servers have an uptime over 1.5 day or more, one of them even showed a continuous uptime for more than 100 days. For regular nodes, very few showed a high availability, 9219 nodes out of 53883(17%) have a uptime longer than one hour after the downloads had been finished, only 1649 peers(3.1%) stood longer than 10 hours, and for 100 hours, the proportion even decreases to 0.34%.

3.1.2 The impact of dynamics on P2P network

Based on the above study, we know that the P2P network is highly dynamic due to the unpredictable pattern of nodes’ behaviors. There are various outcomes of this property, such as the variation of resource availability, and reduced system reliability.

(1) Variation of resource availability

The amount and distribution of available resources fluctuate over time due to the network dynamics. This makes the data management in distributed system a
difficult task. For example, the *Oceanstore* [12] is a famous storage system providing persistent data access over P2P network. High availability of data is the major concern of *Oceanstore*. However, since participating nodes might leave the system frequently, some data would become unavailable which violates the data persistency. An intuitive solution to this problem is to add redundancy to the system. Instead of storing only one copy of a document, multiple copies of a same data set are added. The reason is to keep providing backup replicas to users even when part of the replicas become unavailable. Replication and erasure coding [10] are two methods for adding redundancy. Replication makes multiple copies of one single document and places them in multiple nodes, so when a data request fails to be served at an inactive nodes, it can be immediately re-directed to other active

![Figure 3.1: A Schematic Daily Access Model for a VOD System](image-url)
nodes and be served there. The users won’t notice the unavailability of data as long as their requests are served within a certain time. Erasure coding is another technique to achieve high availability. Instead of storing each replica in its entirety, the original contents are encoded and split into multiple stripes. The stripes are placed in multiple locations over the network. The coding algorithm is designed to ensure that a complete document can be re-composed by gluing only a subset of the stripes. The advantage of erasure coding is the reduction in the amount of data stored. On the downside, the erasure coding requires more computing resource and time on coding/decoding because of the complex coding algorithms used. For simplicity, we use data replication to add redundancy in our simulation.

(2) Dynamic network topology

Another outcome of the network dynamics is the ever changing topology. Since the status of nodes changes frequently, the network topology and connectivity keep changing as well. A concrete instance of this would be the Gnutella network. A node in Gnutella network is only aware of a small set of nodes called neighbor node. It has no knowledge about the global state of the system. The nodes search for a file by making public announcements to all its neighbors, and each of the neighbors relays to pass this query to its neighbors. This process continues on each node received the query till the owner of the file is reached. In reality, a large amount of querying messages get lost on the way due to the poor network connectivity, the speed of query proliferation decreases substantially as a large
number of messages are failed to be delivered.

The dynamics in network topology also has a significant impact on the on demand streaming applications. This is reflected in the increase of search overhead. Since the data queries might be sent to the nodes which are not active, the senders must re-send the query for the same piece. Apparently, the frequent search failures and repeated re-request incur a substantial increase in search overhead. Our system mitigates the effect of dynamics by introducing support from the servers in content search. The server provides information on accurate data locating, thereby reduces the amount of failed searches.

3.2 Scalability

Scalability is another critical issue for the distributed applications. This is due to two facts. First, the revenue of a network service is largely determined by the scale of the service which is represented as the amount of customers that can be accommodated. Second, the scale of a networked system is upper-bounded by the amount of resources it owns for serving users. Therefore, the key to expand system scale is how to properly utilize network resources. Generally, the resources of computer network are classified into three basic types: computing power, bandwidth and storage, while other types of resources like printing, scanning and displaying are not considered, because they won’t directly affect performance. The CPU-bounded applications which perform a lot of computational works
concern more on computing capability, while the I/O bounded applications would have rigid requirements on bandwidth and storage for storing and transferring massive volume of data, such as the distributed data farm and media streaming applications. For the distributed streaming systems, the resource bottleneck is the bandwidth, not storage.

As mentioned, by simply relying on centralized data server to perform VOD multicast is infeasible, since no matter how powerful the server cluster could be, there is an upper bound in its capacity. We notice that the Bittorrent network is a good solution for serving a large number of users simultaneously in streaming service. By using the BT network, media contents are distributed among a large amount of nodes. When a node requests for video streaming, it download contents from both the media server and other participating nodes in the system. The BitTorrent network has two benefits as the platform of streaming applications. First, the centralized servers are no longer necessary. Spare upload capacity from multiple nodes can be pooled and used for transferring data. Second, nodes receive data simultaneously from multiple sources. The increased throughput rate allows the system to provide high-quality, larger size video programs.

3.3 Data management in distributed streaming system

In the distributed streaming system, media contents are distributed over the network. However, this brings up a new problem. That is how to manage the
distributed data contents as they’re stored in local file system. We set the following stipulations for defining a sophisticated data management scheme:

1) The content management unit must make response to any data search request in a time interval no longer than a threshold value. Because most of multimedia applications are time-sensitive.

2) The data management scheme must not incur noticeable increase of search overhead.

3) The data management unit must guarantee the above two points in face of fluctuating workloads and dynamic network topology.
Chapter 4

System Design

In the previous chapters, we have introduced the background knowledge on the on demand streaming services, the motivation of our work, and the major design issues. In this chapter, we make detailed description of the design and implementation of our distributed on demand streaming system.

The objective of our system is to provide on demand streaming service of TV programs after they have been lively streamed by the PPLive streaming network. The users who expect to watch previously released programs can be served based on their demands via this service. The system is comprised of three main units: 1) the recording unit runs on the server which controls the global contents distribution. 2) the content management unit manages data segments as one single file. This can be also installed on the server. (3) the content download unit run on client side which searches contents amongst and download from peers.

4.1 Overview

The media contents distribution is the primary issue of the on demand streaming system. There are two naïve approaches for doing this. First, the media server
holds all the contents in its local storage. When streaming demands are received, the server builds direct communication with the users. As mentioned, the bandwidth constrain prohibits the server from directly accommodating a large amount of users simultaneously. Second, the users cache the contents they need in the local storage during the live streaming so they can be replayed at anytime. However, the problem is that it is impossible to record if the users couldn’t get the program schedules or they don’t wish to spend their valuable disk space for storing media contents.

We propose a contents distribution mechanism which neither fully relying on the media server nor the individual user’s machine. The basic idea is for each node to voluntarily contribute some amount of their storage space for caching some segments of contents. The storage pool formed by aggregating spaces donated by the nodes contains at least one entire copy of each media file released. As a result, users must contact each of the participating nodes in order to retrieve the entire file. The retrieving process is supported by a centralized control unit on the server. In our design, the importance of media server fades since it doesn’t serve data directly, despite it is still responsible for centralized coordination and contents management.

Our system design has 4 advantages. 1) since the system records video programs while they’re lively streamed, the programs are available immediately. The users can initiate interactive request like pause, rewind by downloading the
data segments that have been recently distributed. 2) the playback warm-up time is short. The media session can start once the playing buffer has been filled up. 3) the decentralized content distribution pattern and management schemes allow the system to be scalable. 4) the tracker-supported search approach provides an accurate and efficient search of contents.

4.2 System design

Our VOD system is deployed in a distributed network environment. The nodes are regular PC machines which have been installed with our streaming plug-in. The track servers and media servers are owned and maintained by the service providers. We introduce the implementation from the following three aspects:

(1) content distribution. The server distributes the media contents amongst peers in the distributed BitTorrent-like network.

(2) content download. The participating peers search and download contents cached in the system with the assistance of the media server and track server.

(3) content management. The content management unit manages the media contents after they have been streamed.

4.2.1 Content distribution

In our on demand streaming system, the media contents are streamed and cached amongst peers. The participating peers cooperate to download contents from each
other. The media contents are distributed by the media server and cached among a group of nodes. Each of the nodes saves a certain number of data segments. The media server builds and maintains indices of the distributed contents for the sake of future download. When a streaming demand is received, the media server sends the index of the desired file to the user. The user’s machine contacts a set of nodes by looking up the received index and download data from them. This is the basic step for our system.

The content distribution is governed by the media server while the media programs are lively streamed. For each data segment released, the media server designates a node which is currently alive in the channel to cache it. The responsible node is selected using some random functions. The identification of the selected node and a saving request are attached to the body of the segment. The saving request asks the node to cache the segment. Each node checks the segments received to see if they need to cache them, if not, the request is ignored, otherwise, the segment is cached as requested and a positive acknowledgement is sent back to the server. In the cases of cache failure, such as un-notified nodes departures, or delivery errors, the server gets no response, it selects another node and re-sends the segment.

4.2.2 Content download

The main function of content download unit is searching data segments across the
network and downloading from the responsible nodes. We consider three common search schemes, flood-based search, overlay-based search, and the index-based search in this subsection.

![Figure 4.1: File Saving](image)

(1) **Overlay-based search**

The structured peer to peer systems like Chord [17] or Pastry [18], built up P2P overlays over the physic network. Each object in the network (node or file) is assigned with unique identification. The object is mapped to a specific location which equals to its hashed ID. Known the target’s identification, any object can be
located by traversing the network within several hops without any centralized indexing. The use of the overlay network and the index-free search scheme eliminates the need for keeping and continuously updating the global system state. On the downside, since given a large-scale distributed system, the search queries might be routed through multiple intermediate nodes before reaching the destination. This leads to a substantial routing latency. For the media streaming services which have restrict requirements on the service responsiveness, long routing delay is not expected.

![Figure 4.2: Search in Chord](image)

**(2) Search by flooding**

Flooding search queries to all or a subset of the nodes in the network can also return the desired data. Since the segments of a media file are distributed only to the nodes which have ever stayed in the channel during the live streaming, the
queries can be sent to only this group of nodes. The server can create a list recording all these nodes for each file, we call it the tracking list. Then, a data segment can be located by flooding the queries to each node in the list. The flood-based scheme doesn’t depend on any global state of the system either, and it is also easy to implement and operate. The shortcomings are obvious as well. First, making public announcement over a wide area network is resource-inefficient. Second, without supports from the central indexing, it is impossible to know which nodes in the tracking list have the right segment that is needed.

![Diagram of Search in Gnutella Network](image)

Figure 4.3: Search in Gnutella Network

(3) Index-based search

The easiest way to locate a file would be for the server to keep centralized index recording the location of each segment. The users enquiry the server for where the segment are cached. The server can look up the segments from the index upon
receiving the queries, and sends back the result to the users. Then, the users contact the resulted nodes for their desired data. The index for each file is built and maintained by the media server while it is being lively streamed. The index contains the mapping from each segment to its responsible peers. The indices are sent to each user. Once a user orders the VOD service of a program, it looks up the index and downloads from other peers.

Compared to the overlay-based search which incurs substantial search delay, and the flood-based search which requires significant waste of network bandwidth, the index-based search is accurate and bandwidth-efficient. In our system, we adopt the index-based search for locating the cached media contents.

### Table 4.1: Sample of Index Table

<table>
<thead>
<tr>
<th>Piece A</th>
<th>Piece B</th>
<th>Piece C</th>
<th>Piece D</th>
<th>Piece E</th>
<th>Piece F</th>
<th>Piece G</th>
<th>Piece H</th>
<th>Piece I</th>
<th>Piece J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 2</td>
<td>Node 6</td>
<td>Node 3</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### 4.2.3 Disseminating indices

The distributed on demand streaming system houses mass volume of media contents with size of multi-terabyte. The granulation and distribution of the contents derives millions of data segments which are scattered across the network.
The indices are created to track the data distributed by the server. Commonly, the size of the index table could be several megabytes. There might be thousands of users participated in one single hot program. Each of the users’ node must download and read the index table at least once which causes significant overheads on transferring the indices. So, we need to find an approach to disseminate the indices without substantially increasing the overheads.

We noticed that, it’s much more resource-efficient if we multicast the index instead of sending them individually to each user. Basically, the index files

![Diagram of Content Location and Download](image_url)
have no difference as the media files. We can use the existing BitTorrent-based content distribution platform which is used for delivering media contents to deliver the index file, so the search overheads are evenly shared among nodes.

The index file must be updated and re-published periodically since there are always new contents released and new users join. A critical issue is that how often the index files should be updated. If the updating interval is large, the new users must wait long time for the next update. On the contrary, if the index files are frequently updated, the bandwidth overheads increases as more re-publishing are performed. A compromising approach would be for the server to broadcast the index files of the popular programs (50%), while perform serve individually to users for the less popular ones (50%).

We classify the data flowed in the network into two types. 1) the media contents. 2) the auxiliary information and control messages which include the indexing files and other control messages. Both types of data are divided into segments with the same size and. Each segment is attached with a tag indicating what type the segment belongs to. The content distribution network delivers both types of segments to all the nodes. The nodes identify the segments which make up the index table by reading the tag. The segments are re-composed to index table.

Since the indices of the media contents are disseminated using the existing contents distribution network, the users perform index-based search locally. The servers are freed up from centralized index querying/answering. Also, the search
latency decreases substantially since the search queries traverse only one hop to reach the destination node for each segment. Additionally, the segments can be downloaded in sequential order which supports the play-as-you-download streaming pattern.

### 4.2.4 Tracker-supported search

As aforementioned, the dynamics in large-scale distributed system incurs significant search overhead. Though the index file informs the accurate location of the data segments, it doesn’t inform whether or not the piece is available. Considering the following scenario, piece i is stored in both node A and B, another node C wants to download piece i. In static environment, node C can easily get piece i by requesting from A or B. While in dynamic environment, if node C requests piece i from node A and node A is not active, then a search failure happens. So, node C has to request the piece for the second time from node B. Thus, two searches are needed to retrieve piece i. Frequent search failures has significant impact on the throughput rate and also the media playback rate. To deal with the problem, we incorporate support from the track server.

Similar to the BitTorrent, to begin download, the users must first contact and register to the track servers in order to join the system and discover peer nodes. Thereby, the track server is a central registry which keeps information of all the active nodes. The track servers can publish such information, thus each node knows whether or not a node is active. This is valuable for search in a highly
dynamic environment because the user nodes can avoid sending requests to inactive nodes once noticed, thus avoids effortless bandwidth consumption.

First of all, the track server creates a list of all the currently active nodes, called the “currently active nodes list”. The list is sent to the source sever, and be published in the same way as the index files or other auxiliary information. This list must also be updated and republished periodically. After receive the index files and “currently active nodes list”, the nodes find out to which nodes the search queries should be sent. Then, by checking the “currently active nodes list”, if the target nodes are active, queries are sent out immediately, if not, re-check the indexing table, and find out another target node. By doing so, with a high probability that the queries sent out could reach the right nodes. The possibility of search failure decreases, and the data throughput increases.

4.2.5 Adding interactivity

The interactivity is another important feature of the on demand streaming service. To control the media sessions, the users requests for operations like pause and rewind. The server must make instant response to each of the requests received in order to make users feel like they are receiving dedicated service. However, the cost of providing true VOD service is expensive due to the constrained servers’ outbound bandwidth.

We noticed that in our system, the programs are cached among nodes right
after the live streaming. The caching-while-playing scheme allows users to retrieve cached contents immediately. Thereby, users can pause/rewind and continue by suspending the media sessions and requesting the missing segments which has been cached when come back. The fast-forwarding is not supported, since our system receives contents from a live TV broadcast source. In contrast to the server-based scheme, the past parts of content are downloaded separately from multiple sources, the increased overhead is absorbed by all the engaged nodes instead of only the server. Additionally, the virtual data store acts a role of playing buffer, yet it is considered as having an infinite size of space compared to memory-allocated buffer.

4.2.6 Media playback

The content download is performed based on the metadata and the active nodes list. The user nodes looks up the index table for the locations of pieces. From the set of nodes listed by querying the metadata file, they find out the currently active nodes by querying the “currently active nodes list. Within the remaining nodes refined from the two steps of querying, the requests for data are sent only to the nodes which has the desired data and currently active. The pieces are requested in the consecutive order according to their segment number. As long as the downloaded segments fill up the buffer, the playing of the movie can be started. This process continues as the video programs keep going.
4.2.7 Content management

The size of media contents cached among peers is in a magnitude of multi-terabytes. The content management unit is responsible for managing the cached contents, like creation of metadata for new files, file classification, removing of outdated files.

When a program has been streamed, the media server accepts it as a new media file stored. It makes a new record for the file, includes the information like “start segment#” and “end segment#” of the file, creation date and time, file size and etc. Combined with the index table, a whole metadata file for the media file is created. At last, it is classified by its popularity (maybe defined by daily click-on rate) and inserted into metadata database. When life time of a file expires, the file management component automatically removes it and replace with new contents.

4.2.8 Other Design Issues

In this section, other design issues such as the storage quota allocation, storage redundancy are discussed.

(1) Storage quota allocation

As we mentioned, storing all the media contents in the server is infeasible. Instead, the contents are distributed to a large amount of nodes in the network. Before joinning the system, each node must contribute some part of the storage space. We calculate the size of the per node storage space as follow.
Table 4.2: Storage Quota Calculation

<table>
<thead>
<tr>
<th>Streaming Rate of Channels (KB/s)</th>
<th>Size of Piece (KB)</th>
<th>Lifetime of Media Files (day)</th>
<th>Service Period (hour)</th>
<th>Redundancy Rate</th>
<th>Number of Channels</th>
<th>Number of users</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = 70</td>
<td>L = 256</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>100</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

The estimated size for all the files in stock is $70\text{KB/s} \times 10(\text{days}) \times 8(\text{hrs}) \times 3600 \times 10(\text{red rate}) \times 100(\# \text{of channels}) = 20160000000 \text{KB} = 20000\text{GB}.

The storage load for each user is $20000\text{GB}/1000000 = 20\text{MB}$.

(data using in our computation is collected from a famous network multicast system, “PPLIVE”)

We see that the estimated average storage load for each user is 20MB which is not a big load for PC’s hard-disk.

(2) Adding data redundancy

As we know, network dynamics has serious impact on the resource availability and performance of the distributed systems. An intuitive solution to such problem is to add data redundancy to the system.

By analyzing users’ behaviors, we found that the average stay-up time for ordinary users in Internet TV multicast system is comparable to the cable TV network. So in the simulation, we assume an average stay-up time of 40 minutes. Multiple replicas of a same media content are kept in the system, and the redundancy rate is set to be 10.
Chapter 5

Simulation and Results

The objectives of the simulation are to 1> simulate the major components and operations of the VOD system, 2> measure and evaluate the performance and user-perceived Qos. The simulation is designed based on a BitTorrent simulator called GPS [5] developed by Weishuai Yang and Nael Abu-Ghazaleh from Department of Computer Science, Binghamton University. The simulation project was developed in Java.

5.1 Introduction to GPS: a general peer to peer simulator

The GPS is a general peer to peer simulator that accurately and efficiently simulates the P2P network. The peer to peer file sharing protocol, BitTorrent, is the only protocol been fully simulated, simulation of other protocols still remain to be designed. The GPS employs a message-level rather than packets level simulation for the consideration of efficiency. The simulator has good portability and extensibility for P2P protocols other than BitTorrent. The GPS provides infrastructures that required by most P2P protocols, so new prototypes can be simply plugged in and executed over the existing platform. The GPS provides an
accurate and sufficient simulation on the underlying network protocols details than some existing P2P simulators. The GPS is an event driven, rather than a time driven simulator. Instead of advancing the simulation time in fixed increments and processing events synchronously at each clock tick, processing and time advancement is triggered by occurrence of events.

![Figure 5.1: GPS Architecture](image)

Figure 5.1: GPS Architecture

The architecture of GPS is shown in figure 6.1. The simulation framework consists of several core components: (1). simulation engine, a discrete event simulation engine with global queue and event scheduler. (2). topology, a global object containing network related information (delay, bandwidth values between two nodes) which can be retrieved by any virtual node. (3). protocol, the detailed implementations of all P2P protocols, like the overlay structure, searching and communication mechanism. (4). agent, this is the end point of the network that constructing and consuming messages. (5). document, the object type that is transferred in the network.
5.2 Simulation setup

5.2.1 Simulation parameters

For simplicity, we assume there is only one channel in the IPTV system. The channel is operated by one media server and one track server. The setups of the simulation are shown in table 6.1.

Because of the small percentage of VOD users among the total number of users, we’re not interested in the impact of the VOD traffic to the entire IPTV/VOD network. We assume there is only one user subscribed to the VOD service. We use the exponential distribution to randomly determine when a node join and depart. The frequency of join/depart events is tunable by changing parameter called churn rate. We set the replication rate as 6 which mean the server selects and asks 6 nodes to save 6 copies of a same piece.

5.3 Simulation design

The simulation is designed and performed upon GPS, the BitTorrent simulator which is natural platform for P2PTV system.

5.3.1 Content distribution infrastructure

There are totally 1023 nodes (the VOD user is excluded) engaged in the IPTV streaming networks in one entire run of simulation. The first node joins the
network at the beginning of the simulation (simulation time 0), other nodes join.

Table 5.1: Simulation setups

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Nodes</td>
<td>The number of nodes in the IPTV/VOD network. Each user subscribes to the service via one end node. The number of active nodes in the network varies over time.</td>
<td>1024</td>
</tr>
<tr>
<td>Number of VOD Users</td>
<td>The number of VOD users</td>
<td>1</td>
</tr>
<tr>
<td>Number of Track Servers</td>
<td>Track server</td>
<td>1</td>
</tr>
<tr>
<td>Number of Media Servers</td>
<td>Media server</td>
<td>1</td>
</tr>
<tr>
<td>Churn Generator</td>
<td>the component generating the events of nodes’ join and departure</td>
<td></td>
</tr>
<tr>
<td>Upload Rate per Single Node (KB/s)</td>
<td>The upload rate per single node</td>
<td>10</td>
</tr>
<tr>
<td>Number of Simultaneous Partners</td>
<td>The maximum number of simultaneous partners at any moment during a VOD session</td>
<td>15</td>
</tr>
<tr>
<td>Streaming Rate of IPTV Server (KB/s)</td>
<td>The rate that the media server upload</td>
<td>32</td>
</tr>
<tr>
<td>Piece Size (KB)</td>
<td>The size of pieces</td>
<td>256</td>
</tr>
<tr>
<td>Replication Rate</td>
<td>The number of replicas a piece is saved</td>
<td>6</td>
</tr>
</tbody>
</table>

the network gradually. The source server keeps upload new contents. The newly joined IPTV users only download newly uploaded contents.
5.3.2 Distributed video on demand

After a program was multicast lively, the source server starts to accept requests for VOD services. In the simulation, there is only one node subscribed to the VOD service. The node joins and downloads a media file with size of 500MB. For simplicity, we assume the VOD user never departs till the end of program after joined the network. The IPTV users join and leave the network frequently following the churn rate. After joins the IPTV/VOD network, the node downloads a media file ranges from piece 0 to piece 2000 (lasts 250 minutes).

After the VOD node joins, it gets an index table broadcasted by media server. By looking up the index, the node begins to retrieve pieces one by one. The following literature explains how to request the first piece of a media file from an arbitrary partner. The node first sends request for setup connection with a target node, also called partner. If the partner is currently active, an acknowledgement is sent back and connection between the nodes is built. If the target node is not active, the VOD node waits for 5 seconds and tries connecting again. The VOD node views the target node as offline after three times of trials. After this, the VOD node re-lookups the index, gets the second target node (backup). Once again, the above procedures repeat on the backup partner. If connection is successfully built between the two, the VOD asks its partner whether the other end has the first piece, that’s piece 1. The transfer of piece 1 begins at the next moment if a
positive response is received. The negative response means that the partner doesn’t have piece 1, probably due to the error in index look-up, or expiration of piece 1 in that node. Upon the finish of transfer of piece 1, the node begins to request the next piece. The upload rate of the partner is set as 10KB/s. Since the VOD node can simultaneous request 15 pieces, the node requests the next consecutive piece which is not received yet, after it finishes a currently downloading piece. The above procedures (see figure 6.2) continue until the node receives all pieces in the file (in our simulation, that’s piece 2000).

5.3.3 Churn generator

The performance of most distributed applications is significantly influenced by the frequent turnover of nodes. In our simulation, we use a Churn generator to more accurately simulate networked environment with existence of dynamism. We assume the frequency of nodes join and depart follows the exponential distribution. In our simulation, the nodes’ join and depart are viewed as events. The starting of a VOD session of a node is accompanied with an event of node join. After joining the system, the churn generator generates an event of node departure and set the intriguing time of the event. This event will trigger the
departure of the node at the designated time. Identically, the departing event again generates an event of node’s join in the future. The departed node will be reactivated when the join event is triggered. In addition, the frequency of nodes’ join and departure can be controlled by tuning the churn rate parameter.

Figure 5.2: Procedures of Single Piece Download
5.4 Simulation results

5.4.1 Downloading rate

(1) The influence of dynamism to download rate

The download rate at client’s side must be sufficient to allow fast and smooth playback of high quality media files. So the download rate is an essential performance metric for VOD system. There are several factors determine the client-side download rate: 1> the available inbound bandwidth of end users, 2> the environment condition, for example, network dynamism, variation in the number of active users, bandwidth fluctuation. 3> the topology of the content delivery platform. The first one is upper-bounded by end system’s hardware settings. The second and third factors strongly affect the utilization of bandwidth available.

As we knew, the dynamism of network greatly influences the client-side throughput. Without up-to-date guidance, it often takes long latency for a peer to locate its desired data. In our design, we employ assistances from track server to increase accuracy of search. This scheme significantly enhances the download performance by decreasing the delay of piece request delivery. In the simulation, we collected data on download rate with varying average length of VOD session/Peer lifetime for both tracker-supported search scheme and random search scheme.
In figure 5.3 (a), (b), we plotted the download rate as a function of peer lifetime using tracker-supported search and random search. The larger average peer lifetime translates to lower level dynamism, or in other word, more stable network. We see that the download rate fluctuates greatly with the variation of peer lifetime using the latter scheme, while remains steady if the former scheme is used. Since the regular scheme makes trial of piece request to randomly chosen peer, the download rate becomes lower when nodes frequently join and depart. The tracker-supported one locates data more accurately by consulting tracker, so the download rate won’t be significantly affected as long as there are active nodes with copies of requested pieces in the network.

(2) Playback smoothness

The figure 5.4 (a), (b) shows the playback smoothness, or the variation of download rate in a VOD session as playback point advances. In 6.4 (a), the download rate remains near-constant in duration of a session with exceptions in the beginning of the session and some occasional outlier. The low rate at the beginning attributes to the start-up buffering. The occasional deviations suggest the existence of unexpected changes in dynamism level. In 6.4 (b), the download rate fluctuates noticeably in duration of a session. This indicates that when there is no guidance provided by track server, the search of piece is aimless. The large number of failed search resulted in frequent search redirection, thus causes great and unpredictable delay and fluctuated download rate. We see that the
tracker-supported search guarantees a more stable download rate, and as well more smooth playback.

Figure 5.3(a): Download rate (KB/s) as a function of peer lifetime (sec) using tracker-supported search

5.4.2 End-to-end traffic

The distribution of workloads among seed/server and regular peers determines the scalability, and reliability of the VOD system. Lower workload on the former implies a potentially more scalable, reliable system. In this experiment, we measured total amount of VOD traffic at the server and engaging nodes. One single VOD user downloads a media file with size of 500MBs. In table 6.2, we show the amount of traffic uploaded by the media server during a VOD session.
Figure 5.3(b): Download rate as a function of peer lifetime using random search

Figure 5.4(a): The variation of download rate over time using tracker-supported search scheme
Figure 5.4(b): The variation of download rate over time using random search scheme

We notice that the server still uploads a large portion of traffic. The explanation for the largest share of traffic (one third) is that since the server never departs the network, it is the most available node and also most altruism node.

5.4.3 Index table multicast

Now we analyze and compare the costs for both multicast and unicast index table to VOD users. To achieve a better understanding, we calculate how much bandwidth is consumed for both cases by assuming the following condition:
Then, the total size of files we save in the system is

\[ S = R \times 3600 \times 8 \times 10 \times 10 \times R \]

and the number of pieces stored is

\[ N = \frac{R \times 3600 \times 8 \times 10 \times 10}{L} \]

Table 5.2: The percentage of traffic contributed by server and the number of partners a VOD user has, with various peer lifetime (min)

<table>
<thead>
<tr>
<th>Peer lifetime</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>25</th>
<th>30</th>
<th>40</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic</td>
<td>49%</td>
<td>42%</td>
<td>39%</td>
<td>37%</td>
<td>31%</td>
<td>29%</td>
<td>28%</td>
<td>26%</td>
</tr>
<tr>
<td>Number of Partners</td>
<td>172</td>
<td>211</td>
<td>235</td>
<td>241</td>
<td>296</td>
<td>326</td>
<td>347</td>
<td>351</td>
</tr>
</tbody>
</table>

Table 5.3: Configuration for the IPTV system

<table>
<thead>
<tr>
<th>Streaming Rate of Channels (KB/s)</th>
<th>Size of Piece (KB)</th>
<th>Lifetime of Media Files (day)</th>
<th>Service Period (hour)</th>
<th>Redundancy Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = 70</td>
<td>L = 256</td>
<td>10</td>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>

If \( R = 70 \text{KB/s}, L = 256 \text{KB} \) (same as BitTorrent), the size \( S \) would be 20 gigabytes, and number of pieces \( N \) equals 80000 pieces. We insert an entry to the indexing table for each piece by pairing its piece ID with its storing location.
Assume each entry has a size of 6KB (4KB for IP address, 2KB for piece# and additional info), then for 80000 pieces, the total size of the indexing table would be about 5MB.

Consider the naïve scheme, for a VOD system, at each second, there are 5 users join, each of the users must get the indexing table before it can begin downloading. Then the source server must at least allocate a bandwidth of \(5\text{MB/s} \times 5 = 25\text{MB/s}\) just for transferring the indexing table. This is only a conservative estimation, in the peak hours, the busy workload could be much higher which may leads to even more bandwidth consumption. While, if we multicast the indexing files every 5 minute, then 21MB payload is transferred in an interval. As we computed, the size of indexing files for 10 days is 5MB, 2.5MB out of the 5MB(30%) indexing files are needed to be transferred every 5 minutes. This means the multicast of metadata files adds an extra bandwidth load at about 11.5%.
Chapter 6

Conclusion and Future Work

In this thesis, we proposed a prototype distributed on demand streaming system in a BitTorrent-based network. The proposed system provides on demand streaming service of the media contents which were released by live streaming system. The system provides fast, efficient streaming service to users in peer to peer pattern. The system indicates good performance in the following aspects. First, the decentralized architecture and the efficient content location and search make the system stands well even in face of high nodes join and depart rate, though the system is built upon a wide-area network with a large number of nodes participating and thousands gigabytes of data involved. Second, search in a distributed, wide-area network is always resource-consuming job. Our system reduces the search overheads significantly while introducing slightly increased loads in the server.

As future works, the dissemination of search information can be made decentralized. Since the updating and dissemination of the search information is still a significant overhead. For instance, we can have two-tier content indexing. Some nodes in the network volunteer to be the index nodes which share the
responsibility of indexing with the server.

More future work can be done on designing complex content distribution policy. The server selects better nodes to cache more segments or segments with higher popularity (more access demands). Intuitively, the parameters used to determine the goodness of the peers include the availability of nodes, the upload capacity of nodes and the amount of segments and their popularity cached in the nodes.
Bibliography


