ENABLING LARGE-SCALE PEER-TO-PEER STORED VIDEO STREAMING SERVICE WITH QOS SUPPORT

by

Masaru Okuda

BS, Brigham Young University - Hawaii, 1989

MS, University of Pittsburgh, 1996

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

THE SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Masaru Okuda

It was defended on

August 18th 2006

and approved by

Taieb Znati, PhD, Professor, Department of Computer Science

Richard Thompson, PhD, Professor, Department of Information Science and Telecommunications

Martin Weiss, PhD, Associate Professor, Department of Information Science and Telecommunications

Michael Spring, PhD, Associate Professor, Department of Information Science and Telecommunications

Alexandros Labrinidis, PhD, Assistant Professor, Department of Computer Science

Dissertation Director: Taieb Znati, PhD, Professor, Department of Computer Science

ENABLING LARGE-SCALE PEER-TO-PEER STORED VIDEO STREAMING SERVICE WITH QOS SUPPORT

Masaru Okuda, PhD

University of Pittsburgh, 2006

This research aims to enable a large-scale, high-volume, peer-to-peer, stored-video streaming service over the Internet, such as on-line DVD rentals. P2P allows a group of dynamically organized users to cooperatively support content discovery and distribution services without needing to employ a central server. P2P has the potential to overcome the scalability issue associated with client-server based video distribution networks; however, it brings a new set of challenges.

This research addresses the following five technical challenges associated with the distribution of streaming video over the P2P network: 1) allow users with limited transmit bandwidth capacity to become contributing sources, 2) support the advertisement and discovery of time-changing and time-bounded video frame availability, 3) Minimize the impact of distribution source losses during video playback, 4) incorporate user mobility information in the selection of distribution sources, and 5) design a streaming network architecture that enables above functionalities.

To meet the above requirements, we propose a video distribution network model based on a hybrid architecture between client-server and P2P. In this model, a video is divided into a sequence of small segments and each user executes a scheduling algorithm to determine the order, the timing, and the rate of segment retrievals from other users. The model also employs an advertisement and discovery scheme which incorporates parameters of the scheduling algorithm to allow users to share their life-time of video segment availability information in

one advertisement and one query. An accompanying QoS scheme allows reduction in the number of video playback interruptions while one or more distribution sources depart from the service prematurely.

The simulation study shows that the proposed model and associated schemes greatly alleviate the bandwidth requirement of the video distribution server, especially when the number of participating users grows large. As much as 90% of load reduction was observed in some experiments when compared to a traditional client-server based video distribution service. A significant reduction is also observed in the number of video presentation interruptions when the proposed QoS scheme is incorporated in the distribution process while certain percentages of distribution sources depart from the service unexpectedly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

This dissertation is a culmination of many individuals' labor and sacrifice. Without their support, I have not been able to come thus far. I would like to thank the members of my dissertation committee, Dr. Taieb Znati, Dr. Richard Thompson, Dr. Martin Weiss, Dr. Michael Spring, and Dr. Alexandros Labrinidis. I am ever grateful to my adviser, Dr. Taieb Znati, for the valuable training he has given me so that I may learn how to conduct a research project. Dr. Znati was the sole reason for me to come to Pittsburgh to pursue my graduate study. I thank him for his patience and all the funding I have received through him. I also would like to thank Dr. Richard Thompson for his support. Without him extending me the half-GSA position in the second year of my master's program, I would not have been able to be here today, but to go home without completing the program. He has been the best teacher in all of my school years and has become the model for my teaching career.

I would like to thank the faculty members of the Telecommunications Systems Management program at Murray State University and the university administration. I am grateful to Dr. Danny Claiborne, Dr. Neal Webber, and Dr. Gary Brockway for their invitation to join the MSU faculty and allowing me complete this dissertation during the first year of my teaching appointment.

I am grateful to my church friends that are affiliated with various academic institutions: Dr. Karl Johnson (University of Pittsburgh), Dr. Bradley Agle (University of Pittsburgh), Dr. Brent Adams (Brigham Young University), Dr. Harry Kim (Carnegie Mellon University), and Dr. Neal Williams (Brigham Young University - Hawaii). They have been my inspiration and great mentors. I am a recipient of their genuine love and concerns, which sustained me day-to-day.

I am truly thankful to my parents, Yoshiharu and Atsumi Okuda. Yoshiharu, my father,

taught me, by his example, how to live a happy life by diligence and honest work, without needing to have complicated things of the world. Regretfully, my father passed away without seeing me completing my study. Atsumi, my mother, has been the greatest influence in my upbringing. I am grateful that she had taught me those things that really matter in my life while I was young. I continue to practice those things that she had taught me while I was in kindergarten. I am also grateful to my sister, Yumiko, and my brother, Akira, for quietly observing and supporting what I do.

I am truly thankful to my dear wife, Megumi, my eternal companion, and our precious children, Nozomu, Shari, and Michael, for their support. I am completely and eternally indebted to them for the sacrifices they had made to allow me to pursue my education and finish this research work. I now realize that my zeal for higher learning was made possible by an exchange for something that I can never take back. With all of my heart, I thank them for their faith, long suffering, and love unfeigned.

Finally, I thank my God, for he is my strength. I acknowledge that every good gift comes from Him. Without His mercy and grace, I cannot accomplish anything. With Him, nothing is impossible.

In Murray, Kentucky
November, 2006

1.0 INTRODUCTION

With the advent of recent Internet technological advances, multimedia streaming services are gaining increasing importance. Video streaming applications, such as on-line DVD rentals, have the potential to enrich our lives and create new business opportunities.

In order to support the distribution of high-volume video streams to a large number of users, the video server must be equipped with a substantial amount of transmit bandwidth. A long playback time associated with a typical streaming video also limits the number of users a video server may support in a given period of time. Due to the limited scalability associated with the traditional client-server based network system, the design of a cost-effective network that satisfies the peak-hour demand of high-volume stored-video streaming service is difficult to achieve.

As high performance end-user systems are becoming widely available and the number of subscribers to broadband Internet access is rapidly raising, a new computing paradigm known as *Peer-to-Peer* (P2P) has emerged. P2P enables direct exchange of content among a group of end users without the centralized management structure. Once a user finishes downloading a content, the user takes on a server role to distribute the received content to a small number of other users. P2P offers a framework in which a large-scale, distributed, and self-organizing content distribution network can be constructed.

Although P2P has the potential to overcome the scalability problem associated with the traditional client-server based content distribution networks, it introduces new set of challenges. First, a limited uplink capacity of a typical broadband access technology, such as ADSL and cable modem, may allow only a fraction of what is required to stream a video at the nominal rate. As such, not all participating users may be able to become contributing sources when bandwidth intensive content is distributed. Second, streaming

allows not only the playback of video frames as they arrive, but also the discarding of video frames once they are played back. Since the contents of user buffer is constantly changing in streaming, the design of advertisement and discovery schemes must capture the dynamic nature of video frame availability. Third, video streaming requires an orderly and timely delivery of video fames to ensure a smooth playback. Due to uncertainty in peer's behavior, including a sudden departure from the service, excessive delays in video frame arrivals may be observed by the receiving user. A video distribution scheme which minimizes the impact of distribution source losses to the video presentation is desired. Fourth, support for and exchange of video frames among fixed and mobile users is important in providing seamless video distribution service. Lastly, in order to support the above functionalities, an efficient and robust design of a streaming network architecture is needed.

To address these issues, the design of the streaming network should be efficient in the use of server bandwidth, scalable to accommodate increase in the community size and content volume, versatile in supporting diverse user requirements, including mobility support, and responsive to a dynamically changing community environment. In addition, it should be resilient to sudden network changes to minimize the impact on the streaming service.

A few attempts have been made to enable the deployment of multimedia streaming over the Internet. Their solutions typically focus on the support of applications that require low bandwidth streams to many users, such as news tickers and real-time stock updates, or high quality streams to a small set of users, such as video conferencing. To our knowledge, a viable solution does not exist that adequately addresses challenges associated with the delivery of high-volume video streams to many and diverse users of Internet.

In the remainder of this chapter, background information on video streaming is provided. Existing streaming techniques are described and their shortcomings are discussed. Research objectives are established, design requirements are identified, and research methodology is discussed. The chapter concludes with the contributions made through this research study and the outline of the rest of the document is provided.

1.1 VIDEO STREAMING OVERVIEW

Streaming is defined as "a technique for transferring data (usually over the Internet) in a continuous flow to allow large multimedia files to be viewed before the entire file has been downloaded to a client's computer" [3]. Prior to the availability of streaming technique, multimedia content was distributed no differently than any other ordinary files (i.e. text files, executable files). They were all transmitted as "files" using file downloading protocols such as ftp and http. Due to the large volume of data associated with a typical multimedia file, a long transmission time as well as a large storage space were required before the playback could begin. Furthermore, there was no way for the users to "peek" into the content to see if it is the video they would like to watch. This was often inconvenient, if not unacceptable, to the users due to a long waiting time and a large amount of wasted resources when the content of the video turned out to be something they were not interested in.

Streaming enables near instantaneous playback of multimedia content regardless of their sizes. It is made possible by a steady transmission of data packets in such a way that the users will receive the needed packets moments ahead of the time they must be played back. Streaming reduces the storage space and allows users to "quit" receiving the stream, if not interesting or satisfactory, before the entire file is downloaded.

Streaming allows live and pre-recoded content to be distributed. Live streaming captures audio/video signals from input devices (e.g. microphone, video camera), encodes the signals using compression algorithms (e.g. MP3, MPEG-4), and distributes them in real-time. Typical application of live streaming includes surveillance, broadcasting of special events, and distribution of information that have the prime importance in real-time delivery. In live streaming, the server side has the control over the selection of the distribution content and the timing of their streaming. The user involvement is typically limited to joining and leaving the running streaming sessions.

Pre-recorded or stored streaming distributes pre-encoded video files stored at a media server. Sample applications include multimedia archival retrievals, news clip viewing, and distance learning through which students attend classes on-line by viewing pre-recorded lectures. Under stored streaming, when and what title of the video will be streamed are

dictated by the user. As such, a great amount of load may be placed on the media server when supporting a large number of asynchronous users (i.e. users whose streaming requests arrive at different times) with diverse interests. Our research focuses on the support of stored streaming.

Streaming brings new challenges to the distribution of multimedia content. Due to its strict on-time delivery requirement of data packets, a mechanism is needed to regulate the flow of packets over the Internet. An active research has been conducted in the areas of rate control, to cope with the time-varying bandwidth availability of Internet [39, 37], buffer management, to overcome delay variations [22], and error control, to reduce the impact of packet loss [28]. Progress in standardization work has produced specifications on key aspects of streaming, such as media encoding [24], media transport and session control [33], and media description and announcement [26].

However, these schemes and specifications mostly focus on the behavior of individual streams and the semantics of control messages. They do not address fundamental issues relating to the efficiency, scalability, versatility, and resiliency of streaming infrastructure. Our research focuses on these performance attributes of streaming service.

1.1.1 System Components

This section describes the architectural components of a streaming network. While there are different implementations of streaming systems, the following abstracted system components are always present: content producer system, content distributor system, content consumer system, and content distribution network.

Content producer system is responsible for the encoding of content to a format suitable for the transmission over the network and for the decoding at the consumers. For live streaming, content producer encodes the audio and video signals that arrive from microphones and video cameras in real-time and transfers the encoded streaming media to the content distributor. For on-demand streaming, it performs off-line encoding of content and have them available at media server.

Content distributor system is responsible for the management of content, user profiles,

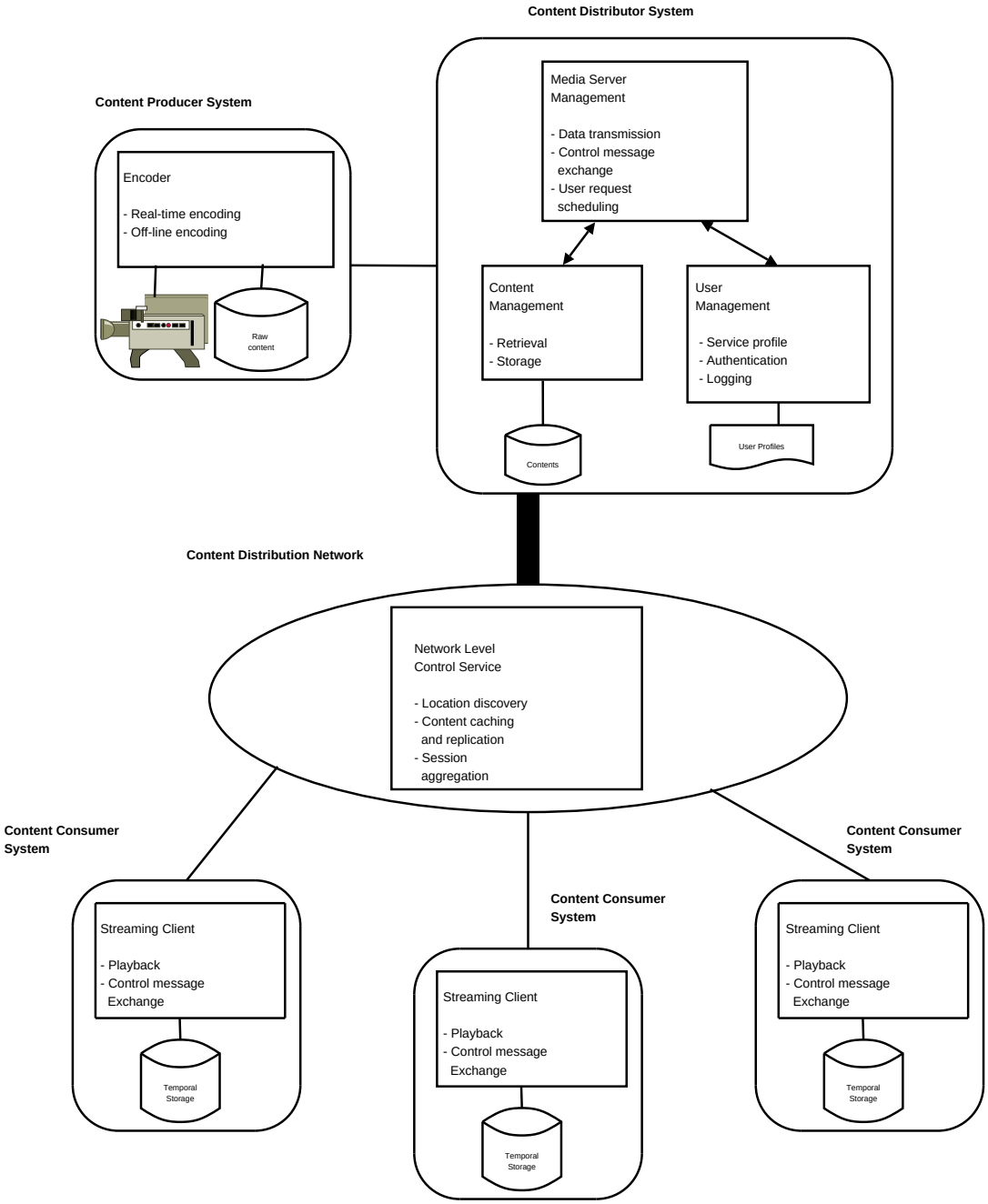


Figure 1.1: Architectural Components of Streaming Network

and transmission of content. Through the content management function, it controls the storing and retrieval of content to and from the media server. The user management function maintains the profile of end-users for service lookup, authentication, and logging purposes. Through the streaming server management function, it exchanges stream control messages, accepts and schedules user requests, and distributes content to the consumer systems via distribution network.

Content consumer system is a part of a user's end system. The types of consumer system include PC, PDA, Internet TV, cell phone, and other types of electronic device equipped with networking capability. It has a streaming client program that decodes the content and plays back the audio and video signals. It also exchanges control messages with the streaming server through the distribution network.

Content distribution network provides physical connectivity between content distributor and consumers through the backbone network (i.e. the Internet). From the backbone network, access link extends to the distributor system and consumer system. In general, the bandwidth available at the content distributor's access link is much greater than one from the content consumer's. Embedded within the content distribution network is the control services that are often not directly visible by the end user systems but vital to the operation of the distribution service, such as location discovery, session aggregation, and caching/replication.

A high-level abstraction of streaming network system with its key components are depicted in Figure 1.1:

In the following section, we identify the types of network on which a streaming service can be offered today, describe their focal features, and discuss their key mechanism being employed to achieve their performance and functional objectives.

1.2 EXISTING STREAMING NETWORKS

There are three prominent ways in which a streaming service may be offered over the Internet. The first approach is to use the Web based distribution mechanism. This is the

most commonly used method to distribute small streaming content. For a large scale service, streaming content is distributed through a *Content Delivery Network* (CDN) which improves the scalability of Web based content distribution. The second method is to use a network specifically designed for the distribution of streaming content. A number of networks have been proposed that are specialized in on-demand delivery of video streams. In this document, we refer them as *On-demand Multimedia Streaming Networks*. The third way is through direct exchange of streaming content among end users in a cooperative network environment. Active research is conducted on the peer-to-peer based streaming service.

1.2.1 Web Based Distribution Networks

World Wide Web (WWW or Web) has become a large and successful information distribution network today. Thousands of web servers dot the Internet and millions of users retrieve information from the servers through web browsers. While various types of content may be distributed through the Web, its primary use is to retrieve small-size documents and images. Due to client-server based computing model, Web based content distribution architecture suffers from server overloading when a large number of access requests arrive. A great deal of research effort have been place on the scalability challenge of Web-based content distribution and the studies mainly focused on how to lighten and distribute the load on the servers. Content caching and replication are two primary methods to cope with the server overloading problem.

1.2.1.1 Caching and Replication Through caching, recently accessed web page content is stored at a temporal buffer so that subsequent requests to the same content can be retrieved locally or from a nearby cache server. Proxy server is a dedicated cache server that provides web caching service for many users. Scale of economy can be achieved through proxy servers since many users may request the same content. Caching is applied at a meaningful unit of content (e.g. individual graphics on a web page) and cached content is refreshed periodically to keep them fresh.

Web caching reduces the access latency, conserves CPU cycle of a Web server, and cuts

down the network bandwidth usage. However, it is generally considered not a good solution for streaming video content. Caching of a video stream requires a very large buffer space since the entire content must be saved. A conflict exists with a refresh interval since the playback time of a typical video stream takes a longer time than the refresh interval of the cache.

Replication is a mechanism to automatically create and manage duplicate copies of data from one system to other systems. Many web sites replicate their content at multiple servers in order to reduce the load on the originating server. Replication also provides server redundancy in case of server and network failures.

1.2.1.2 Content Delivery Network In recent years, an advanced from of replication service is offered through *Content Delivery Networks* (CDNs) (e.g. [1, 29, 20, 10]). A CDN is a set of originating and mirror servers that enable handling of a large amount of web page requests. In addition to increased server capacity and resiliency, a CDN provides controlled load balancing and improved content accessibility. A CDN implements a scheme, such as ping triangulation, to identify the locations of users in relation to geographically dispersed mirror servers. Once the closest mirror server is identified, the originating server redirects web client requests to it. The forwarding operation of client requests is performed transparently to the web clients.

While content replication offers an effective means to distribute the load and increase the overall availability of desired content, it is questionable whether a sufficient number of replication servers can be deployed to overcome the bandwidth bottleneck problem associated with the distribution of high quality streaming content to mass users. Intelligent load balancing performed by a CDN gives an additional level of load distribution, yet the amount of load the network can handle is fixed by the total CDN capacity. Special events and programs often generate more demands than what the network can handle in a short period of time and CDN will not be able to support those excess demands. What is desirable is a mechanism that allows dynamic addition and removal of replication servers at those locations where demand becomes high. In order for a CDN to be truly effective, a large number of replication servers must be deployed throughout the Internet. Such an undertaking may

not be possible by small ISPs and independent business owners.

1.2.2 On-Demand Multimedia Streaming Networks

In addition to WWW and CDN based content distribution, studies have been conducted on how to support on-demand delivery of multimedia streams over the Internet (e.g. [6, 2, 19, 18, 14, 34, 15]). Their goal is to provide efficient *Video-on-Demand* (VoD) service to a group of users. VoD service enables immediate distribution of video streams to users, from the beginning of the content, regardless of the time at which the service request arrives in relation to other on-going streaming sessions.

The fundamental challenge of VoD service is how to meet the on-demand expectation of users without consuming a large amount of bandwidth at the content server. A number of schemes have been proposed that focus on the efficient bandwidth usage of the content server. Their approach involves some form of session aggregation to reduce the amount of data flow and align the service start time. A common thread in all schemes is the use of multicasting. Some schemes propose how to provide efficient and practical multicasting while others assume the availability of multicasting to all participating users. In this section, we review several approaches in providing multicasting as well as how session alignment is achieved by various schemes.

1.2.2.1 Session Aggregation Multicasting enables transmission of packets from one source to multiple destinations without having to send duplicate copies of packets to each destination. IP multicast implements this service at the IP layer and offers efficient group communication. In order for IP multicast to be effective, all IP routers must be equipped with multicasting functionalities. However, very few routers on the Internet can support IP multicast. Overhauling the Internet with IP multicast capable routers is a task considered not feasible in the near future. In addition, a limited number of address space allocated for IP multicast makes its deployment unpractical for general use. For these and other reasons, researchers have looked in other ways to achieve an efficient and effective group communication.

In overlay network based multicasting, a network dedicated for the purpose of multicasting is created on top of existing IP network. Only those routers (i.e. overlay nodes) that are equipped with multicasting functionality participate in multicast specific service (e.g. branch trimming); other routers simply forward packets in multicast sessions as regular unicast flows.

The strengths of overlay network based multicasting include ability to deploy a large-scale multicast network without needing to upgrade all IP routers, support virtually unlimited number of multicast groups, and provide a practical solution for the deployment of group communication infrastructure on the Internet. However, it typically requires semi-permanently installed overlay nodes that will remain in service for an extended period of time or at least for the duration of the multicast session. For this reason, it is difficult to construct and maintain such a network within an environment where network nodes are highly dynamic, such as Ad-Hoc networks and P2P networks.

Application layer based multicast constructs a pseudo-multicast network among participating end systems without special support from the network. It is a pseudo-multicast because all connections between the end systems are actually unicast connections. Those end systems that function as the root or the branch nodes of a multicast tree establish multiple unicast connections to children nodes.

In order to build an efficient pseudo-multicast network, end systems must construct a spanning tree. This requires acquisition of the global network state, including the discovery of all existing nodes and how they are connected to each other. How to accomplish this task effectively is an on-going research topic.

Application layer multicast allows immediate deployment of pseudo-multicasting service by simply installing special software on end systems – no network component upgrade is necessary. It has been shown that application layer based multicast better utilizes the network bandwidth compared to the conventional unicast connections for group communication.

One of the challenges in supporting high quality streaming sessions through application layer based multicast is the lack of excess bandwidth at end systems. They must have adequate amount of transmit (Tx) bandwidth to support multiple streaming sessions to other end systems. Typical residential broadband access links (e.g. cable modem, xDSLs)

offer substantially lower bandwidth on the Tx direction than receive (Rx) direction (about 1/10th). This can be a limiting factor when considering the support of multicast sessions for bandwidth demanding streams.

1.2.2.2 Session Alignment Various session alignment techniques have been developed for the support of VoD service. They are largely divided into three types of techniques: batching, patching, and merging. *Batching* supports quasi-VoD service through queuing of streaming requests at the content server and servicing them in batches. The streaming content is divided into a number of different segment sizes and all segments are transmitted simultaneously and cyclically on separate multicast trees. Any request that arrives while the first segment is being streamed will be queued and serviced in the next cycle. Users who receive the first segment also receive the second segment concurrently while playing back the first segment. Once the first segment finishes, the users start the play back of the second segment from their buffer and begin receiving the third segment. This process is repeated until the last segment is received and played back. By assigning the length of each segment in increasing order (i.e. the first segment being the shortest), the waiting time of user request can be made small.

Patching supports true VoD service by establishing two streaming sessions from the user to the servers. One session goes to the on-going multicast stream and the other to the patching server. Any missed segments will be streamed from the patching server for immediate playback. The segments from the on-going multicast stream will be saved on local storage and played back as soon as the segments from the patching session finish. The common issue among batching and patching is that they require twice or more Rx bandwidth at the user system than the nominal playback rate since users must establish multiple streaming sessions concurrently. They also require a substantial amount of disk space in order to store segments from one of the streams while the other is being played out. In addition, they all assume the availability of multicasting capability at all participating nodes.

Merging incorporates a technique to fuse two independent streaming sessions that are close in their playback time to form a single session. The idea is based on a human factor

that slight fluctuations in the quality of video and audio streams can be undetected or tolerated by humans. The lagging stream plays out faster and leading stream plays out slower than the nominal rate of the playback. In some future time, there will be a point in time where the two streams will come together. Beyond this point, only one multicast streaming session will be maintained. The novelty of this approach is that, unlike other schemes which perform session aggregation *off-line*, it enables *on-line* or *dynamic* merging of multiple streams. However, it may take a long time to complete the merging process or may never be realized when the time gap is large between two sessions. It also requires an encoder/decoder system that dynamically changes the rate of stream.

1.2.3 Peer-to-Peer Networks

A new type of communication architecture, known as *Peer-to-Peer* (P2P), has emerged in recent years. It is a distributed information sharing architecture that enables direct exchange of content among a group of end users. P2P is based on a communication model where each node, called a peer, is capable of taking both the client and server roles. Peers provide hosting service for selected content, such as personal collection of image files, music files, electronic documents, and software programs. P2P provides a way to share information without the expense of maintaining a centralized server and management structure. Though individual peers may have limited service capacity, a P2P community as a whole could service a large volume of requests.

One of the primary challenges of P2P based content distribution is how to locate the desired content among globally distributed peers. It is a challenging task because the availability of content changes dynamically as peers join and leave the network. Effective content discovery has been a major research focus of P2P network. Two types of content and peer discovery mechanisms have been proposed: end application based discovery (e.g. [13, 23, 25, 7]) and overlay network based discovery (e.g. [36, 31, 32, 42]).

1.2.3.1 End Application Based Content Discovery End application based content discovery uses a flooding mechanism to find the existence of other peers and locate desired

content. A peer, X, sends an exploratory packet to a known existing peer[1], Y, in the P2P community. Once the exploratory packet arrives, Y sends X its IP address and port number at which the content distribution service is available. It then forwards the exploratory packet to those peers that had recently visited Y. This process is repeated at those recently visited peers until the exploratory packet ages out. Once X discovers the existence of other peers, it sends a query to them and requests a list of matching content to be returned.

End application based content discovery requires no central server or overlay nodes. It require no modifications to the existing network infrastructure. It is simple and easy to implement, yet very robust in times of node and link failures. No node join or departure process is involved, no recovery operation exists, and no peer is indispensable in this scheme. Popular content (i.e. content owned by many peers) can be located quickly and abundantly since the exploratory packets multiply and spread somewhat randomly by following daisy chained paths. End application based discovery scheme has a potential to work well with a small and dynamic network environment (e.g. Ad-Hoc network).

There are several shortcomings that make this discovery process unfit for global and large-scale search. First, due to flooding based discovery mechanism, it does not perform well when the size of a P2P community becomes large. Second, the search is limited, generally to about seven hops away, in order to constrain the amount of control messages exchanged. This will restrict the extent of the exploration and makes global search nearly impossible when many peers exist in the network. Third, the existing design assumes one and only one global P2P network and does not allow creation of disjoint P2P networks based on interests and purposes. This makes the end application based discovery difficult to provide an efficient and targeted search because the exploratory packets must be forwarded blindly to all peers within the reaching distance. Forth, rare content (i.e. content owned by a small number of peers) may not be easily found since peers are uncovered randomly.

1.2.3.2 Overlay Network Based Content Discovery

Overlay network based content discovery builds and maintains a network dedicated for the management and discovery of

[1]It is known to the peer administratively (i.e. some IP address of peers are published on the web and users can manually specify the starting peer.)

content and peers. This network is laid on top of the existing IP network by installing layer 3.5 (i.e. overlay layer) software modules at selected routers or network nodes. Participating overlay nodes collectively maintain a complete list of content location pointers in a distributed fashion, known as *Distributed Hash Table (DHT)*. Each overlay node is assigned a unique hash value. Every content on P2P network is also associated with a hash value generated from its content name and registered with the overlay node with the closest hash value. The overlay nodes are logically arranged in such a way that by following the pointers maintained at each overlay node, requests are guided to the managing overlay node of the requested content. CAN [31], Chord [36], Pastry [32], and others use overlay network based content discovery.

Overlay network based content discovery scales well compared to centralized or end application based discovery schemes. It requires no central server and no flooding is involved in discovery process. An increase in the number of peers as well as content can be handled gracefully by simply adding new overlay nodes. P2P network based on overlay discovery process allows co-existence of disjoint P2P networks. This is an important attribute because increase in the network load can be dealt with in two ways: by creating a bigger P2P network with greater overlay node capacity or by splitting P2P network and create smaller networks that are manageable in size. Unlike end application based discovery, it works well with the globally spread peers and content. The discovery of "unpopular" content is no more difficult than the "popular" content since both operations take the same amount of processing effort.

While overlay network based discovery offers these noteworthy merits in supporting large-scale P2P network, there are issues that need further study. First, an overlay node failure can result in a complete loss of service of a particular content. Though overlay nodes are distributed, pointers to a particular content are managed by a single overlay node. This is because the content with the same name will generate the same hash value and the hash value determines the managing overlay node. On this particular issue, overlay node suffers from the same shortcoming as the centralized server. Second, the overlay network structure maintenance can be costly and slow. The overlay network based discovery process has a provision for dynamic addition and removal of overlay nodes. While this process has been designed to bring affected overlay nodes to a stable and operational state, it is not designed

for frequent overlay node addition and removal. The convergence time can become long, especially when multiple overlay nodes are lost or inserted and it may introduce instability to a discovery service for a prolonged period of time. Third, it is expected that, as a P2P community grows larger, a query response may include a large number of content location pointers. Once a per receives these entries, there is no good way for a peer to tell which ones offer the "best" service (e.g. nearest). While this issue is not unique to overlay network based discovery, the problem is aggravated by its superior capability to gather globally spread content pointers to a single point in the network. Fourth, while overlay nodes may be inserted and removed as needs arise, the overlay nodes themselves are not dynamically created entities. In fact, they are considered a part of the network infrastructure and assumed to exist semi-permanently. The fixed nature of overlay nodes make them difficult to operate in a dynamic environment. Overlay network based discovery approach is more suitable for a stable and long-lived network than a dynamic and short-lived network. Fifth, a partial string search is not supported easily.

1.2.4 Summary

Table 1.1 summarizes the challenges and focus areas of existing streaming content distribution networks. Web-based network, on-demand multimedia streaming network, and P2P network are three prominent ways in which streaming media can be distributed over the Internet. Driven by a desire and a need for supporting a large user population, Web-based distribution mechanism focuses on the development of various types of caching and replication techniques that improve the scalability of streaming service and alleviate the server overloading problem. On-demand multimedia streaming network focuses on the efficient use of server bandwidth to handle on-demand expectation of users. Their efforts have produced various session aggregation and alignment techniques. P2P based distribution is challenged by the dynamic nature of server and content availability. Since content is scattered globally and their availability changes dynamically as peers join and leave the community, P2P based distribution focuses on ways to provide effective content discovery.

The relationship of problem space and focus area of existing streaming networks is de-

Table 1.1: Summary of challenges and focus areas of existing streaming content distribution networks.

Network Type	Typical Use	Challenge	Focus Area
WWW & CDN	Web page distribution	Server overloading (Scalability)	How to lighten and distribute load (Content caching and replication)
Multimedia Streaming	Streaming video distribution	On-demand expectation Server bandwidth limitation (Efficiency)	How to bundle requests for the same content for efficient streaming (Session alignment and aggregation)
Peer-to-Peer	Personally owned content distribution	Scattered and changing content availability (Dynamism)	How to locate desired content (Content discovery)

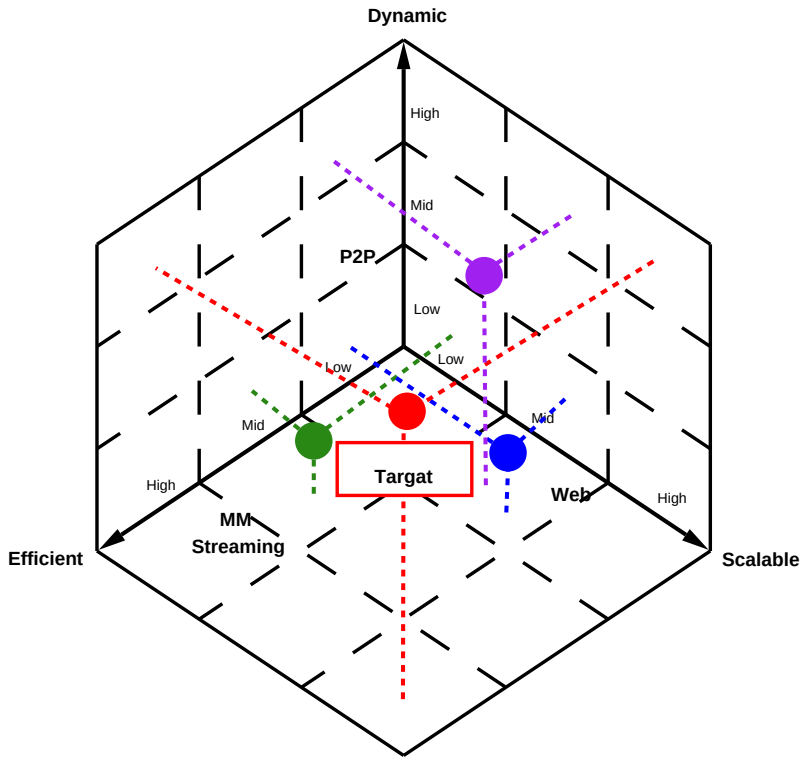


Figure 1.2: Relationship of problem space in streaming content distribution networks

picted in Figure 1.2. Our research focuses on the problem space where a great number of users request very large and long streaming content in a highly dynamic environment. This is marked at the peak of all three dimensions in the figure.

1.3 RESEARCH GOALS

This section presents the objective of this research and describes its functional and performance design requirements.

1.3.1 Research Objective

To design a network architecture, schemes, and algorithms to enable an efficient, scalable, and resilient on-demand peer-to-peer high-volume stored-video distribution service to large communities of dynamic users with diverse system capabilities, any time, any where.

1.3.2 Functional Design Requirements

Support for high-volume streaming service: Our primary goal of this research is to design a network architecture and supporting schemes to enable the delivery of high-volume streaming videos to a large community of users.

Support for on-demand service: We aim to design a streaming video network that can provide immediate delivery of requested content regardless of time it is received. As users join a distribution community, they should be able to find a distribution source and begin receiving desired content immediately.

Support for heterogeneous system: A streaming video network that accommodates heterogeneity of end user systems is one that is capable of supporting diverse user equipment (e.g. workstations, PCs, PDAs) and take advantage of differences in their capabilities (e.g. bandwidth availability).

Support for mobile users: The design of a new streaming video network should support both the fixed and the mobile users. Unique attributes (e.g. roaming) associated with the

mobile users should be incorporated into the design of a video distribution service.

1.3.3 Performance Design Requirements

Efficient: An efficient distribution network system allows effective use of resources, such as server bandwidth. The architecture of a new content distribution model should allow a substantial relief in the use of server access link bandwidth.

Scalable: Scalability is a measure of how well the proposed solution will work when the size of the problem increases. The design principles of a scalable system must allow graceful handling of increase in user population and content size. Furthermore, this should be achieved without exceeding or over burdening the resource capacity and computational complexity of the system.

Dynamic: A dynamic system is one that is capable of responding to constantly changing environment in real-time. The proposed distribution model should be able to handle frequent user joins and departures. In addition, it should provide support for the advertisement and discovery of time-bounded content availability.

Resilient: Resiliency is a measure of how well a system returns to an original state after being exposed to a severe condition. A resilient streaming content distribution network is one that is capable of recovering from unexpected network changes, such as network component failures and excessive congestions, with minimal impact on the distribution service.

There are other important attributes that enhance the design of a streaming network, such as security and fairness. In this study, we focus on the four performance attributes listed above. Other features are left for future study.

1.4 METHODOLOGY

To enable an efficient, scalable, and resilient streaming video service, various shortcomings that accompany the existing stream distribution networks will need to be overcome. A network design which is based solely on a P2P architecture or on a traditional client-server

model may not provide stability and flexibility desired for the support of an envisioned streaming network. We argue that a better approach is to design a hybrid network to take advantage of the stability of client-server architecture while reaping the flexibility of the P2P model. A client-server model is simple, well-studied, and successfully deployed over the Internet. It provides a default point of contact for participating users to help bootstrap the P2P community to distribute the streaming video. P2P provides scalability through decentralized operations in a dynamically changing community environment and allow participation of diverse end systems to help meet individual needs. By combining the two models, we create a design that better copes with the challenges of the distribution of high-volume streaming content to many and diverse users of the Internet.

To provide a streaming service on a hybrid network architecture, a discovery of video frame availability at participating peers is necessary. Existing P2P content discovery schemes do not serve well in the streaming network since the contents of user buffer are constantly updated as new video frames arrive in steady stream while old video frames are discarded. One of the ways to maintain the latest content availability of user buffers at their respective information storage point, frequent updates of video frame availability may be made. Similarly, frequent queries may be required to learn the latest buffer state of other users. We argue that a better approach in solving this problem is to incorporate the temporal property of content availability in the advertisement scheme. Specifically, we allow scheduling parameters of video stream receptions to be included in the advertisement scheme. Users will be able to describe their life-time of video frame availability in a concise and precise manner.

1.5 CONTRIBUTIONS

To enable scalable, efficient, resilient, and versatile streaming service to the users of the Internet, we propose the design of a new streaming video distribution network model, called *Virtual Theater Network*, and accompanying video reception and discovery schemes in a hybrid computing environment. The following contributions are made through this research:

- A design of a new streaming video distribution network model, *Virtual Theater Net-*

work, which allows organization of peer-to-peer communities (*VT Rooms*) to support the distribution of streaming videos among the members of a VT Room.

- A design of a *segmented* video stream reception scheme and an accompanying scheduling algorithm for an orderly and timely video segment retrieval that allows contributions from users with limited resource availability.
- A design of a video segment availability advertisement and discovery scheme which incorporates the parameters of segmented video reception scheduling algorithm. It enables users to capture the constantly changing segment availability information of user buffers and greatly simplifies the advertisement and discovery process such that one advertisement and one query is sufficient to post and retrieve the life-time of the segment availability of a user.
- QoS support in mitigating the video viewing interruptions in face of excessive delays, including those that are caused by one or more video distribution source losses.
- Incorporation of mobility information in the selection of video distribution sources that best satisfy the user needs.

1.6 DOCUMENT OVERVIEW

The remainder of this document is organized as follows. Chapter 2 provides selected literature reviews on P2P based streaming content distribution network. Chapter 3 presents the architectural design of the streaming network model we propose. Chapter 4 describes the video reception scheme and accompanying scheduling algorithm. Chapter 5 details the advertisement and discovery scheme to locate users with desired video segments. Chapter 6 provides the descriptions of a QoS scheme that reduces the chances of video presentation interruptions when distribution sources depart from the service unexpectedly. Chapter 7 presents the design and analysis of experiments to study the behavior of users in the proposed video distribution network. Chapter 8 concludes the document.

2.0 LITERATURE REVIEW

Selected literature review on peer-to-peer streaming systems are presented in this chapter. The reviewed schemes are grouped into two categories depending on the focus of their studies. Those schemes that seek to improve scalability in the data path are grouped in one category while others that try to add flexibility and scalability to the search path are grouped in another category. We identify problems that each scheme tries to solve and give brief descriptions of the approach it takes in achieving its goal. We describe how our work differs from theirs at the end of this chapter.

2.1 DATA PATH SCALABILITY

An overview of selected schemes that aid in the scalability of data path in P2P streaming network is presented. Schemes in this category are further divided in two groups depending on their multicasting approach. The first group of schemes are based on application layer multicast while the other group builds their schemes on overlay network based multicast.

2.1.1 Schemes Based on Application Layer Multicast

NICE [4], ZIGZAG [38], PeerCast [11], CoopNet [27], and OTS_{p2p} & DAC_{p2p} [40] are application layer multicast schemes which aim to enhance the scalability of data path in P2P streaming network. NICE and ZIGZAG are designed to support many live-streams with relatively small payload. PeerCast is designed to support live-streams in highly dynamic environment. CoopNet's main objective is to enhance the resiliency of streaming service of-

fered by the traditional content delivery network by supplementing it with application layer multicast. OTS_{p2p} & DAC_{p2p} evaluates an environment where peers have different transmit bandwidth capabilities. A brief overview of each scheme follows.

2.1.1.1 NICE NICE [4] is designed to support live-streaming applications such as news and sports tickers, real-time stock updates, and Internet radio broadcasts. These applications are characterized by their large user populations, relatively low bandwidth consumption, and real-time nature of data stream which can withstand occasional packet losses. When low bandwidth traffic is sent to a large number of users, control data generated by a large number of peers can occupy a significant percentage of the total number of bytes being sent across the network. In this type of environment, it is difficult to absorb the cost of control overhead. It is desirable to minimize the overhead cost as much as possible. NICE investigates the feasibility of using application-layer multicast to solve this problem.

NICE organizing peers in hierarchy of clusters and builds distribution trees that aim to achieve low latency and small control overhead. NICE members that are *close* to each other, in terms of end-to-end latency, organize themselves into clusters. Members that belong to different clusters but are close to each other are mapped to the same part of clustering hierarchy. This tree structure ensures that the latency is kept low when distributing streaming data. All members belong to the lowest layer, L_0, of the clustering hierarchy. A member that has the minimum maximum distance to all other members in the same cluster becomes the cluster-leader. The cluster-leaders at layer L_0 form layer L_1 cluster(s). A member at the *center* of the cluster at layer L_1 becomes its cluster-leader. This assignment repeats until the highest layer, which has a single cluster with only one member. There are at most $log\ _kN$ layers in total, where k determines the size of a cluster[1] and N is the total number of members. Hierarchically clustered peers in NICE multicast topology also helps maintain low control overhead (i.e. the number of peers a newly joining peer needs to contact) and localizes effects of member failures. Data delivery path is implicitly defined by the the hierarchical structure of the control path that no additional route computation is necessary for the data packet forwarding.

[1]Each cluster has a size between k and $3k-1$, where k is a constant.

NICE members maintain information of all other members in the same cluster. The cluster-leader sends a complete membership listing to each member periodically. With this information, each member sends $HeartBeat$ message to other members at a fixed interval. The loss of consecutive $HeartBeat$ messages beyond a predefined threshold is interpreted as a member failure. The $HeartBeat$ message contains the latency information as seen by the sending member and this information is used to determine the distance among each other. If the cluster-leader is removed, all cluster members independently decide who will be the next cluster-leader from the latency information maintained locally. When a cluster size becomes greater than $3k - 1$, the cluster-leader splits the cluster into two clusters with at least $\lfloor 3K/2 \rfloor$ size that result in two clusters of minimum maximum radius. Once the cluster splits are determined, the current cluster-leader sends $LeaderTransfer$ message to the new cluster-leaders. When the size of a cluster falls below k, it will be merged with a nearby cluster.

2.1.1.2 ZIGZAG ZIGZAG [38] asserts that there are three fundamental challenges associated with the distribution of live media over the application-layer based multicast trees. First, the end-to-end delay from the source to a receiver may become very long since packets follow a path with a number of intermediate receivers. Second, the receiver peers may abandon their descending at any time peers as they are free to join and leave the service. Third, control overhead associated with the support of a large number of users can become significant. ZIGZAG focuses on these issues and tries to overcome them.

ZIGZAG's goal is to build application-layer multicast trees which offer small source to receiver delay, quick and graceful recovery from failures, and small control overhead to maintain the multicast tree. The multicast tree is organized in layers of hierarchy identical to NICE. Unlike NICE, the data path does not follow the control path of the tree. In stead, non-head peers in a cluster (i.e. members that are not functioning as a cluster-leader) receive data from the head peer of another cluster. This unintuitive data path design actually reduces the node degree (i.e. the number of branches extending from a node) and help reduce the failure recovery overhead. Sample data paths of ZIGZAG are illustrated in Figure 2.1.

In this example, the content server, S, forwards packets to its cluster member (4) at

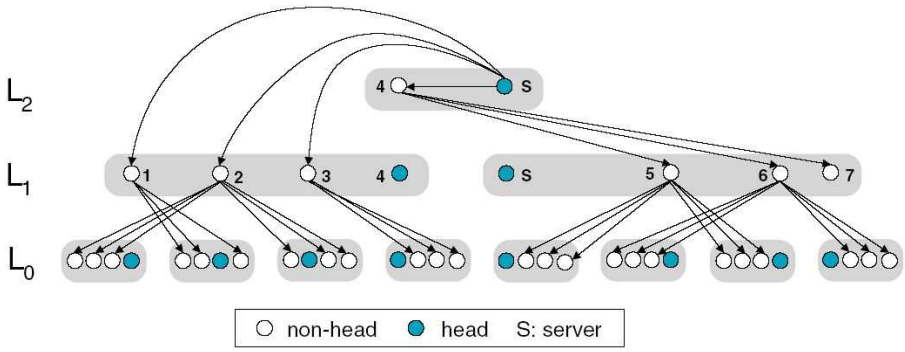


Figure 2.1: Sample data paths of ZIGZAG.

layer L_2. It also establishes data delivery paths to non-head members (1, 2, 3) of the cluster on the left at layer L_1. Notice that the server does not belong to this cluster. Packets are further forwarded by nodes 1, 2, 3 to non-head members of a cluster that they don't belong at layer L_0. A similar arrangement is made from node 4 to the members in the clusters on the right.

Addition of a new peer is attempted first to the cluster with a vacancy that has the least source-to-receiver delay. If all clusters are fully populated, the new peer is added to the shortest delay cluster and the cluster will be split in half during the next interval of cluster split operation. In addition to the shortest delay, ZIGZAG has a provision for using the number of node degree or the available bandwidth capacity as an option for choosing the joining cluster, though the details area not provided.

2.1.1.3 PeerCast PeerCast [11] aims to provide an efficient live media streaming service in a highly dynamic environment. PeerCast argues that while application-layer multicast can provide an effective means to deliver live streams, it is difficult to avoid service interruptions due to autonomy and unpredictable behavior of end user systems. The impact of a node departure can be large when it is the parent node of many children. PeerCast suggests that the design of an application-layer multicast should include a mechanism to address the dynamic nature of end user systems. PeerCast tries to achieve this goal by using a layered

P2P architecture.

PeerCast introduces *peering layer* which is placed between RTSP and UDP layers. Peering layer manages two types of sessions: data transfer sessions and application sessions. Data transfer sessions are established between end nodes and their parent nodes through a multicast tree. Application sessions are established between the peering layer and the end application at each node.

The robustness of data transfer sessions are realized by the implementation of a lightweight session protocol. When a parent node leaves a multicast group, it sends a redirect request to its children so they may re-attach to a new parent. If the new parent is unable to support those children, it redirects them to its immediate children. This process repeats until *unsaturated* node (i.e. a node with sufficient resource available) is found. In an event where all nodes are saturated, the peering layer indicates "resource unavailable" error to higher application layer.

Application sessions that are established between the peering layer and the end application shield the dynamic nature of data transfer layer from the end applications so that applications need not be concerned with the intricacies of the actual data transfer operations. While data transfer sessions may be torn down at any time and for any number of times, application sessions are expected to be intact for the entire duration of the live streaming presentation.

Using the peering layer mechanism, PeerCast investigates a variety of policies for node join, node departure, and tree maintenance. It concluded that a simple mechanism based on a layered architecture seemed to improve the resiliency of streaming service in a dynamic environment.

2.1.1.4 CoopNet CoopNet CoopNet asserts providing a reliable streaming service over P2P network is a challenging task due to sudden node departures expected in P2P environment. This is especially true with application-layer multicast where a departing peer may abandon a large number of children peers. CoopNet focuses on minimizing the effect of sudden node departures in application-layer multicast. It's goal is to avoid a complete loss of service during data path recovery so that the users may receive continual streaming

service in spite of peer departures.

CoopNet uses application-layer multicast trees to compliment, rather than to replace, the existing client-server based content delivery networks (CDNs). Unlike other application-level multicast schemes being reviewed in this document, it tries to support both the live stream service as well as on-demand audio/video streaming.

CoopNet tries to achieve resiliency through distributing a content over multiple application multicast trees. Each multicast tree is built independently and the sum of the packets delivered by all multicast trees reconstructs the original content. Packets carried by each multicast tree are coded in such a way that the streamed content can be played back at a fraction of the total quality independent of other multicast trees. Even if there are absence of packets from some multicast trees, it enables the continual presentation of streaming content with a lesser quality. This type of coding is known as *Multiple Description Coding (MDC)*.

The membership and the structure of multicast trees are managed centrally at the content server. When a node wishes to join a multicast group, it contacts the content server and registers its resource availability. The content server responds with a list of randomly chosen set of distribution nodes (one node for each multicast tree) that are capable of supporting children nodes. The joining node contacts the specified parent nodes to receive the stream.

On-demand streaming is carried out in a very similar manner as live media streaming. Joining node sends a request to the content server. If the content server is overloaded, the server returns a list of distribution nodes that have recently requested the same content who have indicated their willingness and capability to take on a CoopNet content server role. These nodes store the received content at their local storage and make them available for other peers to retrieve. The joining client contacts each node in the list given by the server to determine which peers have the needed MDC coded strips and requests them to send the stream. [27] does not provide details of on-demand streaming procedures.

2.1.1.5 OTS_{p2p} **and** DAC_{p2p} P2P network users generally have different system capabilities and resource availabilities. Taking these differences into considerations when supporting streaming media distribution over a P2P network is important. Xu, *et al.* [40] focuses on a specific case where peers have different and limited (i.e. less than nominal playback rate)

transmit bandwidth capabilities. In this type of environment, a requesting peer will have to establish multiple sessions to different supplying peers so that the accumulated bandwidth will be equal to the nominal playback rate. A requesting peer must also ask different segments to be sent from each supplying peer in order to playback the whole content. [40] raises the following two research questions under such P2P environment: 1) Given a limited availability of Tx bandwidth, what is the order in which a supplying peer should service requests waiting in a queue that came from different requesting peers? 2) Given a requesting peer must establish sessions to multiple supplying peers, how should it decided which peer supplies what segment?

Xu, *et al.* highlights a unique characteristic of a P2P based content distribution that more a content is downloaded, a greater the total content availability it becomes. For example, a content originally existed at a single site can be made available at 100 other cites if it were shared with 100 other peers. Xu, *et al.* uses this knowledge to define the design goals for the two questions being raised. For the first question, Xu, *et al.* services requesting peers in a queue in such an order that it results in minimizing the time taken to proliferate a particular content on the network. In other words, the goal is to maximize the total content availability in a shortest period of time. This will help other peers to locate and receive the same content quickly. For the second question, it arranges the supplying peers in such a way that it will minimize the Rx buffer size of the requesting peer. This ensures the shortest delay between the content request time and the start of playback time and helps shorten the time to proliferate the content in the network. Xu, *et al.* refers the first scheme as *Differentiated Admission Control protocol (DAC_{p2p})* and the second as *optimal data assignment algorithm (OTS_{p2p})*.

In addition to different and limited Tx bandwidth of peers, Xu, *et al.* assume the following attributes of the participating peers: 1) Supplying peer, P_s, can service at most one streaming session at a time, 2) Requesting peer, P_r, requires two or more supplying peers to reconstruct the entire stream, 3) Each P_s contributes a discrete value of bandwidth in steps of $(Nominal\ playback\ rate\ BW)/2^n$ and a peer is said to belong to class n. 4) Media can be divided into small and equal seize segments and each segment takes the same amount of time to transmit and process, 4) P_r can playout a segment only after the entire segment

arrives.

DAC_{p2p} is briefly described. In DAC_{p2p}, each supplying peer maintains an admission probability vector that favors the requesting peers that have the potential to become a supplying peer with a greater bandwidth contribution (i.e. higher class peers). Suppose there are four classes of requesting peers $P_r^1, P_r^2, P_r^3, P_r^4$ and the P_s belongs to $Class2$. Then, the admission probability vector of P_s for those P_rs would be $\{1.0, 1.0, 0.5, 0.25\}$. If P_s receives a request from a $Class1$ or $Class2$ peers, they are always accepted. If a request comes from $Class3$ P_r, the service decision is made according to a coin toss. If no request arrives beyond a predetermined threshold, the probability vector that has a value lower than 1.0 will be doubled in its value.

2.1.2 Schemes Based on Overlay Network Multicast

Vcast [17], Range Multicast [16], Overcast [21], and OMNI [5] are overlay network based multicast schemes which aim to enhance the scalability of data path in P2P streaming network. Patching [18] is also described as a predecessor of Vcast. Patching, Vcast, and Range Multicast offer solutions to on-demand requirement of streaming service. Overcast optimizes overall bandwidth while OMNI optimizes end-to-end latency in building overlay network based multicast trees.

2.1.2.1 Patching On-demand delivery of bandwidth intensive streaming media to a large number of users has two challenges. First, it must immediately service user requests regardless of their arrival time in relation to the playout of the current streaming session. Second, it must use media server's bandwidth effectively since the server must support many users and each streaming session consumes a significant amount of bandwidth. IP multicast offers an efficient group communication mechanism without requiring an additional server bandwidth. Yet, it falls short on the on-demand aspect of the challenge and requires a separate IP multicast session to be established for each late comer. Patching [18] was designed to support on-demand requirement of IP multicast based video streaming service. A new user who wishes to receive a particular video streaming joins a corresponding IP multicast

tree that broadcasts the most recent session. The user receives the on-going video stream from the multicast tree and saves the receiving segments at a local storage area. At the same time, the user establishes a unicast connection, known as a *patching stream*, to a *patching server* to receive the missed segments. The missed segments will be sent from the patching server for an immediate playback. Once all missed segments have been played back, the user starts the playback of the stored content received from the IP multicast tree.

While patching enables efficient on-demand delivery of streaming content, the actual deployment over the Internet is not feasible today because the scheme is based on IP multicast.

2.1.2.2 Vcast Vcast's [17] goal is to provide an on-demand streaming service by adapting the IP multicast based patching scheme to an overlay network based multicast scheme. Overlay nodes in Vcast perform dual-roles: they provide multicast server functionality as well as caching server functionality. As a multicast server, it participates in the building and maintenance of multicast trees. As a caching server, it temporarily stores the forwarding streaming content locally. For subsequent user requests for the same content, *relay nodes* (i.e. overlay nodes in the delivery path of a streaming session) can provide the streaming service on behalf of the media server. This reduces the bandwidth concentration at the media server and improves latency.

The missed segments will be streamed from the media server as a direct unicast connection, similar to patching.

When a new overlay node, X, wishes to join the overlay network, it contacts the media server to learn IDs of all other operating overlay nodes in the network. X contacts each overlay node to let them know of its existence.

A new client, C, wishing to receive a streaming service sends a request message to the server. Sever returns a *representative node*, X, to C, which is chosen in round-robin fashion out of all the existing overlay nodes. C sends X a join message and requests a streaming service for the desired content, V. If X is not a *relay node*, X picks a node Y in round-robin out of all the overlay nodes known to X and forwards the join message to Y. If Y is a relay node, Y returns a setup message to X and sends streaming data back the path as recorded in the join message header. The intent of choosing overlay nodes in a round-robin fashion is

to distribute the load evenly.

2.1.2.3 Range Multicast Range Multicast (RM) [16] aims to provide on-demand video streaming service with VCR-like functionalities without requiring additional bandwidth from the content server. Its architecture is based on the overlay network based multicast where selected network nodes perform multicast functionalities along with specific tasks designed for RM protocol. RM is realized by having each overlay node along the path of a multicast tree cache blocks of streaming segments as its local storage space permits. An important characteristics of this approach is that as the number of clients who join a multicast tree to receive a particular streaming content increases, the total number of video stream segments being cached within the network increases.

A client wishing to receive a video stream sends a request to a nearby RM overlay node called *representative node*. If the representative node is unable to service the client request from its local buffer, it floods the request to other RM overlay nodes in the network. Overlay nodes that can satisfy the request returns a response to the representative node and video stream is forwarded to the client through its near-by RM node. A new multicast session from the root will only be established when there is no other RM overlay nodes that can satisfy the client request.

VCR-like functionalities, such as fast forwarding and backward jumping are realized by requesting needed segments from one of the RM nodes in the distribution tree. They require no additional streaming session to be established from the content server. The only exceptions are when the leading client (i.e. clients receiving segments directly from the root) requests viewing of future segments and the trailing client (i.e. clients receiving segments at the end of a range multicast tree) requests viewing of past segments beyond what's found in the local buffer. Any other clients should be able to find needed segments in one of the RM overlay nodes, regardless of the viewing direction of the video stream.

2.1.2.4 Overcast Overcast [21] is designed to offer an on-demand delivery of non-interactive, bandwidth demanding, video streaming service to a self-similar community of users through overlay network based multicast. The design goal of Overcast is to build single source multi-

cast trees that maximize bandwidth availability from the root to each overlay node without knowing the details of the underlying network topology. It also tries to respond quickly and efficiently to transient network failures and congestions in the underlying network.

Through a centralized lookup mechanism, a client wishing to receive a video streaming service finds a root node of a multicast tree that distributes a desired content. The client requests the root node to help measure the available bandwidth between the client and the root on a direct connection and saves it as a nominal download bandwidth. The client also requests the root node for a list of children nodes and their descendant nodes that are attached to the root node. The client checks the availability of bandwidth from the root through each one of every descendants and determines which overlay nodes can offer the same amount of bandwidth as the nominal download bandwidth. The most distant node, in terms of tree hierarchy, from the root that satisfies the bandwidth requirement will be selected as the node to which the client will attach itself in the multicast tree.

On-demand service is supported by the notion of *archive*. Each overlay node buffers data it forwards in archive and distributes the archival index to participating overlay nodes in a multicast tree. A user can request the starting point, such as the beginning, when it joins an archival group of a particular tree.

Since root node holds both the content as well as the distribution tree information, communication failures (e.g. link down, root node down) can jeopardize the entire distribution system. To cope with this problem, Overcast performs root node replications and forwards the streaming requests to replicated nodes in round-robin fashion. It also employees a series of replication nodes cascaded in front of the root node so that any one of them can overtake the responsibility of the root if necessary. In order to cope with non-root node failures, each child node maintains a list of ancestors so it can attach to a next higher level parent.

2.1.2.5 OMNI *Overlay Multicast Network Infrastructure (OMNI)* [5] is designed to support large-scale, latency sensitive media-streaming applications using overlay network. Since live-media cannot be pre-delivered to different distribution points, maintaining efficient delivery paths are crucial. OMNI's definition of an efficient path is one that minimizes the packet delivery delay. It's goal is to build and maintain a single-source multicast tree that

is optimized for *degree-constrained minimum average latency* using decentralized algorithm. Given a capacity and degree constrains, overlay nodes are organized into overlay network which adapts to changes in client distributions and network load by executing sequence of optimization steps. OMNI assumes that overlay nodes are part of network infrastructure that they do not appear and disappear suddenly under normal operations. Overlay node failures are assumed to be rare incidents.

During the initial attempt to bring up OMNI overlay network, a root node performs a centralized tree building. All overlay nodes measure the latency between the root and themselves and send $joinRequest<latencyToRoot, DegreeBound>$ message to the root. Once $joinRequests$ from the participating nodes are collected, the root constructs the distribution tree in the order of closest to farthest (in terms of latency). The root selects N number of closest nodes as its direct children, with each child having N number of next closest set of children, where N is the $DegreeBound$ of each overlay node. This process repeats until all nodes belong to a cluster. Once all overlay nodes joined, each node performs local transformation procedures periodically to optimize the minimum average latency. OMNI defines five set of procedures, such as child promotion to a higher level, child-parent swapping, and others for the optimization. These operations are executed only after they are determined to reduce the average latency. In addition, OMNI defines a random swapping of children at a certain interval and probability. This is an attempt to achieve global latency minimization since local transformation procedures can only produce local subtree minima.

Those overlay nodes that join after the initial bring-up phase will contact the root node. The root node determines the best placement of the joining node in the distribution tree that results in the minimum average latency in the network. The root node then returns the address of the parent node to which the joining node should attach. A departure of an overlay node is handled by local transformation.

2.2 SEARCH PATH FLEXIBILITY AND SCALABILITY

This section presents overview of schemes that add flexibility and scalability to the search path of a P2P streaming network. The Review is grouped based on unstructured P2P architecture (i.e. Gnutella like) or the DHT based P2P architecture.

2.2.1 Schemes Based on Unstructured P2P Model

Associative Overlay [8], Interest-Based Locality [35], YAPPERS [12], and Search-Index Links [9] are unstructured P2P architecture based search schemes which aim to enhance the flexibility and scalability of search path in P2P network.

Associative Overlay is designed to add scalability to search operation by restricting search to selected peers according to predefine rules. YAPPERS divides the peers into a small number of color coded space and restricts the search within the same color of peers. Interest-Based Locality prioritizes the peers to which the queries should be sent according to the past history. Search/Index Links is designed to give flexibility to search operations by allowing any peer to take on the search proxy server role as well as the client role, depending on the peers to which it establishes neighbor relationship.

2.2.1.1 Associative Overlays Associative Overlays [8] focuses on two desirable services of P2P search: 1) the ability to find rare content efficiently, and 2) to execute partial-match queries. Existing P2P architectures, such as Gnutella [13] and Chord [36], can only support either one or the other of these properties. Associative Overlays tries to develop a decentralized P2P architecture that effectively finds rare content. The proposed scheme is based on an unstructured P2P model, such that it retains many of the desired properties of Gnutella like P2P search, such as partial match queries and high resiliency to failures.

The search on the Associative Overlays is directed only to those peers that are believed to have a relevant information without having to physically cluster peers into separate communities. This search is called *guided search*. The basic premise of the guided search is that peers that had or would have been able to satisfy previous queries are more likely candidates

to answer the current query.

Guided search is driven by a set of rules called *guide rules*, which is created by the originating peer. Guide rule defines some predicates regarding the previously visited peers. A set of peers that belong to same guide rule should contain data items that are semantically similar. Each peer that belongs to a guide rule maintains a small list of other peers that belong to the same guide rule. A guided search query is sent to all the peers that are listed in a guide rule and the query will be forwarded by those peers to their list of peers in their guide rule. During the guided search, each peer collects and analyzes the results of guided search. This statistic is used to refine the guide rule.

2.2.1.2 YAPPERS YAPPERS [12] proposes a peer-to-peer content search scheme that is a hybrid of unstructured and DHT based P2P architectures. Design objectives of the lookup service in YAPPERS are 1) impose no constraints on overlay topology, 2) optimize for partial lookups (i.e. an efficient search among a limited set of peers), 3) contact only those nodes that can contribute to the search result, and 4) minimize the effects of node and link insertion/deletion within a small area of the network.

YAPPERS partitions a large overlay network into many small and overlapping neighborhoods. Two types of neighborhoods are defined: immediate neighborhood (IN) and extended neighborhood. Immediate neighborhood of node A, $IN(A)$, is composed of all nodes within h hops distance from node A in the overlay network. Within $IN(A)$, YAPPERS assigns the nodes into a small number of color space using the hash value of their IP addresses: $HASH(K) \equiv (HASH(IP_x)\ mod\ b)$, where K is the key (e.g. filename), IP_x is the IP address of a node x, and b is the total color space $(C_0, C_{(b-1)})$.

Consider a simplified case where key space is divided into white and gray colors. Every node in an immediate neighborhood is given a color assignment of either white or gray. When a white color node A wishes to store a white key item, it stores it locally. If node A wishes to make a gray key item available, it requests a neighbor gray node to store it. Queries for a particular color key item will be forwarded to respective color nodes only. To guarantee this operation, each node will have a knowledge of all nodes within $(2h + 1)$ hops away, where h is a small constant, such as 3.

Suppose node x wishes to store $<k, v>$. If $HASH(k) \equiv (HASH(IP_y)\ mod\ b) \equiv C(k)$, then node y must store the $<k, v>$ pair. If there are other nodes with $C(k)$ in the immediate neighborhood, x may request any one of those nodes to store the pair. If there is no node with $C(k)$, the color $C_i = C(k)$ is assigned to a node with color $C_{(i+1)} mod\ b$. If multiple such node exists, the node with the smallest IP address will store the pair.

Extended neighborhood includes those nodes that are $(2h + 1)$ hops or more away from node x. Suppose node x requests color $C(k)$ items to be retrieved. It first sends a query to a $C(k) = (HASH(IP_y)\ mod\ b)$ node in $IN(x)$. Subsequently, node y forwards the query to a node $C(k)$ in its extended neighborhood, $EN(y)$. This process is repeated until all $C(k)$ nodes are visited. A unique identity is associated with each query request that a duplicate request will be dropped at each node.

2.2.1.3 Interest-Based Locality Sripanidkulchai, *et al.* [35] propose a content location solution in which peers loosely organize themselves into an interest-based structure over Gnutella network. It is built on the premise that if a peer X has a content that a peer Y is interested in, it is very likely that X will have other items of Y's interests. The relationship between peer X and Y is called *interest-based locality*. Peers that share similar interests create shortcut to one another, called *interest-based shortcut*, which efficiently exploits interest-based locality for content location. Interest-based shortcuts provide a *loose* structure on top of Gnutella's unstructured P2P architecture. The design goal of Interest-based locality is to preserve the simplicity of Gnutllla network and make it scalable so that the amount of queries being flooded in Gnutella network may be reduced significantly.

When a peer first joins the P2P network, a normal Gnutella discovery process is used to discover other peers and their content availability. The newly joined peer randomly selects one peer being discovered out of the query hit list and adds it to a shortcut list. A shortcut list is a fixed-size table that contains entries of interest-based shortcuts. Every node maintains a shortcut list. Each entry in the shortcut list has a rank given to it. Ranks are based on properties associated with a particular shortcut, such as probability of query hit, latency of the path, bandwidth availability, the number of content availability, amount of load, and any combinations of the above. Shortcut list is sorted in ranking order and the

queries are sent on the highest ranked shortcut first, followed by entries in the subsequent ranking order. The entries in the shortcut list are removed as their usage diminishes below a certain threshold.

2.2.1.4 Search/Index Links As the size of unstructured P2P network grows, the load on the network and peers becomes increasingly heavy due to flooded control messages. Some peers can be overwhelmed by the processing of flooded control messages that they can no longer attend to other important services such as downloading user requested files. To minimize such effect, some P2P systems, such as KaZaA [23] and Morpheus [25], allow capable and willing peers (i.e. supper-peers) to provide proxy search service for other peers (i.e. normal-peers) to forward query requests and content indices. However, in this scheme, peers must take on either the server or the client role. [9] examines a way to reduce the load on peers by allowing them to self-organize into a decentralized network where each peer is allowed to make local decision on whether to take on a supper-peer role or normal-peer role for query requests and content indices independently.

Under the proposed scheme, peers are connected by *search links* and/or *index links*. A peer at one end of a search link provides proxy service for the other peer for the exchange of query requests with the rest of the community members. Similar relationship exists for content indices exchange on an index link. Those nodes that receive updates of indices from connected nodes can perform content lookup locally for themselves and for any other queries that are received.

Through *connect*() operation, a peer connects to another peer and establish search and/or index links. Those peers that provide proxy service may use *break*() operation to drop any one of their search and index links to lighten the load when necessary.

2.2.2 Schemes Based on Structured P2P Model

eCAN [41] is a DHT based P2P architecture which aims to enhance the scalability of search path in P2P network. Binning [30] enables a peer to locate a set of near-by peers that host the desired content. Both eCAN and binning are developed as extensions to CAN [31] and

we first describe how CAN works.

2.2.2.1 CAN Content-Addressable Network (CAN) [31] is a hash table based (i.e. a list of $<key,\ value>$ pair where key is generated by a hash function) content management system that can be used to implement large-scale distributed storage systems on the Internet. CAN divides the network space into a d-dimensional Cartesian space (i.e. a d-torus) with one or more CAN nodes in each zone as owner(s). The basic operations performed on a CAN are insertion, lookup, and deletion of $<key,\ value>$ pairs. The key translates to a $<x,\ y>$ coordinate in the Cartesian space and the $value$ is the corresponding content to the key, such as a music file. Many $keys$ fall on the same CAN zone and the CAN node(s) in that zone is responsible for the storage of the corresponding $values$. A CAN node also holds information regarding four-adjacent neighbor zones. This information is used to take over the neighboring zones in case any of the CAN nodes in those zones departs or fails. Requests (i.e. insert, lookup, delete) for a particular key are routed by intermediate CAN nodes towards the destination CAN node whose zone stores the $<key,\ value>$.

CAN requires no form of centralized control, coordination, or configuration. CAN nodes maintain only a small amount of control state information which is independent of the size of the system. CAN's search and insertion operations can route around failure zones.

In spite of its elegance and various merits it brings, CAN faces some important challenges. First, routing is not optimized, as with all other DHT based P2P search schemes, since the logical layout of the CAN nodes in the Cartesian space does not reflect their physical layout. Second, it is intended that each CAN node stores actual values corresponding to the key. The resource requirements at CAN nodes may become very high. Third, the number of hops needed to reach the destination increases as the number of CAN zones increase. In other words, a destination gets farther away as CAN network becomes populated with CAN nodes. This problem poses a serious question on the scalability of CAN.

2.2.2.2 eCAN To cope with the scalability problem of CAN, eCAN [41] was introduced. eCAN is an extension to CAN that enables efficient routing to logically distant destinations. eCan divides the Cartesian space in j^k areas, where j is a constant, $j > 0$, and $0 \leq k \leq m$.

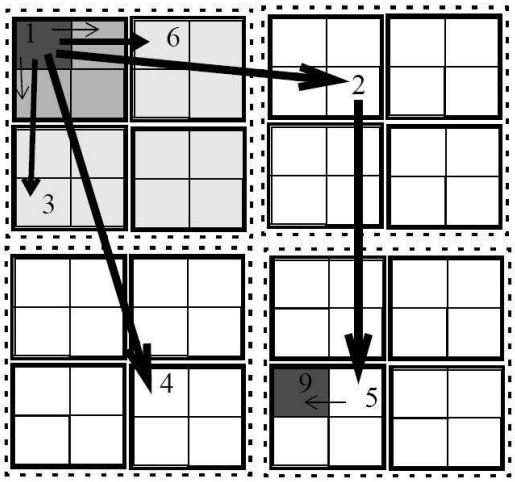


Figure 2.2: A sample eCAN topology and routing paths.

Consider a case where $j = 4$. Each area created by partitioning the search space in 4^m pieces is a CAN zone. A CAN zone belongs to an $order1$ zone. An $order2$ zone is created by dividing the search space in 4^{m-1}. $order2$ zone is overlayed on top of four $order1$ zones. One of the CAN nodes in the four $order1$ zones function as an $order2$ zone leader. This hierarchical clustering of zone organization repeats till $orderm$ zone is created.

eCAN enables efficient routing by establishing *expressways* between neighboring zones in the same order. Figure 2.2 illustrates eCAN with 64 CAN zones. In this example, four neighboring CAN zones make one $order2$ zone. Four $order2$ zones make one $order3$ zone. Suppose CAN node 1 represents an $order1$, $order2$, and $order3$ zones. The routing table of CAN node 1 consists of the default routing table of CAN (represented by the thin arrows in the figure) that links only to node 1's immediate CAN neighbors, and high-order routing tables (represented by the thick arrows) that link to one node in each of node 1's neighboring high-order zones. This figure also illustrates how node 1 can reach node 9 using eCAN routing (1-2-5-9). The expressways help shorten the number of hops required to reach the destination CAN zone.

2.2.2.3 Binning Binning [30] is a landmark based peer positioning scheme to identify a set of server peers that are close to the requesting peer. One of the challenges of P2P discovery service is how to locate the nearest server peers among possibly many and globally spread peers listed at an overlay node. Binning organizes nearby peers in a group known as a *bin*. The peer to bin assignment is done as follows. Each peer measures the distance to several landmark nodes (Binning suggests 8 to 10 landmark nodes to cover the entire Internet) that are scattered throughout the Internet and order the measured results in the closest to the farthest ranking. Those peers that have the same order of landmark nodes will be placed in the same bin. The idea behind this logic is that peers that have the same or similar landmark-distance-order should reside in the same region of the network. When a peer requests a content from a CAN zone, it first measures the distance to each landmark and determines its bin order. This bin order is attached to the query from the peer to the CAN node. The destination CAN node returns only those server peers that have the same or similar bin orders to the requesting peer.

2.3 DISCUSSION

We've reviewed selected literatures that propose streaming service support over P2P network with various emphases and focuses. In this section, we highlight their contributions and discuss the similarities and differences in goals and approaches between our research and theirs.

2.3.1 Evaluation of Schemes that Improve Data Path Scalability

Under application-layer multicast based schemes that aim to improve the data path scalability, we reviewed NICE, ZIGZAG, PeerCast, CoopNet, and OTS+DAC. Under overlay network based multicast schemes, we reviewed Vcast, Range Multicast, Overcast, and OMNI.

NICE [4] and ZIGZAG [38] both focus on the building of a delay optimized end application multicast tree that requires small tree management overhead and bounded node degree.

They both organize multicast tree in hierarchy of clusters whose membership is composed of nearby peers. NICE uses this tree structure to send both the data and control packets in order to avoid the maintenance of two separate routing tables. ZIGZAG, on the other hand, uses this structure for control message exchanges only and it defines separate routing paths for the data packet delivery. ZIGZAG performs better than NICE in some areas, with the cost of maintaining an extra set of routing table, while there was no difference in other areas of overhead costs. The unique data path assignment of ZIGZAG helps peers become more resilient to node failures than NICE which was demonstrated by failure related experiments with 2000 peers.

NICE, ZIGZAG, and our research all try to distribute streaming content to very many users. However, our goals and theirs differ in two major accounts. First, NICE and ZIGZAG try to support *live* streaming applications whereas we mainly focus on VoD service support. In live streaming applications, a subscription of a new user to an on-going session can be achieved simply by adding the user to an appropriate branch of a multicast tree because the network is only responsible for the delivery of future content. The missed segments can be lost forever in live streaming sessions. The problem space where on-demand support is required is more complicated in this aspect and that a mechanism is needed to ensure that the new users will receive the streaming content from the beginning regardless of when they join a distribution community.

Second, NICE and ZIGZAG try to support low bandwidth applications, such as sports tickers and real-time stock updates, while we focus on the distribution of high bandwidth content, such as high quality VoD applications. Since the support of low bandwidth streaming content do not induce large amount of stresses on participating peers, resource concerns are not as sever as the support of high bandwidth streaming content. For example, in both NICE and ZIGZAG, a cluster-leader must be able to support peers ranging from k to $3k-1$ at each level of cluster hierarchy it belongs to. When distributing a high bandwidth content, even a small value of k can saturate the access link of the cluster-leader. This argument can be further extended to make a general statement that the use of end-system multicast may not provide sufficient node degrees when supporting high bandwidth content.

PeerCast [11]'s layered architecture follows the pattern of existing networking protocols

that have successfully demonstrated the establishment of reliable communication paths over unreliable underlying network. PeerCast is unique in that it is the first of its kind to apply layered approach to P2P environment. The simulation results indicated that the layered protocol approach seem to help shielding the unreliable nature of P2P network when serving mid-size P2P multicast group whose members are highly dynamic. However, PeerCast focuses only on *live* media streaming applications — an environment which is quite forgiving of lost packets. The unreliable behavior of peers in the content delivery path are shielded, not avoided or adjusted, by the cost of lost packets. The applicability of PeerCast architecture on non-live streaming applications has not been established yet. Neither does it answer the challenge posed by the on-demand expectation of users when supporting VoD applications.

CoopNet [27] supplements the client-server based content delivery architecture with application-layer multicast to provide resilient streaming service. The resiliency is achieved through the establishment of multiple set of application-layer multicast trees that allow playback of the content at a fraction of the total quality by each independent tree using Multiple Description Coding. This unique approach is noble and promising in an environment where a lower quality content presentation is preferred over loss of or delayed quality content presentation for an exchange with the higher cost of increased data and control overhead. In addition, unlike other application-level multicast schemes, CoopNet tries to support both the live streaming as well as on-demand streaming services. However, the simulated environment in which on-demand service was conducted assumes unlimited availability of access bandwidth at each end node. This assumption voids the applicability of simulation results in our proposed research space where each stream consumes a very large amount of bandwidth that the content server can be overwhelmed with relatively small number of concurrent streams.

OTS_{p2p} and DAC_{p2p} [40] investigates a specific and realistic P2P environment where supplying peers have limited transmit bandwidth availability that a receiving peer needs to open multiple streaming sessions to different supplying peers. Given a typical transmit bandwidth of broadband access service in the United States is less than $1Mbps$, this follows a very realistic arrangement of peers in support of high quality streaming content distributions. For smaller size content distributions, the approach presented in OTS_{p2p} and DAC_{p2p} can be used to answer the following question: "What can be done to make a content distribution

more efficient when a desired content is found at multiple servers?". Peers in OTS_{p2p} and DAC_{p2p} establish connections to multiple servers and download portions of the content in parallel from each source in order to expedite the distribution process. One of the fundamental differences between our research and theirs is in its assumption on the availability of streamed content. OTS_{p2p} and DAC_{p2p} is based on the premise that the total copies of the available streaming content increases as number of people download it increases. In contrast, we seek to design a streaming content distribution network in an environment where streamed content only come from the content servers and users are allowed to buffer only a block of segments for both playback and distribution purpose.

Vcast [17] aims to support on-demand delivery of streaming content through the establishment of overlay network based multicast tree and a unicast connection to the media server. Two improvements made on Vcast over Patching [18] are 1) increased deployability due to no reliance on IP multicast, and 2) reduced media server loading through partial server shadowing at participating overlay nodes (i.e. overlay nodes buffer forwarded content so they can become the root of a multicast tree of a particular content to off-load the media server). In spite of these improvements, a unicast connection which was known as the *patching stream* continues to extend from the media server to the end user. When high quality streaming content are distributed by Vcast, a relatively small number of patching streams can overwhelm the media server.

Range Multicast [16] is similar to Vcast in that the overlay nodes along the distribution path buffer the forwarding content. It differs from Vcast in that it does not establish multicast tree for the data delivery path. In stead, it relies only on the partial server shadowing technique as described in the Vcast section above. Using this infrastructure, Range Multicast tries to support VCR-like functionalities. While Range Multicast demonstrated a feasibility of using overlay network to support VCR-like functionalities, it is questionable whether the overlay nodes have sufficient bandwidth to support the serious demands required by the support of high quality streaming content as it has no means to balance the load on the network. VCR-like functionalities are outside of the scope of our research.

Overcast's goal is to support VoD applications through the overlay network based multicast tree that optimizes the overall network bandwidth. Streamed content are buffered at

participating overlay nodes, similar to Vcast and Range Multicast, and users are directed to one of those nodes to receive the service. Overcast uses a simple protocol to find bandwidth maximized communication paths. The communication paths selection logic, which is based on "the most distant overlay node from the source which provides the same bandwidth as a direct connectivity to the root" is a unique optimization algorithm. However, the search for the most distant node may require a substantial number of measurements when the size of the overlay network becomes very large since every feasible paths (i.e. having the same bandwidth) need to be explored to determine the farthest node(s). While the authors state that the end-to-end delay is a non-goal, it is an issue that should receive an attention. Overcast assumes that the on-demand service is offered to a community of users with similar system capabilities. One of the challenges we try to overcome in our research is how to satisfy the needs of many user with diverse system capabilities. In this regard, our proposed research's operating environment is fundamentally different from the Overcast's.

OMNI [5] builds a delay optimized multicast tree on overlay network for the support of streaming service. It is a counter part to NICE which operates over an application-layer based multicasting. OMNI also differs from NICE in that the root node has the complete view of the network and is the central authority in the management and organization of multicast tree whereas NICE uses a distributed algorithm to organize clusters of peers. In this regard, OMNI lacks robustness and scalability compared to NICE. OMNI assumes that overlay nodes are part of the network infrastructure and remain in service for an extended period of time. An OMNI node failure is assumed to be a rare incident. OMNI's two-stage multicast tree building process (i.e. all nodes joining at once at the initial stage and individual nodes are added later) is also an indicative of assumption that addition and removal of OMNI nodes are infrequent events. Due to these assumptions, OMNI will less likely respond well in situations where the network must respond quickly to dynamically changing environment, such as sudden increase in the user requests beyond perceived capacity. An example of useful OMNI application may be to connect content servers within a CDN. Content servers function as OMNI overlay nodes and organize themselves into a delay sensitive multicast tree to distribute streaming content within the CDN efficiently. Content servers in a CDN fulfills the OMNI nodes assumptions that they are semi-permanently installed, have large

capacity, and seldom fails. However, this is a much different application environment than what our research intends to operate in.

2.3.2 Evaluation of Schemes that Improve Search Path Scalability and Flexibility

Associative Overlays, YAPPERS, Interest-Based Locality, and Search/Index Links were reviewed under the unstructured P2P architecture based content search schemes that enhance the search scalability and flexibility. Under the structured P2P search schemes we reviewed CAN extensions called eCAN and binning.

Associative Overlay [8]'s goal is to design a search scheme that supports the partial match queries and the efficient rare content discoveries both at the same time, which is not possible by the existing unstructured and structured P2P search schemes. *Guided search* can be viewed as a middle ground between the blind search and the routed search. Since the routing decision is based on the the assumption that peers that had been able to satisfy previous queries are more likely candidates to answer the current query, its effectiveness is determined by the degree of correlation between the past query results and the current query. If this assumption holds, guided search can help eliminate unnecessary queries to many unrelated peers. One of the drawbacks of this scheme is that guided search can diminish the chance to uncover other peers that may offer superior service (e.g. higher content availability, greater bandwidth, shorter delay). While guided search may improve one aspect of unstructured P2P search scheme, it does not address other shortcomings of blind search. For example, Associative Overlay follows the normal unstructured P2P search process and uses flooding to discover new peers. This makes Associative Overlay unsuitable for global search due to limited number of hops the discovery queries can travel in order to contain the amount of flooded messages.

YAPPERS [12] proposes a noble search scheme based on the hybrid of structured and unstructured discovery schemes. YAPPERS organizes a *neighborhood* of peers using unstructured discovery scheme and supports an efficient content search within a neighborhood using DHT based discovery. The beauty of YAPPERS is that since every peer organizes its

own neighborhood according to a set of rules that every node follows, the network can be seen and treated as a collection of identical and overlapping cell structure. A search request originated at a particular neighborhood can be forwarded to and processed at any other neighborhoods by using the same procedure.

While YAPPERS' DHT based content storage assignment enables efficient content discovery within a local neighborhood, it brings other challenges that need to be addressed. First, YAPPERS is not design to locate globally scattered content efficiently since the search requests are propagated in hop-by-hop manner from one neighborhood to another. Second, it may not be able to support live streaming service since live streaming media is not generally considered something that can be copied from one node to another node prior to its broadcast. Third, physically copying content from one peer to another can be costly for large size content, especially considering the fact that peers generally have limited bandwidth availability and each peer may not be around for a long time. Fourth, a hash function based content location assignment can cause uneven distribution of content among the neighborhood peers and that the resources within a neighborhood may not be used effectively. Fifth, since the DHT based content assignment does not consider the end user system differences, not all peers in a neighborhood may be able to provide adequate level of support required for the distribution of the assigned content. These issues will significantly impair the operation of YAPPERS when trying to support the distribution of high quality streaming content.

Interest-based locality [35] seeks improved scalability in Gnutella network by directing queries to a list of peers that satisfied queries in the past. Queries are flooded only after none of the peers in the list can return the requested content. The concept is similar to Associative Overlay [8], but the interest-based shortcuts are much more simple and easy to implement. However, its usefulness is limited to those peers that are one-hop away which had been uncovered through flooding in one of the queries in the past. The proposed scheme gives a slight improvement over the existing Gnutella network but it does not solve fundamental issues associated with the unstructured P2P search.

Search/Index Links (SIL) [9] offers a simple and flexible solution to the supper-peer concentration/overloading problem of KaZaA and Morpheus. It also decouples the search query exchange service and the content index exchange service, which are currently offered

as one unit of service by supper-peers. SIL improves the supper-peer scalability and gives resiliency to the query paths. While SIL gives important enhancements to KaZaA and Morpheus, does not address many other fundamental issues associated with the unstructured P2P search, such as how to find rare content more effectively and how to reach those peers many hops away.

eCAN [41] overcomes the hop count increase problem of CAN as the number of CAN zones increases. eCAN groups adjacent CAN zones in layers of hierarchy to create larger CAN zones in increasing order. The shortcuts between zones at each layer are called *expressways* which reduce the number of hops to reach the destination CAN zone. The approach is elegant and effective, though it comes with an increased overhead cost. The proposed scheme can only be applied to CAN. However, other DHT based P2P overlay networks provide a similar search cost performance as eCAN without special enhancements. Expressways supplement the deficiency of CAN, rather than putting CAN ahead of game with other DHT based schemes.

Binning [30] is a simple yet effective mechanism to lead a requesting peer to a set of nearby server peers. Though it was developed by CAN researchers, the concept is applied to any DTH based P2P overlay networks. While the goal of binning is not to lead a requesting peer to "the closest" server peer, but to find nodes in proximity, it is possible to find the closest peer by a simple modification to the logic. Binning relies on the presence of fixed and perpetual landmarks. The placement of landmarks also plays an important role in the accuracy and evenly distributed assignments of peers to bins. Due to the way in which binning uses the landmark measurement information, addition or migration of landmarks may be difficult to handle. Since network grows and shrinks in time, it is inevitable that addition and/or migration of landmarks will be needed in some future time. There is no simple way to handle these events other than re-measuring and re-labelling the each peer in the network

2.4 CONCLUSION

This section provided selected literature review on peer-to-peer streaming systems that try to achieve data path scalability and search path scalability.

Schemes that aid in the data path scalability are further divided in two groups depending on their multicasting approach. One group develops their schemes based on application layer multicast while the other group builds their schemes on overlay network based multicast.

Under application layer based multicast schemes, we reviewed NICE, ZIGZAG, PeerCast, CoopNet, and OTS_{p2p} & DAC_{p2p}. NICE and ZIGZAG aim to support live-streams with small payload to a large number of users. PeerCast's goal is to support live-streams among dynamically changing peers. CoopNet enhances resiliency of streaming service by sending multiple streams that are independently playable. OTS_{p2p} & DAC_{p2p} considered an environment with limited Tx bandwidth capabilities at supplying peers.

Under overlay network based multicast schemes, we reviewed Patching, Vcast, Range Multicast, Overcast, and OMNI. Patching, Vcast, and Range Multicast proposed solutions to on-demand requirement of streaming service. Network-wide bandwidth optimization technique as well as end-to-end latency optimization scheme were discussed in Overcast and OMNI respectively.

We reviewed schemes that add flexibility and scalability in the search path of a P2P streaming network. Reviewed schemes are grouped into two types based on their underlying P2P architecture.

Schemes based on unstructured P2P architecture included Associative Overlay, Interest-Based Locality, YAPPERS, and Search/Index Links [9]. Associative Overlay adds scalability to search query by restricting search to selected peers according to guide rules. YAPPERS partitioned search space into many and overlapping neighborhoods that are logically identical in their construct so that query in one neighborhood can be easily extended to another neighborhood. Interest-Based Locality keeps track of search results and use this information to prioritize the peers to whom the next query should be directed to. Search/Index Links adds flexibility to the designation of supper-peer and normal-peer relationship among neighbor peers.

Under structured P2P architecture based search schemes, we reviewed CAN, eCAN, and Binning. CAN uses a form of distributed hash table to store and locate <*key*, *value*> pair in a d-dimensional space. CAN search scheme has a scalability problem that the number of hops required to route to the destination increases as the number of CAN zones increase. eCAN overcomes this limitation by employing expressways which bypasses many intermediate CAN nodes. Binning enabled a requesting peer to find nearby server peers.

While some noble ideas were proposed in the reviewed literatures, none was able to provide the complete solution to the problems we try to solve. Specifically, we seek to enable an on-demand distribution of high quality streaming content to a large number of dynamic and diverse users. Schemes reviewed in this section share similar goals as our research in some aspects, yet we differ in various other ways. For instance, most schemes that try to improve the data path scalability support live-streaming applications only. When VoD support is considered, they are not designed to distribute high quality movies or to a very large and dynamic user community. In addition, reviewed schemes all rely on multicasting to achieve data path scalability. This approach is acceptable for small- to mid-sized content distribution, yet it does not resolve the bandwidth shortage problem at forwarding nodes' access link when distributing high quality streaming content.

Schemes that aim to provide scalable and flexible search paths we reviewed generally focus on achieving overhead reductions by restricting peer selections based on certain assumptions. These assumptions lead to actions such as contacting the peers that satisfied previous requests or seem to have similar interests as the requesting user. While these actions may lead to a better hit ratio and reduced overhead among certain groups of users, they do not solve may other issues fundamental to P2P search, such as finding globally spread peers.

There is a need and a research opportunity for designing a content distribution network that supports high quality VoD applications to a large and diverse users f Internet.

3.0 ARCHITECTURE DESIGN

This chapter presents the architectural overview of Virtual Theater Network, identifies its components, and describes their functionalities, and presents the formal definition of the model.

Virtual Theater Network is a network model designed to support a large-scale, on-demand, peer-to-peer, stored-video streaming service over the Internet. It is based on a hybrid architecture between a traditional client-server model and an emerging peer-to-peer computing paradigm. Central to this model is a set of *Virtual Theaters*. A Virtual Theater provides a means to mass distribute video streams to users in a certain geographical area, similar in function to that of movie theaters and video rental shops. Within each Virtual Theater, there exists a content distributor, known as *VT Distributor*, which receives video feeds from content providers (e.g. Disney, MGM, Paramount). A VT Distributor organizes one or more *VT Rooms* to service local video distribution needs. A *VT Room* is a group of peers who form a P2P community to receive and distribute a video stream at a specific playback rate and in a specific encoding format. A cyber-cinema created by a VT Distributor and a set of VT Rooms constitute a Virtual Theater.

3.1 ARCHITECTURE COMPONENTS

The proposed streaming distribution network model consists of a two-tier hierarchy of network, connecting three types of network nodes, forming one virtual network space.

- Types of Network Nodes

  - Content Producer
  - Content Distributor (VT Distributor)
  - Community of Content Consumers (VT Room)
- Virtual Network Space
  - Virtual Theater
- Connectivity Between the Nodes
  - Content-Producer-to-Content-Distributor Network
  - Content-Distributor-to-Content-Consumer Network (Virtual Theater Network)

A brief description of each entity follows.

Content Producer is a content server which owns the copyright of streaming content (e.g. MGM, Disney, Paramount). It is the origin of the streaming distribution service and occupies the highest level of the distribution hierarchy. A limited number of content producers are assumed to exist over the Internet.

Content Distributor (VT Distributor) is a media server which has the right to distribute content from the content producers. A VT Distributor distributes streaming videos from multiple content producers. Many VT Distributors, both independent and affiliated, exist throughout the Internet. They provide a means to mass distribute streaming content to users in different geographical areas. VT Distributors are committed to remain in service for extended period of time. Each VT Distributor organizes *Virtual Theater Rooms* (VT Rooms) to distribute a streaming video in each room to service local consumer demands.

A Community of Content Consumers (VT Room) is a group of users who form a P2P community to receive and distribute a video stream among them. It is also known as a *Virtual Theater Room* (VT Room). Each VT Room is managed by a VT Distributor and receives the streaming feed from it. For "popular" streaming content there will be many VT Rooms with members in close proximity, organized by different VT Distributors throughout the Internet. Only one streaming content is distributed in each VT Room.

Virtual Theater is a cyber-cinema created by a VT Distributor and a set of VT Rooms. A Virtual Theater provides a means to mass distribute video streams to users in a certain geographical area of the network, similar in function to that of movie theaters and video rental shops.

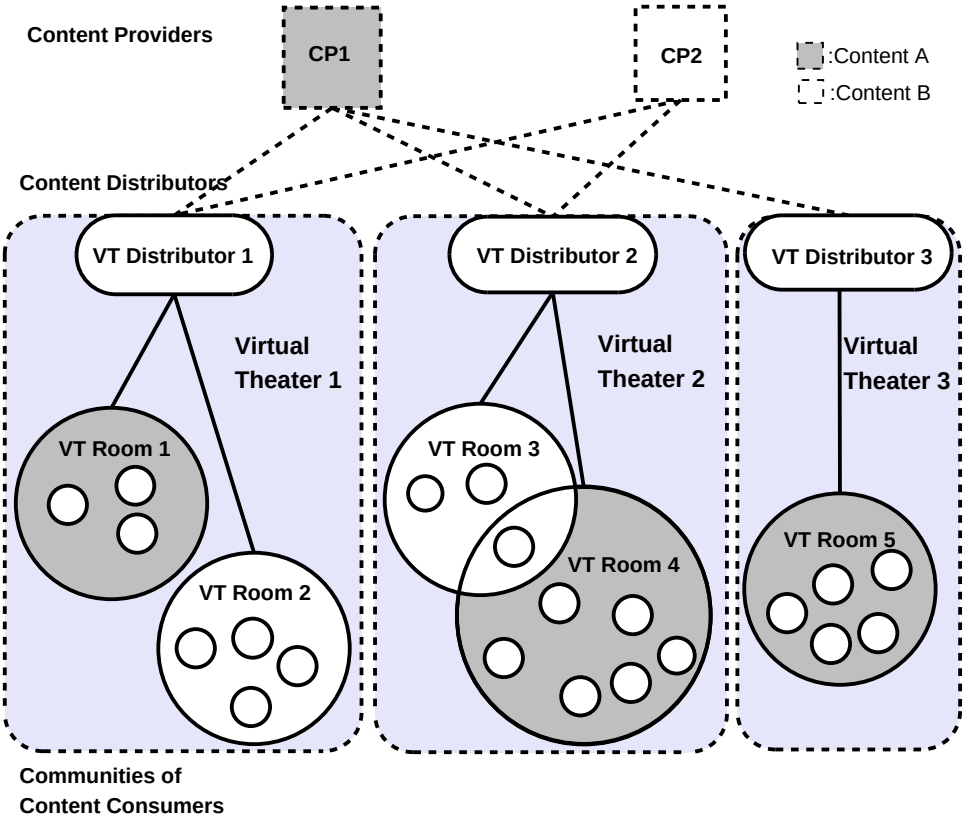


Figure 3.1: Virtual Theater Network

Content-Producer-to-Content-Distributor Network is the higher-tier network and supports *business-to-business* (B2B) video distribution service. It offers video feeds from content providers to content distributors. The design of this network is outside of the scope of this research. While it is an important part of distribution service as a whole, this network can be created separately without affecting the lower-tier network. Since we assume a limited number of content producers and many but stable content distributors that are committed to be in service for a long-term, the existing video distribution mechanisms, such as *Content Delivery Networks* (CDNs), can be used to support this service. In contrast, the problem on the lower-tier network is much more difficult and interesting since existing solutions only address part of the issues and more complete solution is needed.

Content-Distributor-to-Content-Consumer Network (Virtual Theater Network) is the lower-tier network which provides *business-to-consumer* (B2C) streaming video distribution service. It enables the distribution of streaming video from a content distributor (i.e. Virtual Theater) to a group of content consumers (i.e. users in a VT Room). This network along with its supporting network entities and control services is called the *Virtual Theater Network*. The focus of our research is to design the video distribution and discovery schemes for the Virtual Theater Network.

Figure 3.1 illustrates the Virtual Theater Network model. In this example, two content providers, *CP1* and *CP2*, supply video feeds to a set of content distributors. CP1 provides a video feed to VT Distributors 1, 2, and 3. CP2 provides a video feed to VT Distributors 1 and 2. VT Distributor 1 and 2 manage two instances of VT Rooms (i.e. VT Rooms 1, 2, and 3, 4 respectively) while VT Distributor 3 only manages one VT Room (i.e. VT Room 5). Small circles within each VT Room represent peers. With the support from a respective VT Distributor, users of each VT Room work together to sustain the distribution of a streaming video.

The keystone of our proposed streaming service architecture is a dynamic and distributed resources available at the members of a VT Room. To take advantage of this architecture, a video stream is divided into blocks of segments and dispersed at the caches of user systems at various locations in a VT Room. Users playback the video by locating and retrieving the segments in their playback sequence. As video segments are being downloaded, the user

makes them available for others to access.

Initially, when a VT Room is organized, users receive streaming feed from the VT Distributor. Once a sufficient number of cached and replicated video segments become available within the community, the distribution of streaming content becomes self-sustainable. Only those occasions where the needed video segments are unavailable within a VT Room, will the user request the missing segments from the VT Distributor.

The main challenge in the design of video distribution service based on the proposed architecture is twofolds: 1) how to organize dynamically changing video segment availability information over the network to provide effective video segment advertisement and discovery service, and 2) how to manage the reception of video segments that ensures the orderly and timely delivery of video segments and contributes to the self-sustainability of a VT Room.

3.2 FORMAL DEFINITION OF VIRTUAL THEATER NETWORK

Formally, the Virtual Theater Network's hierarchical structure and the relationship among its components are defined as follows:

Let $VTD(t)$ be the set of all VT Distributors in the network at time t.

$$VTD(t) = \{VTD_i,\ i = 1, 2, \ldots\}$$

Each VT Distributor has one or more streaming videos in its local storage and makes them available for distribution. Let $C_i(t)$ be the set of streaming content available at VTD_i at time t. Then, the set of all content that are available and may be distributed by all VT Distributors throughout the network is given by:

$$C(t) = \bigcup_{i \in VTD(t)} C_i(t)$$

The content distributed at each VT Distributor may not be unique. In deed, we expect that many of the content will be available at multiple VT Distributors.

$$\exists i, \exists j \mid C_i(t) \bigcap C_j(t) \neq \emptyset$$

VT Distributor provides video streaming through one of its managing VT Rooms. Let VTR_i^j be the j^{th} VT Room being managed by VTD_i. Then, VTD_i and the set of all VT Rooms that VTD_i is responsible for at time t constitute a Virtual Theater, $VT_i(t)$ and is given by:

$$VT_i(t) = \{VTR_i^j,\ j = 1, 2, \ldots\}$$

A VT Room is defined as, in a most generic sense, a set of users or a group of peers, P, that participate in the reception and distribution of segmented streaming video and whose membership change in time.

$$VTR_i^j(t) = \{P_n,\ n = 1, 2, \ldots\}$$

A streaming video, V, is divided into blocks of segments, S_m, and each segment is composed of blocks of frames, F_i.

$$V = (S_m,\ i = 1, 2, \ldots N)$$

$$S_m = (f_l,\ l = 1, 2, \ldots F)$$

On this hierarchical structure of Virtual Theater Network, two main services are offered to facilitate an efficient, scalable, and versatile on-demand peer-to-peer video distribution service: *Video Segment Reception Management* and *Video Segment Advertisement and Discovery Service*.

3.3 TARGET OPERATING ENVIRONMENT

The proposed video distribution network architecture, with its accompanying video segment discovery and distribution schemes, are designed with a specific operating environment. It is suited for an environment where a moderate to high rate of segment distribution requests

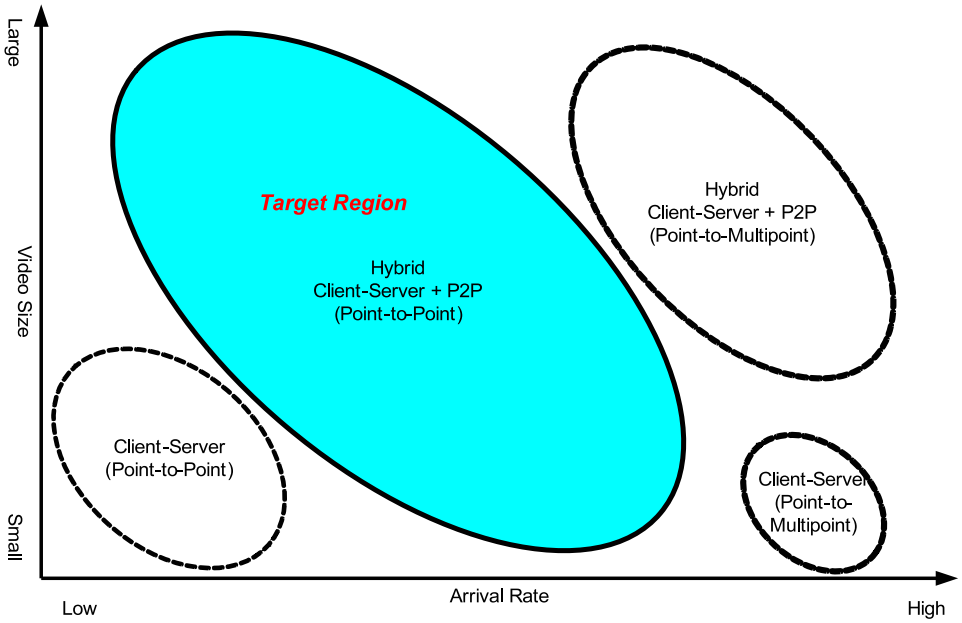


Figure 3.2: Target Operating Environment

arrive to the media server. The schemes are also designed for the distribution of moderate- to large-size video. The colored oval in Figure 3.2 depicts this region in two-dimensional (streaming request arrival rate vs. video size) space. For the remaining regions of the map, other approaches may be more suitable. For example, when distributing a small volume of video to a small number of users, a traditional client-server network works best for its simplicity. As the rate of streaming requests increases as well as the volume of video stream increases, the proposed distribution model becomes more practical video distribution approach. Once the streaming requests reach an extremely high rate, processing of streaming requests in batches becomes feasible and point-to-multipoint (P-MP) based streaming service is considered more practical. Under such environment, when the volume of video stream is very small, client-server based P-MP streaming service would provide adequate support. When the size of video is not very small, peer-to-peer based P-MP service is more suitable.

In the following chapters, we describe the design of video segment discovery scheme, *Virtual Chaining*, and video segment reception management scheme, *Sliding Batch*, and its

variant, *Restrained Sliding Batch*. Virtual Chaining and Sliding Batch work hand-in-hand to achieve segmented video distribution service within a VT Room. The details of Sliding Batch is given first followed by the descriptions of Virtual Chaining.

4.0 VIDEO SEGMENT RECEPTION MANAGEMENT

This chapter describes *Sliding Batch*, a video segment reception scheme used in a VT Room. It allows users of a VT Room to retrieve a video stream as a sequence of small video segments from multiple distribution sources. It allows users with excess receive bandwidth and buffer space to prefetch future segments at rates below the nominal streaming rate. This enables users with limited transmit bandwidth to become contributing sources and helps ease the load on the VT Distributor. The scheduling algorithm defined in Sliding Batch determines the timing and the rate of each video segment reception to ensure an orderly and timely video playback. This chapter provides the definition of the key concepts in Sliding Batch and describes how they relate to the management of the available receive bandwidth and buffer space to determine the scheduling of segment receptions.

4.1 DEFINITION OF KEY CONCEPTS

The video segment reception scheme in Sliding Batch is expressed in terms of *segments*, *epochs*, and *batches*. This section provides their definitions.

4.1.1 Segments

A video stream is a continuous flow of a sequence of compressed video frames transmitted over a network so that the recipient may playback the video frames as they arrive. In Sliding Batch, a block of a sequence of video frames makes up a *Segment*, S_i. In turn, a sequence of segments constructs a video, V. S_i is a logical unit in V, similar to a chapter in a DVD, and

may vary in size and length. The number and the sizes of segments in a video are VT Room specific parameters. The segment is also a unit of blocks of video frames being exchanged among the users of a VT Room.

Segment S_i is characterized by its sequence position, i, in V, a sequence of frames that belongs to S_i, its starting playback time, $\alpha(S_i)$, and a batch it belongs to, $\beta(e_j)$.

$$V = (S_i, i = 1, 2, \cdots, N)$$

$$S_i = \begin{cases} (f_k, k = 1, 2, \cdots, n_i,\ n_i \leq F) & \text{if } i = 1 \\ (f_k, k = n_{i-1}+1, n_{i-1}+2, \cdots, n_i,\ n_i \leq F) & \text{if } 2 \leq i \leq N \end{cases}$$

$$S_i \cap S_j = \emptyset$$

$$\alpha(S_i) = \begin{cases} t_0 & \text{if } i = 1 \\ \alpha(S_{i-1}) + \delta(S_{i-1}) & \text{if } 2 \leq i \leq N \end{cases}$$

$$S_i \in \beta(e_j)$$

where N is the total number of segments in V, t_0 is the time the user joined the VT Room and began playing back the first segment, S_1, and $\delta(S_i)$ is the playback duration of S_i.

Let $|V|$ and $|S_i|$ be the size of V and S_i respectively. Let $\delta(V)$ be the total duration of video playback time. Then, η, the nominal streaming rate of a video is given by:

$$\eta = \frac{|V|}{\delta(V)}$$

$$|V| = \sum_{i=1}^{N} |S_i|$$

Accordingly, the playback duration of S_i, $\delta(S_i)$, is defined as

$$\delta(S_i) = \frac{|S_i|}{\eta}$$

While every segment may have the same number of frames, each frame may have a different size depending on the mode of encoding being used. If it is encoded in CBR mode, the size of each frame will be the same, whereas in VBR mode, each frame may have a different number of bits. In practice, a streaming video is encoded using CBR mode to

facilitate the steady flow of data required for a smooth playback. Sliding Batch, as described in this paper, assumes a CBR mode of encoding.

The detail of $\beta(e_j)$ in relation to segments is given later.

4.1.2 Epochs

In Sliding Batch, a life-time of a video streaming is divided into a sequence of time intervals, known as *epochs*. There are N epochs in a V and their durations may vary from epoch to epoch. Both the number and the durations of epochs in a video are VT Room specific parameters. An epoch, e_i, is characterized by its starting epoch time, $\alpha(e_i)$, its duration, $\delta(e_i)$, and its associated batch, $\beta(e_i)$. An e_i is closely related to the playback property of S_i as shown below:

$$\alpha(e_i) = \alpha(S_i)$$

$$\alpha(e_{i+1}) = \alpha(e_i) + \delta(e_i)$$

$$\delta(e_i) = \frac{|S_i|}{\eta} = \delta(S_i)$$

Each epoch is associated with a batch, $\beta(e_i)$, and the detail of their relationship is given next.

4.1.3 Batches

A *batch*, $\beta(e_i)$, is a set of segments whose downloading is initiated at the same time at the beginning of e_i. There are total of N batches in a video and each batch consists of a set of segments that are unique to itself, except for those batches with an empty set of segments. $\beta(e_i)$ is characterized by an associated epoch, e_i, a set of associated video segments, and a set of streaming sessions that are initiated at epoch e_i with rate r_j each.

$$\beta(e_i) = \begin{cases} \{S_j, j = 1, 2, \cdots, n_i, \quad n_i \leq N\} & \text{if } i = 1 \\ \{S_j, j = n_{i-1}+1, n_{i-1}+2, \cdots, n_i, \quad n_i \leq N\} & \text{if } 2 \leq i \leq N \text{ and } n_{i-1} < N \\ \emptyset & \text{otherwise} \end{cases}$$

$$\beta(e_i) \cap \beta(e_j) = \emptyset$$

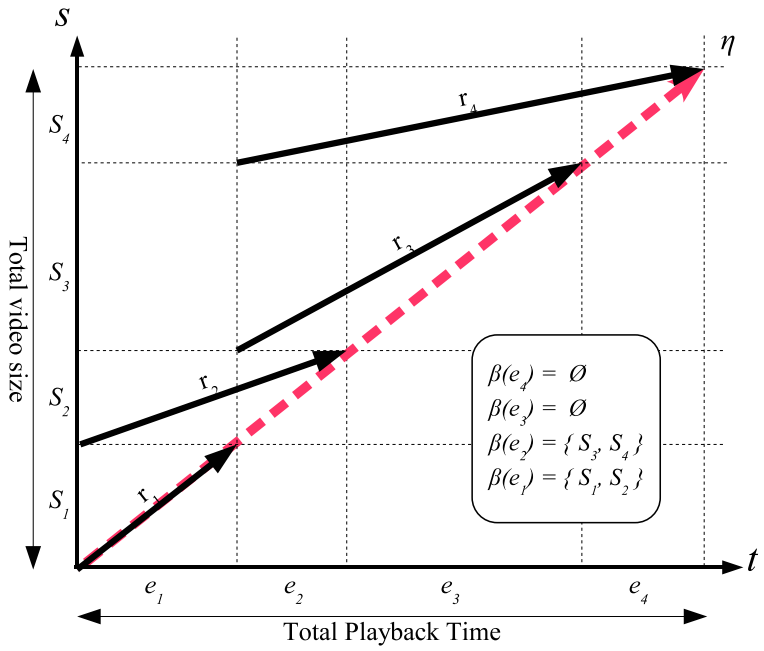


Figure 4.1: Relationship among segments, epochs, and batches

Segments that belong to the same batch all start downloading at the same time at the beginning of the associated epoch.

$$A(S_j) = \alpha(e_i), \quad \forall S_j, \ S_j \in \beta(e_i)$$

where $A(S_j)$ is the starting download time of S_j.

The ending download time of segment j, $\Omega(S_j)$, differs from segment to segment and it is the ending playback time of segment j.

$$\begin{aligned} \Omega(S_j) &= A(S_j) + \frac{\|S_j\|}{r_j} \\ &= \alpha(e_j) + \delta(e_j) \end{aligned}$$

where $\|S_j\|$ is the size of S_j in number of bits and r_j is the rate of S_j being downloaded. r_j is given by:

$$r_j \quad = \quad \frac{\|S_j\|}{\Omega(S_j) - A(S_j)}, \quad S_j \in \beta(e_i)$$

Figure 4.1 illustrates the relationship among segments, epochs and batches in a simplified video reception scenario. In this example, a video is divided into $N = 4$ segments of varying lengths. Batch $\beta(e_1)$ consists of segments S_1 and S_2. Segment downloading for $\beta(e_1)$ was initiated at time $\alpha(e_1)$ for both S_1 and S_2 at rates r_1 and r_2 respectively. Batch $\beta(e_2)$ consists of segments S_3 and S_4. Segment downloading for $\beta(e_2)$ was initiated at time $\alpha(e_2)$ at rates r_3 and r_4 respectively. No segment is associated with $\beta(e_3)$ or $\beta(e_4)$.

4.2 SCHEDULING OF SEGMENT RECEPTIONS

This section presents the scheduling algorithm used in Sliding Batch. It describes how a batch size relates to the scheduling of segment downloads and how the available receive bandwidth and buffer space are incorporated into the batch size determination process.

4.2.1 Batch Size Determination

One of the important parameters of Sliding Batch is the number of segments that belong to each batch, or the size of a batch, $|\beta(e_i)|$. To determine the size of a batch means to decide on which set of segments will begin downloading simultaneously in that same batch and at what rates. In other words, it serves as a key factor in the scheduling of segment downloads. The size of each batch is a user specific parameter and is driven by the user's available receive bandwidth and buffer space.

Lemma 4.2.1. *Let $\overline{N}(e_i)$ be the number of remaining segments yet to be received by a user at time $\alpha(e_i)$. Let $|\beta_R(e_i)|$ be the "rate-limited" batch size at time $\alpha(e_i)$. It is the maximum number of segments a user may begin downloading in a batch given the available receive bandwidth is the only limiting factor to be considered in determining the batch size. Let $|\beta_B(e_i)|$ be the "buffer-limited" batch size at time $\alpha(e_i)$. It is the maximum number of*

segments a user may begin downloading in a batch given the available buffer space is the only limiting factor to be considered in determining the batch size. Then, the size of a batch $|\beta(e_i)|$ *is given by:*

$$|\beta(e_i)| = min(\ \overline{N}(e_i),\ |\beta_R(e_i)|,\ |\beta_B(e_i)|\ )$$

Proof. By applying the commutative law, consider the $min(\ |\beta_R(e_i)|,\ |\beta_B(e_i)|\ )$ portion of the above equation first. It states that the lesser of the two values, the"rate-limited" batch size or the "buffer-limited" batch size, will be further considered. In other words, the smaller batch size dictated by the availability of receive bandwidth or buffer space will be used in the next stage of *min()* operation. The selected smaller batch size value will be further capped by $\overline{N}(e_i)$, the number of remaining segments yet to be received at time $\alpha(e_i)$. The final value constitutes the total number of segments a user may begin receiving concurrently at the beginning of the associated epoch. The formal definitions of $\overline{N}(e_i)$, $|\beta_R(e_i)|$, and $|\beta_B(e_i)|$ are given below. □

The number of remaining segments, $\overline{N}(e_i)$, is defined as:

$$\overline{N}(e_i) = \begin{cases} N & \text{if } i = 1 \\ N - \sum_{k=1}^{i-1} |\beta(e_k)| & \text{if } 2 \leq i \leq N \end{cases}$$

The rate-limited batch size, $|\beta_R(e_i)|$, refers to the size of a batch being computed based solely on the available receive bandwidth, R_A, and is determined by the maximum number of concurrent segment downloading sessions that can be sustained given the available receive bandwidth at time $\alpha(e_i)$.

$$|\beta_R(e_i)| = \begin{cases} m_1, & \text{if } i = 1 \\ \quad \exists max(m_1) | \sum_{k=1}^{m_1} r_k \leq R_T, \quad m_1 \leq N & \\ m_i - m_{i-1}, & \text{if } 2 \leq i \leq N \\ \quad \exists max(m_i) | \sum_{k=m_{i-1}+1}^{m_i} r_k \leq R_A(e_i), \quad m_i \leq N & \end{cases}$$

Similarly, the buffer-limited batch size, $|\beta_B(e_i)|$, is computed based solely on the available buffer size, B_A, at time $\alpha(e_i)$, as if there were infinite amount of receive bandwidth available.

$|\beta_B(e_i)|$ is determined by the maximum number of concurrent segment downloading sessions that can be sustained given B_A at $\alpha(e_i)$.

$$|\beta_B(e_i)| = \begin{cases} m_1, & \text{if } i = 1 \\ \quad \exists max(m_1) | \sum_{k=1}^{m_1} \|S_k\| \leq B_A(e_1),\ m_1 \leq N & \\ m_i - m_{i-1}, & \text{if } 2 \leq i \leq N \\ \quad \exists max(m_i) | \sum_{k=m_{i-1}+1}^{m_i} \|S_k\| \leq B_A(e_i),\ m_i \leq N & \end{cases}$$

4.2.2 Receive Bandwidth Management

The receive bandwidth available at the beginning of the first epoch, $R_A(e_1)$, is defined as the total receive bandwidth, R_T, that can be set aside for the support of the streaming service. The available receive bandwidth for the beginning of the subsequent epoch is determined by how much bandwidth has been consumed in the previous epoch, $R_U(e_{i-1})$, and how much bandwidth has just been added due to the release of a segment reception session, r_{i-1}.

$$R_A(e_i) = \begin{cases} R_T & \text{if } i = 1 \\ R_T - R_U(e_{i-1}) + r_{i-1} & \text{if } 2 \leq i \leq N \end{cases} \tag{4.1}$$

The used received bandwidth, $R_U(e_i)$, during epoch e_i is given by

$$R_U(e_i) = \begin{cases} \sum_{k=1}^{m_1} r_k, & \text{if } i = 1 \\ \quad \exists max(m_1) | \sum_{k=1}^{m_1} r_k \leq R_T,\ m_1 \leq N & \\ R_T - R_A(e_i) + \sum_{k=m_{i-1}+1}^{m_i} r_k, & \text{if } 2 \leq i \leq N \\ \quad \exists max(m_i) | \sum_{k=m_{i-1}+1}^{m_i} r_k \leq R_A(e_i),\ m_i \leq N & \end{cases} \tag{4.2}$$

Figure 4.2 depicts a sample segment downloading scenario under the rate-limited batches. In this example, the video is divided into 24 equal-length segments. A sufficient amount of segment downloading buffer is available to allow storing of all 24 segments simultaneously. The total available receive bandwidth is limited to twice as much as the nominal streaming rate. Under this condition, six batches are needed to begin retrieving all 24 segments. The rate of segment downloading varied from the nominal rate to as little as one-eighteenth of the nominal rate.

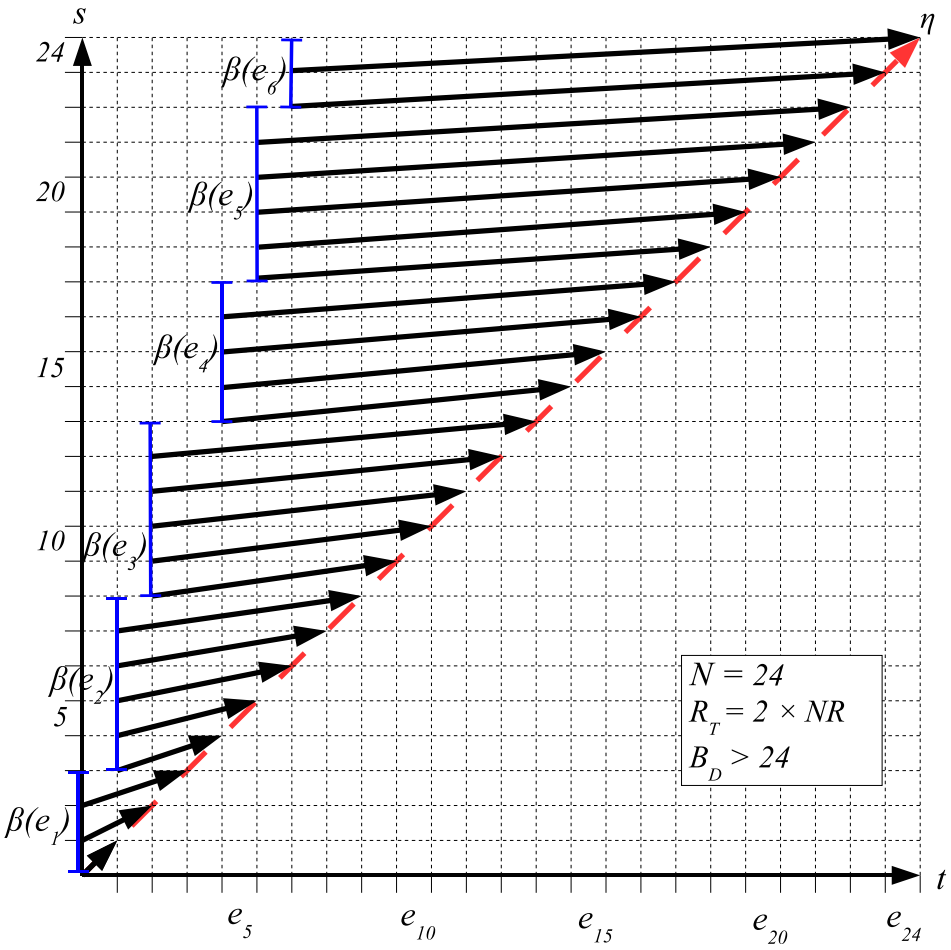


Figure 4.2: Sample segment retrievals under the rate-limited batches

4.2.3 Buffer Management

The total buffer space, B_T, consists of two buffer areas: the downloading buffer area, B_D, and the post-playback buffer area, B_H. The downloading buffer area is used as a temporal storage space to buffer segments that are being downloaded and played back. The size of downloading buffer, $|B_D|$, determines the maximum number of segments that can be downloaded at the same time.

The post-playback buffer area is used to retain segments that have finished playing back. The size of post-playback buffer, $|B_H|$, determines the duration of time the segments will be held in a user system after their playback. The total buffer size, $|B_T|$, downloading buffer size, $|B_D|$, and post-playback buffer size, $|B_H|$, have the following relationship:

$$|B_T| = |B_D| + |B_H|$$

For a simplicity of operation, a fixed size buffer space is allocated for B_D and B_H.

Snapshots of a conceptual view of a sample buffer space usage with $|B_T| = 10$, $|B_D| = 7$, and $|B_H| = 3$ during e_1 through e_5 are depicted in Figure 4.3. At e_1, segments S_1 through S_7 begins occupying B_D. B_H is empty. At e_2, S_1 finishes playing back and is logically moved to B_H. S_8 begins downloading and uses space in B_D. At e_3, S_2 finishes playing back and is logically moved to B_H. S_1 and S_2 are in B_H. S_9 begins downloading and uses space in B_D. The same process repeats at e_3. At e_4, S_1 is being dropped from B_H. S_4 finishes playing back and is logically moved to B_H. S_11 begins downloading and uses space in B_D. The process repeats at e_5.

Let $\|B_D\|$ be the size of downloading buffer area in number of bits. The available buffer space, $B_A(e_i)$, at time $\alpha(e_i)$ is given by

$$B_A(e_i) = \begin{cases} \|B_D\| & \text{if } i = 1 \\ \|B_D\| - B_U(e_{i-1}) + \|S_{i-1}\| & \text{if } 2 \leq i \leq N \end{cases}$$

The used buffer space, $B_U(e_i)$, during epoch e_i is given by

Figure 4.3: A sample buffer space usage

$$B_U(e_i) = \begin{cases} \sum_{k=1}^{m_1} \|S_k\|, & \text{if } i = 1 \\ \quad \exists max(m_1) | \sum_{k=1}^{m_1} \|S_k\| \leq B_A(e_1), \quad m_1 \leq N & \\ \|B_D\| - B_A(e_i) + \sum_{k=m_{i-1}+1}^{m_i} \|S_k\|, & \text{if } 2 \leq i \leq N \\ \quad \exists max(m_i) | \sum_{k=m_{i-1}+1}^{m_i} \|S_k\| \leq B_A(e_i), \ m_i \leq N & \end{cases}$$

Figure 4.4 depicts a sample segment downloading scenario under the buffer-limited batches. As with the previous example, the video is divided into 24 equal-length segments. A sufficient amount of receive bandwidth exists to allow retrievals of all 24 segments simultaneously. The download buffer space is limited to 10 segments. Under this condition, 15 batches are needed to receive all 24 segments. The segments 10 through 24 are all received at 1/10 of the nominal streaming rate.

Figure 4.5 illustrates an example of how a user may receive segments under both rate-limited as well as buffer-limited batches in real life. As with the previous examples, a streamed video consists of 24 equal-size segments. The total receive bandwidth, R_T, of the user is twice as much as the nominal streaming rate of the video segment. The total download buffer space, B_D, can accommodate a maximum of 10 simultaneous segment downloads. Segments are received in a total of 15 batches. Notice that the first two batches are rate limited, the next eight batches, $\beta(e_3)$ through $\beta(e_{15})$, are buffer limited, and the final nine batch sizes are determined by the remaining number of segments, $\overline{N}(e_i)$, yet to be received.

The algorithm for the segment reception scheduling is given in Appendix B.

4.3 RESTRAINED SEGMENT RECEPTIONS

The receive bandwidth and buffer management scheme of Sliding Batch allows users of a VT Room to prefetch future video segments as much as their resources permit. While this approach contributes to a greater segment availability within a VT Room, the greedy nature of Sliding Batch can induce a high bandwidth demand on a VT Distributor. This condition can be observed when a VT Room consists of many users with a limited transmit bandwidth

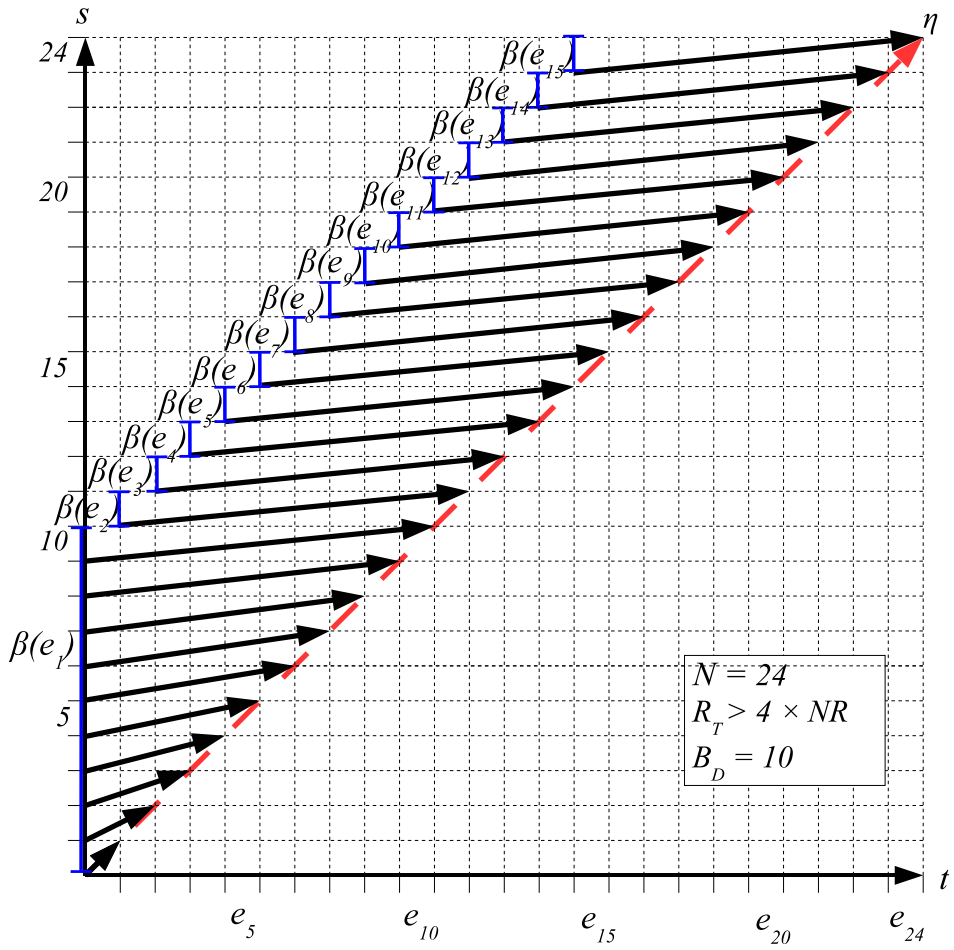


Figure 4.4: Sample segment retrievals under the buffer-limited batches

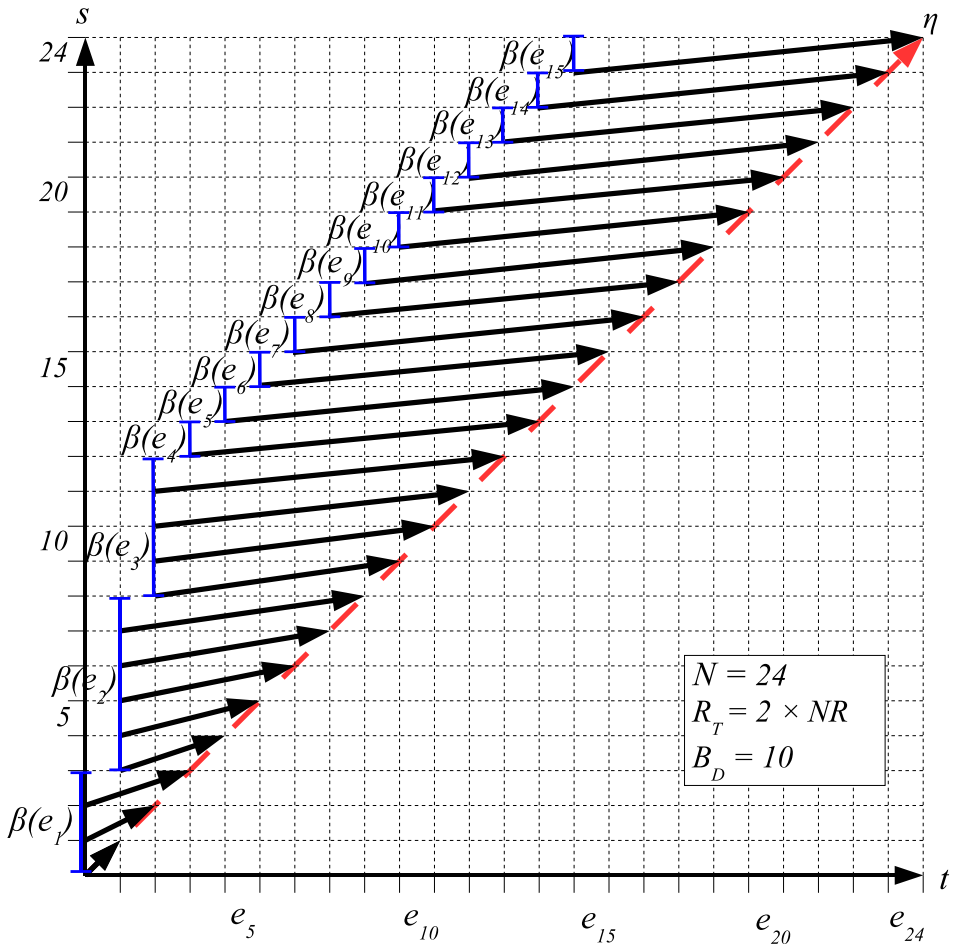


Figure 4.5: Sample segment receptions: bandwidth and buffer limited case

capacity, yet they have a large amount of available receive bandwidth and buffer space. Under such environment, the majority of segment distribution requests will be directed to and supported by the VT Distributor.

In order to avoid a concentration of segment requests at the VT Distributor, Sliding Batch will need to be tamed and become more moderate in its segment prefetching behavior. *Restrained Sliding Batch* aims to reduce bandwidth demand on VT Distributor by imposing an upper limit on the amount of segment downloads users may request.

For a VT Room to be highly self-sustainable (i.e. low degree of reliance on VT Distributor) in distributing video segments among its users, there must be sufficient transmit bandwidth available among its members to sustain the total needs. Let $T_A(t)$ and $R_A(t)$ be the available transmit and receive bandwidth of a user at time t. To be self-sustaining, the following condition must be satisfied.

$$\sum_{U_i} T_A(t) \geq \sum_{U_i} R_A(t), \quad \forall i,\ i \in VTR$$

One simple way to ensure the above condition is to limit the use of receive bandwidth at or below their initial available transmit bandwidth. In other words, a user is allowed to download segments if the total segment downloading rate does not exceed the initial available transmit bandwidth.

$$T_A(e_1) \geq R_U(e_i), \quad 1 \leq i \leq N$$

The exception to the above rule is when the user's initial available transmit bandwidth is less than the nominal streaming rate, $T_A(e_1) < \eta$. Users that fall in this category are allowed to receive one segment at a time in sequence at the nominal streaming rate. Accordingly, the value of R_T in equations (4.1) and (4.2) for the receive bandwidth management is initialized as follow:

$$R_T = min\{\eta, max\{R_T, T_T\}\}$$

The analysis of the performance gain achieved by the Restrained Sliding Batch over Sliding Batch is detailed in Chapter 7.

4.4 USER PROFILE AND VT ROOM PROFILE

The parameters used in Sliding Batch for the computation of batch sizes belong to two types of profiles. *VT Room profile* describes the attributes of a video being distributed in a VT Room. They include parameters such as the size of a streaming video, $|V|$, its playback duration, $\delta(V)$, the total number of segments, N, and the size of each segment in number of bits $\{\|S_i\|, i = 1, 2, \cdots, N\}$. VT Room profile is given to all users in each VT Room at their join time by the VT Distributor.

User profile describes the attributes of an individual user, primarily its resource availability, and consists of the following parameters: the time a user joined a VT Room, t_0, the total receive bandwidth set aside for the streaming service, R_T, the downloading buffer size, $|B_D|$, which dictates the maximum number of concurrent segment downloads, and the size of post-playback buffer space, $|B_H|$, which determines how long a segment will remain in buffer after its playback. Users in a VT Room advertise their user profile through the advertisement and discovery scheme described in the next chapter.

Sliding Batch significantly simplifies the advertisement of segment reception states of users. It enable users to express their complete segment reception order, timing, and rates from the first epoch to the last in one advertisement at the time of their VT Room join. Similarly, one query is sufficient to know the entire video segment download and playback sequences of another user. In the following chapter, the details of how the user profile is shared among the users of a VT Room are given.

5.0 SEGMENT ADVERTISEMENT AND DISCOVERY

This section describes the video segment advertisement and discovery scheme used in a VT Room. *Virtual Chaining* allows users of a VT Room to cooperatively maintain a collection of user profiles, known as a *state table*, to share the segment reception state information of users. Through a selection process, users identify a set of prospective distribution sources from the state table. Virtual Chaining also defines a mechanism through which users communicate the changes in their transmit bandwidth availability. The details of the state table maintenance and distribution source identification procedures are described.

5.1 STATE TABLE

A state table is a collection of user profiles maintained cooperatively among the members of a VT Room. It describes each user's segment reception state and the transmit bandwidth availability. An entry in the state table consists of the following fields: IP address of the user advertising its state, parameters of the user profile (t_0, R_T, $|B_B|$, $|B_H|$), available transmit bandwidth (T_A), and the time of its entry. This is depicted in Figure 5.1.

An entry is added to the state table when a new user joins a VT Room. It is removed when the last video segment is dropped from the user's buffer. This condition can be determined by the current time, t_c, exceeding the expected time of the final video segment departure from the post-playback buffer, $|B_H|$.

$$t_c > t_0 + \delta(V) + \frac{|B_H|}{\eta}$$

Figure 5.1: Entry fields of a state table

5.2 STATE TABLE SHARING

The state table is shared among the users of a VT Room as follows. VT Distributor maintains the tail-end portion of the state table, which contains user profiles of the last n users who joined the VT Room. A newly arriving user U_i receives the tail-end portion of the state table from the VT Distributor and reports its profile. The VT Distributor adds U_i's profile in the state table and drops the oldest entry if the table becomes greater then n. VT Distributor waits for the next user arrival. In the mean time, U_i examines the tail-end portion of the state table and tries to identify other users who may be able to provide segment distributions. If more users needed to be discovered, U_i requests U_{i-n}, the oldest entry in the tail-end portion of the state table, to send the subsequent portion or portions of the state table U_{i-n} maintains, which contains the user profiles of U_{i-n} through U_{i-2n} and beyond if U_{i-n} had requested further information from other users. This process is repeated until qualified distribution sources are located. If no qualified distribution source is found after iterating through the chain of state tables, U_i requests VT Distributor to transmit the needed segment.

Figure 5.2 illustrates a sample trace of state table sharing process. In this figure, circles represent users and shaded boxes represent portions of the state table maintained by the user directly above them. While all users have a portion of overlapping state table, this figure only shows ones maintained by users whose user ID is a multiple of n. Each shaded box contains n entries of user profiles. The user pointed to by the oldest entry in the box is denoted by the dashed arrow extending from the box to the user. Solid arrows represent the transfer of state table entries.

A newly arrived user, U_k, joins a VT Room and receives the tail-end portion of the state table from the VT Distributor (step (1)), which contains the user profiles of U_{k-1} through

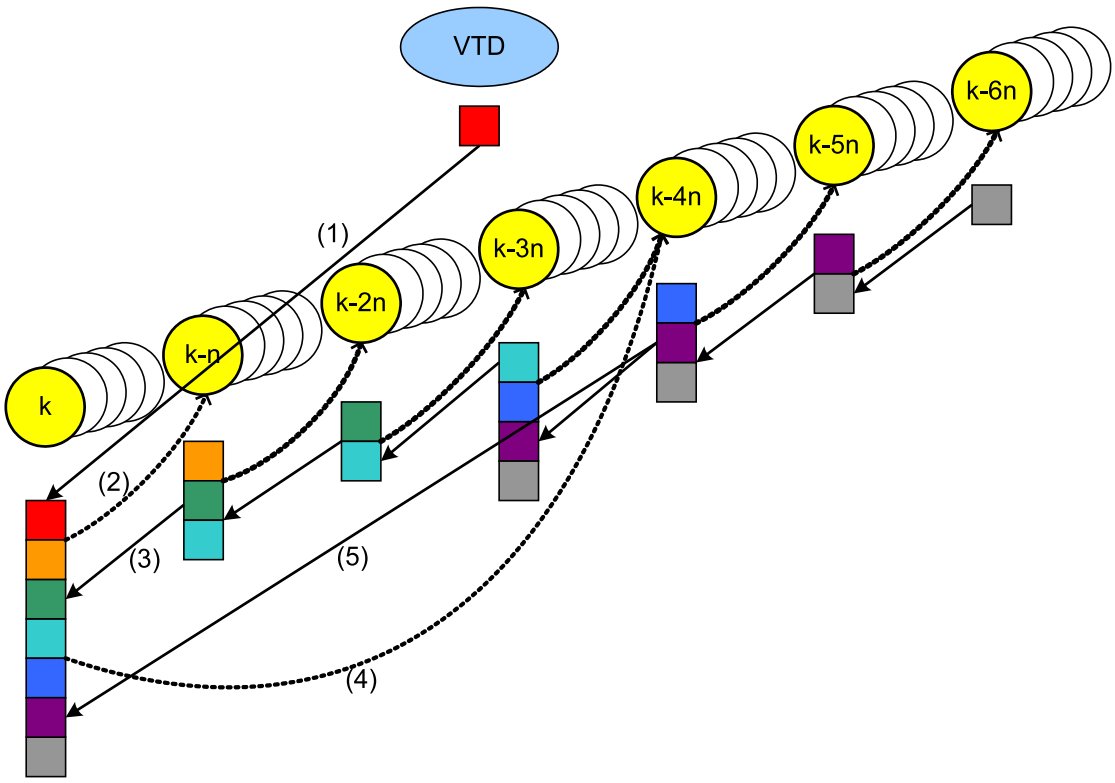


Figure 5.2: A sample view of state table sharing instances

U_{k-n}. U_k requests the next portion of the state table from the oldest entry, U_{k-n}, in the received state table (step (2)). U_{k-n} maintains the user profiles of U_{k-n-1} through U_{k-2n}, which where received from VT Distributor at its join time. U_{k-n} also has the user profiles of U_{k-2n-1} through U_{k-4n}, which was received from U_{k-2n}. All of these entries are sent from U_{k-n} to U_k (step (3)). U_k determines that it needs to discover other users and requests U_{k-4n} to send its portion of the state table (step (4)). U_{k-4n} sends the user profiles of U_{k-4n-1} through U_{k-7n} to U_k (step (5)).

If no response were received from a requested user (i.e. U_{k-n}), the next oldest entry in the state table (i.e. U_{k-n+1}) would have been contacted.

5.3 DISTRIBUTION SOURCE IDENTIFICATION

The selection of a distribution source from the state table is a two-step process. First, U_i identifies a set of *prospective* distribution sources. This is executed once at the time of the VT Room join. Second, U_i identifies a *qualified* distribution source among the prospective sources. This process is executed for each segment download at the beginning of each epoch.

To be considered for a prospective distribution source, an entry in the state table must satisfy the following two requirements: playback distance requirement and segment availability requirement. The playback distance requirement states that U_i can receive a video segment from another user, U_j, only if U_i begins playing back the first video segment after U_j and before it is being dropped from U_j's post-playback buffer. Let $t_0^{U_i}$ and $t_0^{U_j}$ be the time U_i and U_j joined the VT Room respectively. Let $|B_H^{U_j}|$ be the size of the post-playback buffer allocated at U_j. To meet the playback distance requirement, the following condition must be met:

$$t_0^{U_j} < t_0^{U_i} \leq t_0^{U_j} + \frac{|B_H^{U_j}|}{\eta}$$

The segment availability requirement states that U_i can receive a segment from U_j only if U_j began downloading the segment before U_i. Let $A^{U_i}(S_k)$ and $A^{U_j}(S_k)$ be the starting

download time of segment S_k by U_i and U_j respectively. To meet the segment availability requirement, the following condition must hold:

$$A^{U_i}(S_k) > A^{U_j}(S_k)$$

To be qualified for and selected as an actual distribution source, the transmit bandwidth availability requirement must be satisfied. It states that U_i can receive S_k from U_j only if U_j has the sufficient transmit bandwidth to support a segment distribution session at rate r_k, as required by U_i. Let $r_k^{U_i}$ be the rate at which U_i must download S_k. Let $T_A^{U_j}(e_m)$ be the transmit bandwidth available at U_j at time $\alpha(e_m)$. To satisfy the transmit bandwidth availability requirement, the following condition must be true:

$$T_A^{U_j}(e_m) \geq r_k^{U_i},\ S_k \in \beta(e_m)$$

When multiple users qualify, the user who departs first from the VT Room will be selected. This is done to minimize the loss of unused resources in a VT Room.

5.4 TRANSMIT BANDWIDTH AVAILABILITY UPDATES

In order for U_i to be able to identify a qualified distribution source, the time-varying transmit bandwidth availability of prospective distribution sources must be known. To achieve this, U_i subscribes to a state change notification service at each prospective distribution source using a publish-subscribe method. Each time a change in transmit bandwidth occurs at a prospective distribution source, a notification message is sent to each subscriber through a point-to-multipoint link.

5.5 MOBILITY SUPPORT EXTENSION

The distribution source selection algorithm can be extended to include location and movement information to provide mobility support. Mobility extension of the distribution source

selection algorithm requires four additional parameters to be supplied by mobile users: 1) time of mobility information update, t, 2) location (i.e. (x, y) coordinate of GPS) of the mobile user at time t, (x_t, y_t), 3) general direction of the mobile user movement, θ, and 4) average rate of mobile user movement, r. We assume that each mobile user is capable of detecting and expressing its location and movement information. This information is included in the user profile and advertised to other users through the state table sharing scheme. The mobility information updates are made at the beginning of every epoch to those users who have subscribed to the state change notification service as described in the previous section.

Using the above parameters, the scheme computes the mean distance from the requesting user, U_i, to each prospective distribution source, U_j. U_i selects the one with the minimum mean distance. To compute the mean distance, U_i estimates the distances to U_j for the duration of the segment reception and takes the average. Let $\overline{d_{i,j}}(t, t + \Delta(S_k))$ be the mean distance between users U_i and U_j from time t to $t + \Delta(S_k)$, where $\Delta(S_k)$ is the time it takes to download segment S_k. U_i selects the distribution source U_j user the following condition:

$$\exists U_j \mid min(\overline{d_{i,j}}(t, t + \Delta(S_k)), \quad j \in \chi_i$$

$$\overline{d_{i,j}}(t, t + \Delta(S_k)) = \frac{\sum_{m=0}^{\Delta(S_k)} d_{i,j}(t + m)}{\Delta(S_k)}$$

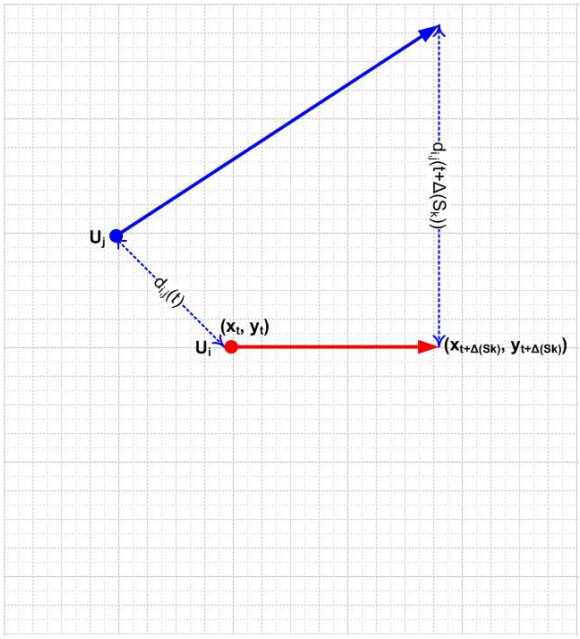
$$d_{i,j}(t + m) = \sqrt{(x^i_{t+m} - x^j_{t+m})^2 + (y^i_{t+m} - y^j_{t+m})^2}$$

where χ_i is the set of prospective distribution sources maintained by U_i and $d_{i,j}(t + m)$ is the distance from U_i to U_j at time $t + m$.

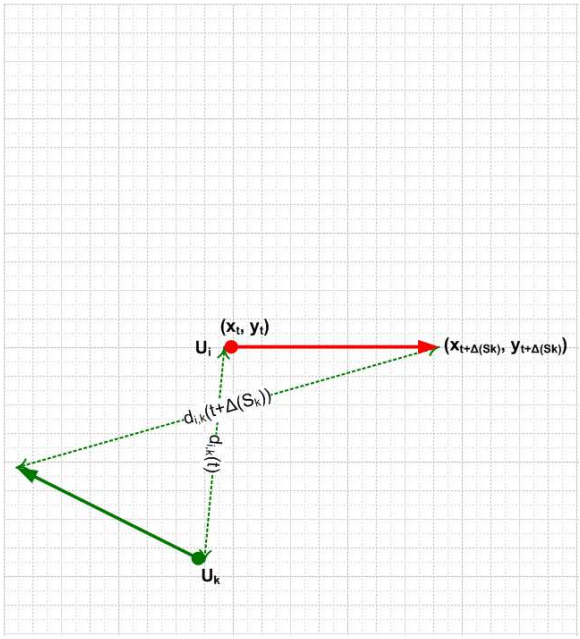
In practice, estimates from several discrete points in time will be used in the determination of the mean distance. At minimum, the distances at present time, t, and at the segment download finish time, $t + \Delta(S_k)$, will be included in the computation.

$$\overline{d_{i,j}}(t, t + \Delta(S_k)) = \frac{d_{i,j}(t) + d_{i,j}(t + \Delta(S_k))}{2}$$

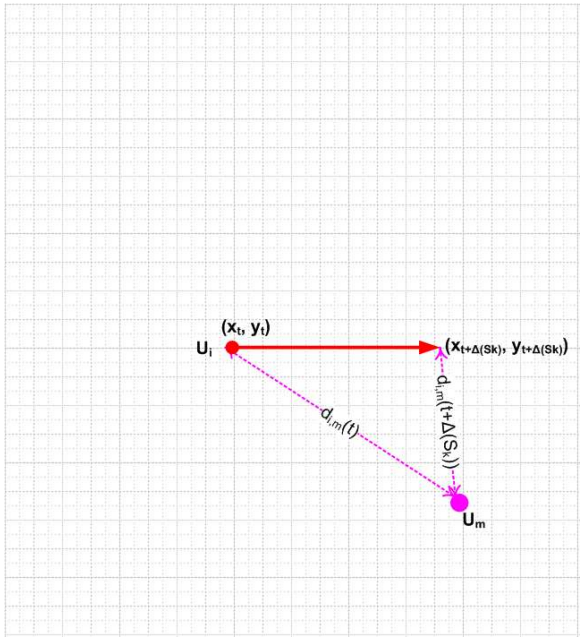
Figure 5.3 illustrates a conceptual view of sample distance estimations from U_i to three other nodes. In this example, U_i takes two samples of distance estimation to other nodes. U_i is located at (x_t, y_t) at time t and needs segment S_k. By examining the state table, U_i identifies three prospective distribution sources, U_j, U_k, and U_m. This is depicted in Fig 3(a),



(a) Distance from U_i to U_j



(b) Distance from U_i to U_k



(c) Distance from U_i to U_m

Figure 5.3: Sample distance estimations from U_i to three other nodes at time t and at time $t + \Delta(S_k)$

Fig 3(b), and Fig 3(c) respectively. Note that U_m is a stationary node in this example. From the parameters of the user profiles of the prospective distribution sources, U_i determines their locations and estimates the distances, $d_{i,j}(t)$, $d_{i,k}(t)$, and $d_{i,m}(t)$ respectively. By the time $t+\Delta(S_k)$, U_i finishes downloading S_k and will have migrated to location $(X_{t+\Delta(S_k)}, y_{t+\Delta(S_k)})$. U_i determines the locations of U_j, U_k, and U_m at time $t+\Delta(S_k)$, and estimates the distances, $d_{i,j}(t+\Delta(S_k))$, $d_{i,k}(t+\Delta(S_k))$, and $d_{i,m}(t+\Delta(S_k))$. U_i computes the mean distance of each prospective node and selects the node with the minimum mean distance as the distribution source for S_k.

In addition, users may select a distribution source according to some priorities associated with the parameters of the user profile, as determined by the user. Each parameter, m, is associated with a weight, $0.0 \leq \alpha^m \leq 1.0$, with 1.0 being the most preferred. The value associated with the parameter, β^m, is transposed and scaled to a value between 0.0 to 1.0, with 1.0 being the best possible value. The user selects the distribution source who maximizes the weighted sum of the evaluated parameters.

$$\exists U_j \mid max(\sum(\alpha_j^m \cdot \beta_j^m)), \quad j \in \chi_i$$

$$0.0 \leq \sum(\alpha_j^m \cdot \beta_j^m) \leq 1.0$$

$$1.0 = \sum \alpha_j^m$$

For example, a user may give a weight of α^1 to the parameter for the rate of user movement, α^2 to the current distance between the two nodes, α^3 to the mean minimum distance, and α^4 to the available transmit bandwidth. The user who scores the highest in the summation of the weighted and scaled values will be chosen as the distribution source.

$$\exists U_j \mid max((\alpha_j^1 \cdot r) + (\alpha_j^2 \cdot d_{i,j}(t)) + (\alpha_j^3 \cdot \overline{d_{i,j}}(t, t+\Delta(S_k))) + (\alpha_j^4 \cdot T_A)), \quad j \in \chi_i$$

Due to its simple operation, Virtual Chaining is relatively easy to implement, deploy, and study its behavior. A distributed and redundant state table available at participating users offers resiliency such that a loss of a few users do not break the segment advertisement, discovery, or distribution operation. Virtual Chaining is fair, in terms of the carried workload

among the users, that no single user is expected to perform more work than others. Virtual Chaining is also scalable in that the workload placed upon each user remains a constant regardless of the size of the membership in the P2P community.

6.0 QOS SUPPORT

For a smooth playback of a streaming video, a timely reception of video frames is essential. Video freezing and other perceptual quality losses result when video frames do not arrive in time. Although attempts have been made in the past to control and minimize packet delays (e.g. IntServ, DiffServ), the Internet remains to be a *best-effort* network today. Due to lack of delay control in the underlying network, users may observe a wide range of fluctuating delays. Furthermore, users may experience excessively large delays due to sudden loss of distribution sources, which is a likely event in peer-to-peer based distribution service.

To provide a level of performance assurance necessary to support a smooth playback of a streaming video in a VT Room, delays that are inherent in the best-effort network as well as delays caused by the loss of distribution sources will need to be addressed.

This chapter presents a QoS scheme which aims to provide a level of assurance in preventing an interruption of video viewing while users experience diverse levels of delays. First, a delay coping mechanism of existing streaming applications is presented, followed by a description of a set of challenges faced by Sliding Batch in applying the existing mechanism. Second, an overview of the proposed QoS scheme is presented, followed by a description of key mechanisms used to mitigate the impacts of delays in our proposed QoS scheme.

6.1 EXISTING DELAY COPING MECHANISM

Streaming applications that operate over a traditional video distribution network employ *playout buffer*, B_P, to cope with fluctuating network delays. The goal of B_P is to prevent video frame starvations during playback. This is achieved by prefetching an initial portion

of a video stream and withholding its playback for a predetermined duration of time. The delay incurred by this operation is referred to as a *playout delay*, D_P, or a *start-up delay*. In exchange for inducing D_P, it is hoped that the subsequent delays during the life-time of a video playback may be absorbed by B_P. D_P should be long enough to be able to cope with typical delays, yet short enough for users to tolerate the initial waiting time.

Two issues need be considered when incorporating a traditional delay coping mechanism in Sliding Batch. First, the playout buffer was never designed to cope with an excessive delay caused by a loss of a distribution source. As such, once B_P becomes empty, rebuffering of video stream is necessary to handle the future delays, which interrupts the playback of a video for the duration of D_P. Simply increasing the size of B_P does not solve the problem as it only induce a longer interruption period.

Second, a detection of a distribution source loss in Sliding Batch is not as straightforward as in the traditional video distribution network. Under the conventional delay coping mechanism, the loss of distribution source may be declared when B_P becomes empty. In contrast, it may not be practical to wait until B_D becomes empty to declare the loss of a distribution source in Sliding Batch. When a segment is downloaded over multiple epochs, as it is often the case in Sliding Batch, the draining of video frames from B_D begins only during the last epoch that a substantial waiting time may be involved before the distribution loss decision can be made. A mechanism which allows timely detection of a distribution source loss condition to quickly switch to an alternate qualified distribution source is needed.

To address these issues, a set of delay management mechanisms are proposed. *Delay Monitor* (DM) provides timely detection of excessive delay condition. *Extended Playout Delay* (EPD) offers protection against a loss of distribution sources by sacrificing the immediate playback. *Expedited Segment Reception* (ESR) protects against a loss of distribution sources by sacrificing the bandwidth. They work together to prevent buffer underrun conditions during multiple instances of distribution source losses. The details of Delay Monitor, Expended Playout Delay, and Expedited Segment Reception are presented next.

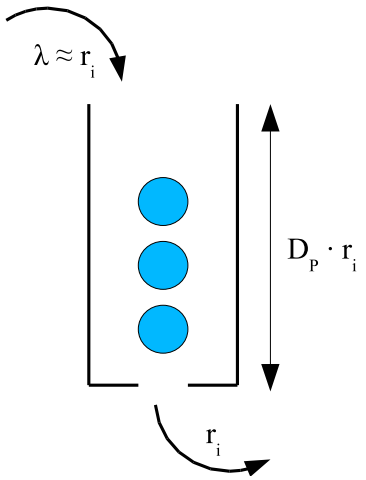


Figure 6.1: A Sample Delay Monitor

6.2 DELAY MONITOR

Delay Monitor, *DM*, monitors the flow of data arriving to B_D and detects an excessive delay condition. An excessive delay condition is declared when the arrival rate of data drops below a certain threshold. DM uses a leaky bucket for the monitoring and detection of excessive delay condition. At each data arrival, a token equivalent to the size of arriving data is added to the bucket. After the start-up delay period has elapsed, the tokens are removed from the bucket at rate r_i, the nominal downloading rate of S_i. When the bucket becomes empty, an excessive delay is declared and a new connection will be established with a different distribution source.

Figure 6.1 depicts a conceptual view of a DM which monitors data arrival for S_i, whose expected arrival rate, λ, is r_i. Tokes are removed exactly at rate r_i from the bucket with the depth of $D_P \cdot r_i$.

The value of playout delay, D_P, should be selected in such a way that it is long enough to absorb typical delay fluctuations, yet short enough for users to tolerate the initial delay. Furthermore, it should not be too long before switching to a new distribution source; however, it should not be too short to cause unnecessary distribution source switches.

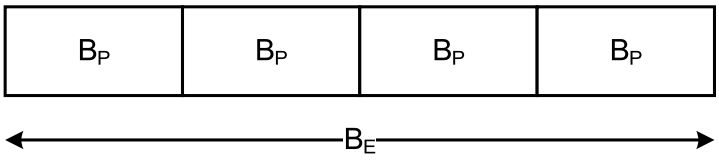


Figure 6.2: A sample extended playout buffer

6.3 EXTENDED PLAYOUT DELAY

Extended Playout Delay (EPD) is designed to prevent a buffer starvation condition after an excessive delay is detected. This is achieved by having users wait extra time before the initial video playback can begin. EPD decouples the excessive delay detection period (i.e. D_P) from the initial playout delay period (i.e. extended playout delay, D_E) and the following relationship exists between the two:

$$D_E = n \cdot D_P = \frac{n \cdot B_P}{\eta}$$

where n is the level of protection against excessive delays and corresponds to the number of times a user may encounter excessive delays but not experience video presentation interruptions.

Figure 6.2 displays a logical view of a sample extended playout buffer, B_E, with the excessive delay protection level of $n = 4$.

Let D_{En} be the start-up latency introduced by EPD to provide n^{th} level of excessive delay protection. At $D_{E1} = 1 \times D_P$, EPD only provides delay absorptions up to D_P with no protection against a distribution source loss, just as a traditional playout buffer does. At $D_{E2} = 2 \times D_P$, EPD offers a one-time distribution source loss protection during the life-time of a segment download, in addition to delay absorptions up to D_P. It provides an uninterrupted video presentation if a user looses a distribution source and begins receiving the segment from a new distribution source. However, if the user looses the new distribution source, there will be a playback interruption. At $D_{E3} = 3 \times D_P$, a two-time distribution

source losses can be tolerated during the life-time of a segment download, in addition to delay absorptions up to D_P. By extending the start-up delay, the level of protection against multiple instances of distribution source losses over the life-time of a segment download can be increased. In other words, a higher degree of protection can be archived if a user is willing to wait for a longer initial playout delay.

Sample segment receptions with EPD is depicted in Figure 6.3. In this example, start-up delay has been extended to $D_{E3} = 3 \times D_P$. The dotted arrow lines running diagonally across the middle of the figure represent the nominal streaming rate and show the playback positions of the streamed video at various levels of protections, ranging from none to the 3^{rd}. S_1 and S_3 both experience an excessive delay twice during their download, which is depicted by a horizontal line with a diamond shaped starting point. After D_P period of time, the user begins retrieving the affected segment from another distribution source. Note that, even after moving to a new distribution source twice, a sufficient amount of data has been prefetched in buffer to provide the absorption of delays up to D_P.

6.4 EXPEDITED SEGMENT RECEPTION

Expedited Segment Reception (ESR) provides a protection against a loss of distribution source by increasing the segment downloading rate. By expediting the ending time of a segment download, ESR attempts to gain enough time to recover the time loss experienced by the detection of excessive delays.

Let r_i^{En} be the rate of ESR for downloading S_i at n^{th} level of protection against excessive delay. At $r_i^{E_1} = \|S_i\|/(\Delta(S_i) - 1 \times D_P)$, where $\Delta(S_i) = \alpha(e_i) + \sigma(e_i) - \alpha(S_i)$, one-time distribution source loss protection can be offered during the life-time of S_i download. It provides an uninterrupted video presentation if a user looses a distribution source and begins receiving the segment from a new distribution source. However, if the user looses the distribution source the second time, there will be a playback interruption. At $r_i^{E_2} = \|S_i\|/(\Delta(S_i) - 2 \times D_P)$, a two-time distribution source loss protection can be offered during the life-time of S_i download. At $r_i^{E_3} = \|S_i\|/(\Delta(S_i) - 3 \times D_P)$, a three-time distribution source loss protection can

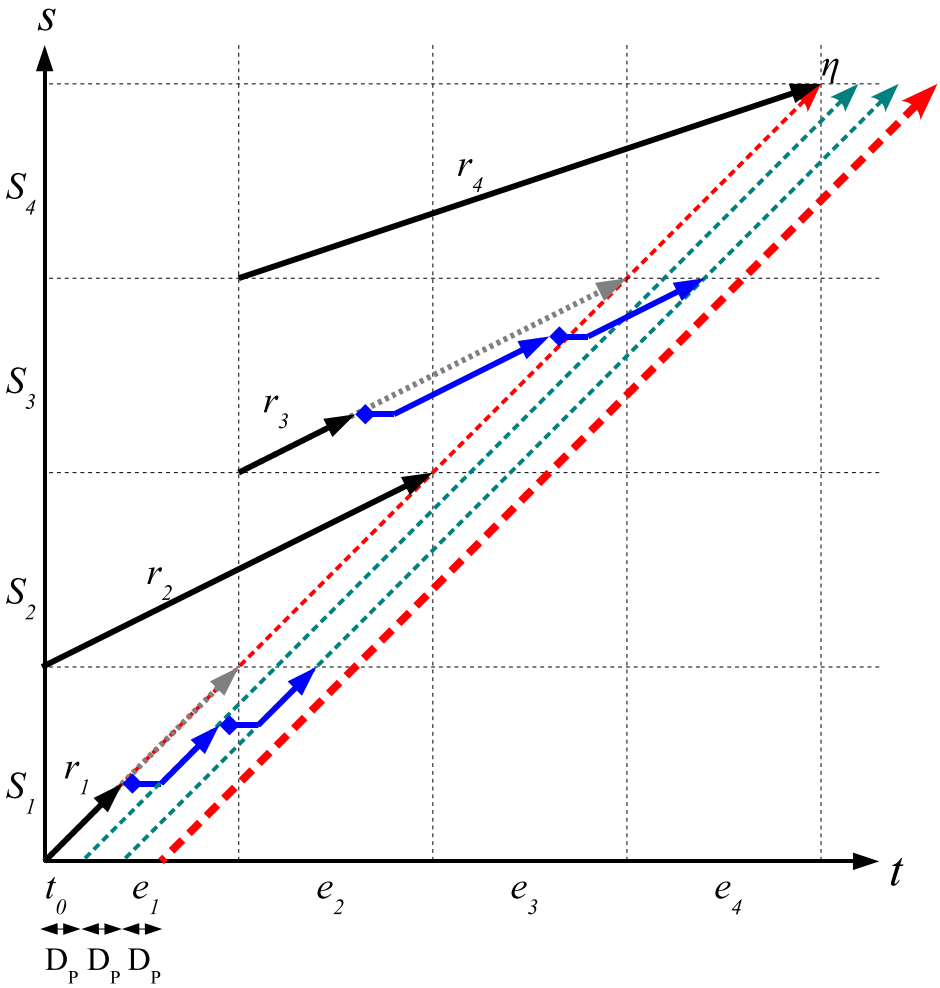


Figure 6.3: Sample Segment Receptions with Extended Playout Delay

be offered during the life-time of S_i download. A higher degree of protection can be achieved with relatively small amount of additional receive bandwidth.

Figure 6.4 shows an example of how video segments may be received under ESR. In this example, S_1 is protected against one-time distribution source loss by receiving data at $r_i^{E_1}$. S_2 and S_3 are protected against two-time losses by increasing the reception rate to $r_i^{E_2}$. At $r_i^{E_3}$,S_4 can withstand three-time distribution source losses. The example shows that excessive delays have been observed while downloading S_1 and S_3, but they have not affected the performance of video playback due to a sufficient amount of data prefetched through ESR.

Each segment downloading session can be associated with a different degree of protection through ESR. For example, if a measurement shows that a significantly higher rate of distribution source loss is experienced in downloading S_i, a higher level of protection can be associated with the reception of S_i. Let L_i be the maximum number of distribution source losses a user anticipates when downloading a segment S_i. The expedited rate of reception, $r_i^{E_L}$, that protects against L number of distribution losses is given by

$$r_i^{E_L} = \frac{\|S_i\|}{\Delta(S_i) - L_i \times D_P}$$

In order to determine the level of protection needed for each segment download, the statistic on the loss of distribution source must be collected. This is achieved by informing the VT Distributor every time a user experiences a distribution source loss. The VT Distributor keeps the statistical information of distribution source losses for each segment and shares it when users join the VT Room. The user determines the required level of protection for a successful segment downloading by consulting the distribution source loss statistic being supplied by the VT Distributor.

Let P_i be the probability of a user experiencing a distribution source loss when downloading S_i, assuming loss of distribution sources are IID. To achieve a successful download of S_i at or above a protection goal, g, such as $g = 0.95$, the following condition must be met:

$$g \geq 1 - P_i^{L_i}$$

The level of protection needed for S_i download can be computed by solving for L_i.

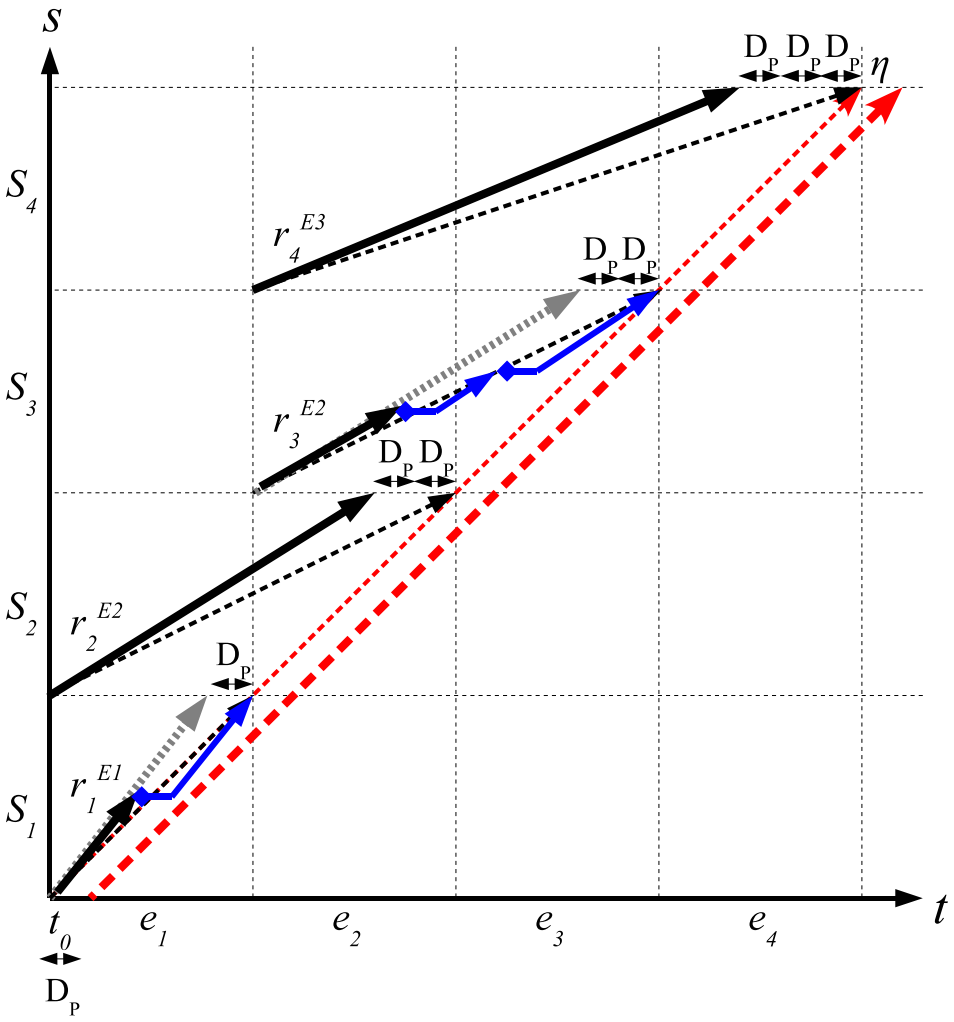


Figure 6.4: Sample Segment Receptions with Expedited Segment Reception

The user executes the *Excessive Delay Protection Algorithm*, as depicted in Algorithm 1, to determine the amount of Extended Playout Delay and the rate of Expedited Segment Reception. The strategy used in this algorithm is to let the user wait as long as it is willing at the initial playback time through EPD. If necessary, expedite individual segment receptions through ESR. Let L be the maximum level of protection required to meet g. Initially, L is set to the maximum value of L_i, the greatest level of protection required among all segment receptions. Let w be the maximum time a user is willing to wait for the start-up latency. If the extended playout delay, $D_E = L \times D_P$, is greater than w, L is set to $\lceil w/P_D \rceil$. This is the level of protection offered by EPD. For each segment that belongs to a batch, $\beta(e_i)$, a need for an additional level of protection through ESR is investigated. If the level of protection, L_i, required for downloading S_i is greater than the level of protection provided by the EPD (i.e. L), the rate at which the segment is downloaded will be increased to $r_i^E = \|S_i\|/(\Delta(S_i) - (L_i - L) \times D_P)$. If not enough receive bandwidth is available, the segment download will not be initiated.

Algorithm 1 Excessive Delay Protection algorithm

```
1: // initialize
2: L = max(L_i)
3: // let user wait as long as it is willing
4: if (L × D_P > w) then
5:     L = ⌈w/D_P⌉
6: end if
7: for each segment ∈ β(e_i) do
8:     // increase the download rate as needed
9:     if (L_i > L) then
10:        r_i^E = ||S_i||/(Δ(S_i) − (L_i − L) × D_P)
11:    end if
12:    // not enough RxBW to meet the protection requirement
13:    if (r_i^E > R_A) then
14:        print warning and break
15:    end if
16: end for
```

7.0 SIMULATION DESIGN AND ANALYSIS

This chapter describes the design and analysis of experiments performed on Virtual Theater Network. A software model was created to simulate the behavior of a VT Room. Two types of experiments were conducted. The focus of the first sets of experiments was to study how well the proposed video distribution scheme would alleviate the load on the VT Distributor under different operating environment. Specifically, the peak transmit bandwidth usage of a VT Distributor is measured under different user inter-arrival times, nominal streaming rates, video segment sizes, and user receive bandwidth availability. We call these sets of experiments "Sliding Batch" experiments. The second sets of experiments focused on how well the proposed QoS scheme would mitigate the impact of distribution sources losses on the video presentation. The average number of video presentation interruptions is collected at participating users under different rate of distribution source losses. We call this group of experiments "QoS extension" experiments. The description of the model, the design of the experiments, and the analysis of simulation results are given below.

7.1 THE MODEL DESCRIPTION

The simulated VT Room consists of a VT Distributor and a series of user processes that arrive to the VT Room. VT Distributor supplies the parameters defined in VT Room profile to the newly joining users, such as the total video playback time *(120 minutes)*[1], the nominal streaming rate *(1.0 Mbps)*, and the total number of segments *(24)* in the video. The video is divided into equal length of segments and equal duration of playback time *(5 minutes)*.

[1]The value in the parenthesis denotes the default value used in the experiments

The user processes simulate the behavior of peers joining the VT Room, discovering other users, identifying possible distribution sources, receiving video segments, distributing video segments as requests arrive, and departing from the VT Room. The inter-arrival time of user processes is exponentially distributed *(mean 10 seconds)*. To reflect the asymmetrical nature of the transmit and receive bandwidth capacity of typical broadband access technologies, and the diversity in the amount of transmit bandwidth availability, each user is equipped with a fixed receive bandwidth *(2.0 Mbps)* and varying transmit bandwidth *(30% to 100% of the receive bandwidth; uniformly distributed)*. Each user executes Virtual Chaining to identify possible distribution sources and implements Sliding Batch to receive video segments. All experiments simulate the bandwidth-limited network environment where sufficient amount of downloading buffer exists at each user *($|B_H| \geq N$)*. The default post-playback buffer size allows a segment to remain in buffer for a finite period of time *(15 minutes)* after its playback. The first group of experiments allows all users to receive needed segments successfully and complete the viewing of the entire video. In the second group of experiments, a set of users departs from VT Room prematurely.

7.2 EXPERIMENTAL DESIGN AND ANALYSIS

Four sets of experiments are conducted in the Sliding Batch experiments, each measuring the effectiveness of proposed schemes in mitigating the bandwidth demand on the VT Distributor. The experiments focus on the following areas: 1) the effects of user arrival patterns, 2) the effects of nominal streaming rates, and 3) the effects of user receive bandwidth availability, and 4) the effects and interactions of video segment sizes and the post-playback buffer sizes on the VT Distributor load.

For each set of experiments, the measurements are taken on the peak transmit bandwidth usage of VT Distributor using the following schemes: a traditional client-server video distribution scheme, Sliding Batch, Restrained Sliding Batch, and Chaining [34]. A traditional client-server video distribution refers to a scheme where all users receive video feeds from a central server. The measurements taken from this scheme are used as the base-line and the

measurements from other schemes are normalized to the base-line values when graphed. *Restrained Sliding Batch* is a variant of Sliding Batch and it limits the use of receive bandwidth at or below their initial available transmit bandwidth. In other words, a user is allowed to prefetch segments if the total segment downloading rate does not exceed the initial available transmit bandwidth. It tames the greedy nature of the Sliding Batch and further reduces the bandwidth demand on the VT Distributor. The simulation results from Chaining are used as a reference point. Chaining is a peer-to-peer based streaming video distribution scheme. A major difference between Chaining and Sliding Batch is that Chaining distributes video stream in its entirety from one user to another at the nominal playback rate while Sliding Batch distributes video stream in multiple segments at or below the nominal playback rate.

The fourth set of experiments, which studies the effects and interactions of video segment sizes and the post-playback buffer sizes, incorporates 2^2 factorial design and conducts the analysis of variations (ANOVA). The details of its purpose and the results of the analysis is given in the respective subsection.

Each experiment is repeated 20 times to compute the mean and the variance. 90% confidence interval is used to establish the significance in the computed means for all experiments. When no statistical differences are observed between two means, it will be stated as such in the analysis section. Otherwise, a statistical significance has been verified and the null hypothesis, $H_0 : \mu_1 = \mu_2$, has been rejected for all experiments.

The first set of experiments studies the effects of user inter-arrival times on the load at VT Distributor. The mean inter-arrival times of user processes are varied from 5.0 to 60.0 seconds while other parameters were kept constant. The simulation results are shown in Figure 7.1. Restrained Sliding Batch offers as much as 90% of bandwidth reduction at the VT Distributor. Sliding Batch and Chaining achieved roughly 80% and 70% of bandwidth reductions respectively at their peaks. A common and assuring trend observed among the three schemes was that the greater the rate of user arrivals, the greater the bandwidth reductions at VT Distributor. This is most apparent in Restrained Sliding Batch and is a sign of scalability.

The second set of experiments studies the effects of playback rates on the load at VT Distributor. The nominal playback rate of a video stream was varied from 0.5 to 2.0 Mbps.

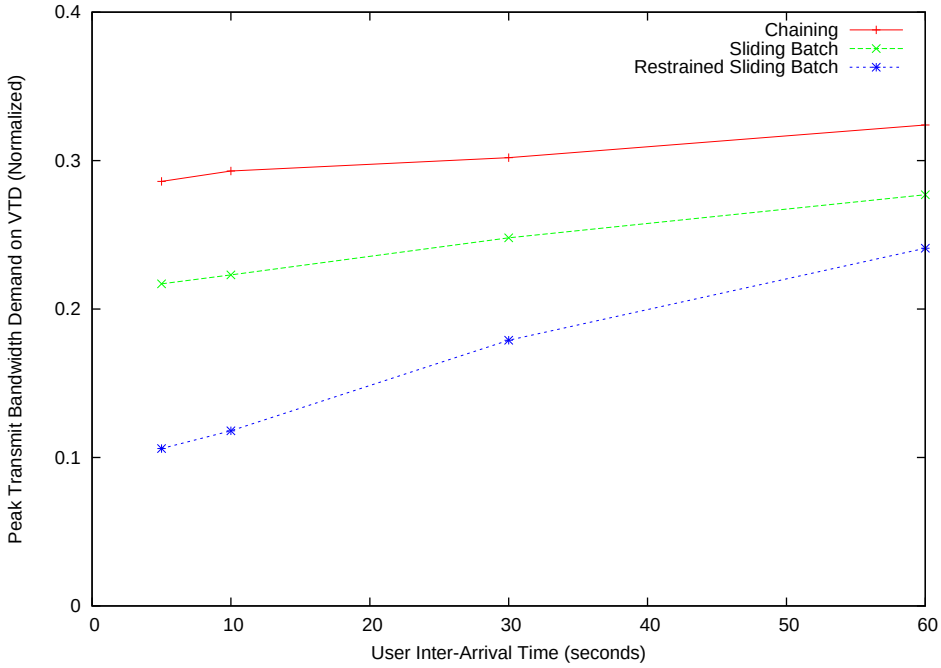


Figure 7.1: Effects of user arrival pattern

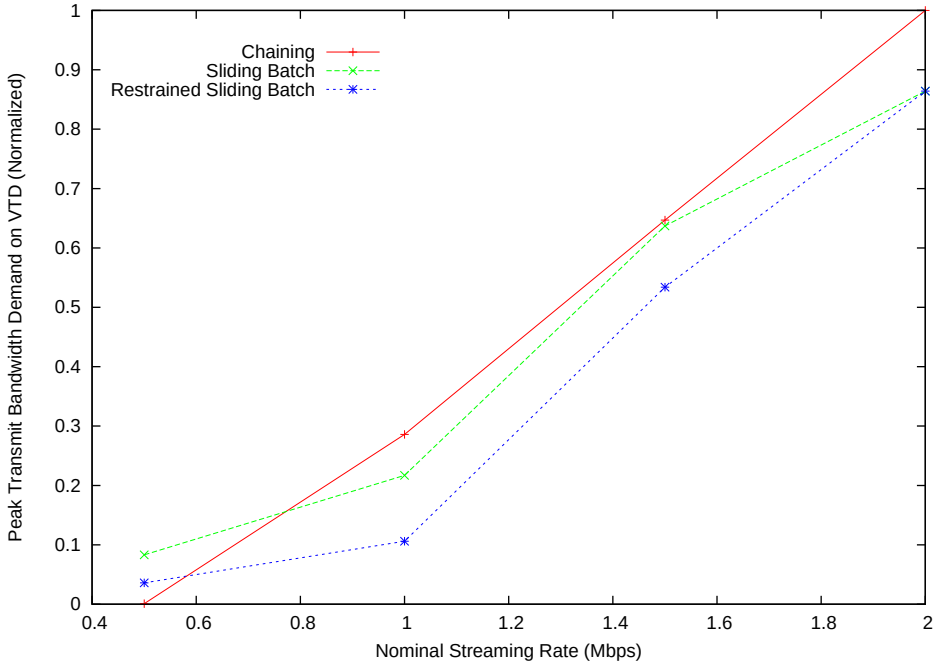


Figure 7.2: Effects of streaming rate

Under Chaining, the load on the video server increased linearly as the playback rate increased. Sliding Batch and Restrained Sliding Batch both have a milder incline when the playback rates are below the mean available transmit bandwidth of users. Chaining performed the best among the three schemes when the playback rate is very low. There was no statistical significance observed at 1.5 Mbps between Chaining and Sliding Batch. In all other regions of playback rates, Sliding Batch and Restrained Sliding Batch both achieved a greater load reduction. This is depicted in Figure 7.2.

The third set of experiments focuses on the effects of the amount of available receive bandwidth at users. As expected, a sharp increase in the bandwidth demand at VT Distributor is observed on Sliding Batch when users have a large amount of excess receive bandwidth. This is due to the greedy nature of Sliding Batch that it will prefetch a series of segments until all available receive bandwidth is consumed. In contrast, both Chaining and Restrained Sliding Batch maintain a constant level of transmit bandwidth demand at VT Distributor regardless of how much receive bandwidth is available at each user. Figure 7.3 shows that Restrained Sliding Batch requires roughly one-third of bandwidth at VT Distributor than Chaining.

The fourth set of experiments evaluates the impact of segment sizes to the load on VT Distributor. It also evaluates the interactions between the segment sizes (i.e. segment playback duration) and the post-playback buffer sizes (i.e. duration of time a segment is held in post-playback buffer).

Segment size is a parameter, within the VT Room profile, that the provider of the video distribution service may be able to engineer to achieve performance gain. Other parameters of the VT Room profile, such as the total video length and the nominal streaming rate, are either fixed or undesirable to be altered as they pertain to the fundamental property of the video distribution service. On the User Profile side, the size of the post-playback buffer is a parameter that the user may be able to adjust to achieve performance gain relatively easily, compared to other parameters such as the total transmit bandwidth and receive bandwidth. It is of our interest to discover an optimal combination of segment size and post-playback buffer size in a given environment or unfavorable combinations that should be avoided. We are also interested in interactions between the segment sizes and the post-playback buffer

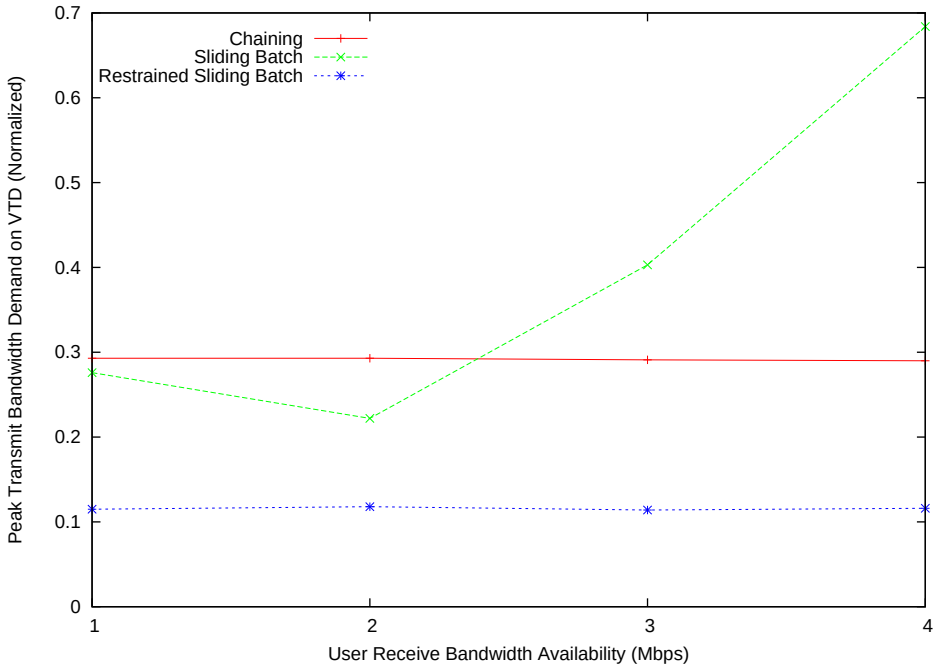


Figure 7.3: Effects of user bandwidth availability

sizes.

In order to achieve these goals, experiments are designed with 2^2 factorial design with 20 replications. Segment size is designated as factor A and post-playback buffer size is designated as factor B. Factor A has two levels, three minutes per segment (i.e. 40 total segments in a video) and 15 minutes per segment (i.e. 8 total segments in a video). Factor B also has two levels, five minutes worth of post-playback buffer size and 15 minutes worth of post-playback buffer size for each segment being downloaded. Each test is run 20 times to allow computation of mean square error. Table 7.1 shows the collected data with the factorial design.

The results of the experiments are summarized as ANOVA in Table 7.2. This table shows that the post-playback buffer has the largest influence on the performance of the video distribution service. The size of segments follows closely after the post-playback buffer in the influence it has on the outcome of the experiments. A strong interaction between the size of segment and the size of post-playback buffer is shown. By evaluating F_0 values, we

Table 7.1: Collected data with 2^k factorial design

Post-Playback Buffer Size	Segment Size							
	3 Min. (40 total segments)				15 Min. (8 total segments)			
5 Min.	0.348	0.323	0.352	0.339	0.351	0.336	0.345	0.357
	0.364	0.346	0.349	0.357	0.345	0.362	0.335	0.340
	0.341	0.348	0.322	0.347	0.329	0.364	0.362	0.342
	0.343	0.339	0.345	0.325	0.344	0.343	0.350	0.366
	0.378	0.332	0.329	0.335	0.348	0.345	0.346	0.343
15 Min.	0.217	0.202	0.223	0.234	0.351	0.336	0.345	0.357
	0.218	0.247	0.217	0.205	0.345	0.362	0.335	0.340
	0.209	0.202	0.215	0.215	0.329	0.364	0.362	0.342
	0.220	0.205	0.201	0.214	0.344	0.343	0.350	0.366
	0.211	0.202	0.213	0.243	0.348	0.345	0.346	0.343

Table 7.2: ANOVA for segment size and post-playback buffer size interactions

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0
Segment Size	0.0812	1	0.0812	574.9671
Post-Playback Buffer Size	0.0932	1	0.0932	660.0043
Interaction	0.0812	1	0.0812	574.9671
Error	0.0107	76	0.0001	
Total	0.2664	79		

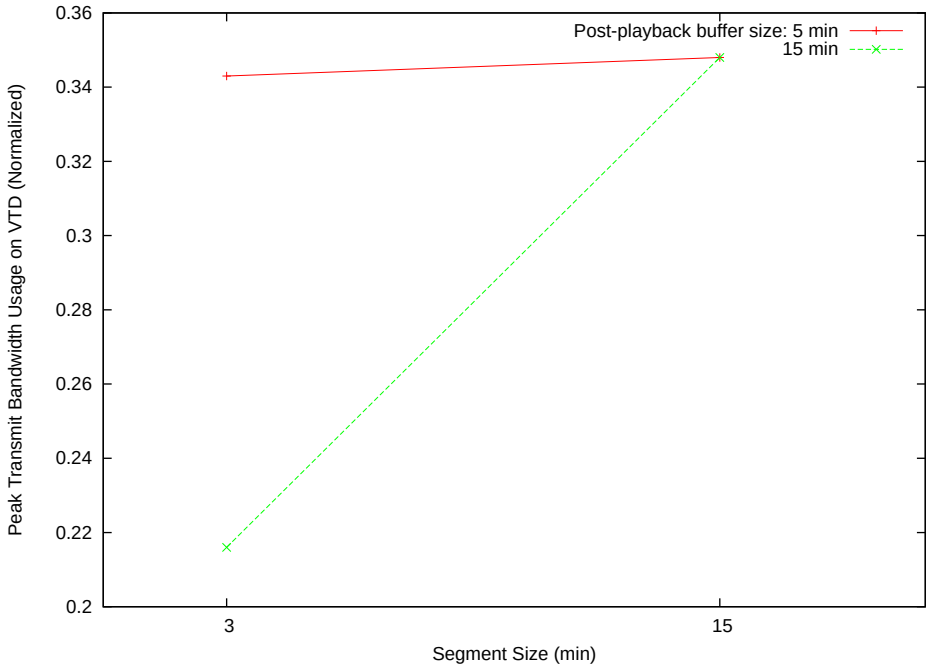


Figure 7.4: Interaction between segment size and post-playback buffer size

conclude that both the main effects as well as the interactions between the size of segment and the size of post-playback buffer means are significant, meaning the changes in the size of segment and post-playback buffer resulted in significant improvement in the offered load on the VT Distributor and this outcome is strongly influenced by each other. Segment size and post-playback buffer size plot of this experiment is depicted in Fig 7.4. Note that the strong interaction between two factors are displayed as two lines crossing or converging at the segment size of 15 minutes. Also note that there is a trend in the decrease in the VT Distributor load as the size of the segment becomes smaller and as the size of post-playback buffer increases.

Further experiments have been conducted to verify the above findings and to gain further knowledge of the behavior and interaction of segment sizes and post-playback buffer sizes. In addition to 5-minute and 15-minute post-playback buffer size, experiments with 30-minute post-playback buffer size are included in the new set of tests. Further more, observations with segment sizes of one minute, five minutes, 30 minutes, and 60 minutes are added to

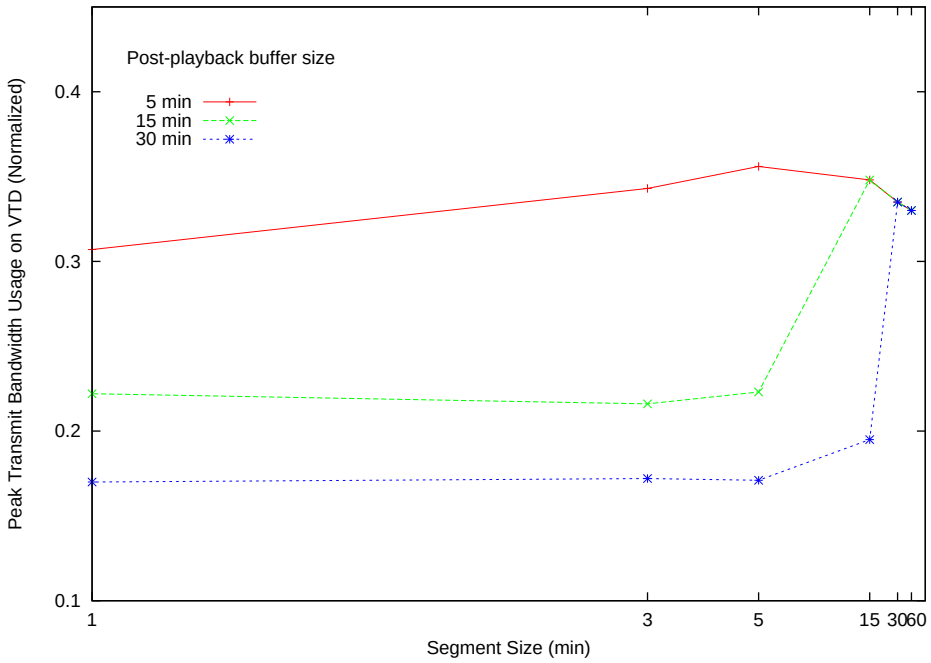


Figure 7.5: Effects of segment size and post-playback buffer size

the new set of experiments. Fig 7.5 shows the test results in graphs. The outcome verifies that, in generally, lower bandwidth demand is placed on VT Distributor when segments are held in buffer for a longer duration of time after their playback. When the segment size is larger than the post-playback buffer size, the same amount of load is placed on the VT Distributor regardless of how long the segments are kept in post-playback buffer. A decrease in the bandwidth demand at VT Distributor is observed once the segment size becomes smaller than the post-playback buffer size. With 15-minute and 30-minute post-playback buffer experiments, a substantial amount of bandwidth decrease is observed for segment sizes down to 5 minutes; however, very little load change is observed beyond that point. We conclude from these observations that the optimum balance between the segment size and the post-playback buffer size lies in the vicinity of 1:2 ratio (e.g. 15 minute segment size vs 30 minute post-playback buffer size). Any post-playback buffer configuration that are at or below the segment size should be avoided.

The above four sets of experiments have confirmed that peer-to-peer based streaming

service can alleviate load on the video server in significant amount. They also gave an assurance that splitting a video stream into multiple segments and distributing them concurrently at rates below the nominal playback rate allow further reductions in the transmit bandwidth demand on the video server. Additional load reduction was achieved by taming the greedy nature of Sliding Batch by restraining the amount of segment prefetching.

The next three sets of experiments study the effectiveness of QoS scheme in mitigating the video presentation interruptions when users experience excessive delays while receiving video segments. An excessive delay is defined as absence of frame arrival beyond playout delay such that it results in playout buffer starvation. An excessive delay may be caused by various reasons, such as unexpected departure of a distribution source from the service, persistent or sever network congestions, or a loss of connectivity due to user mobility. Regardless of the cause, an excessive delay introduces a video presentation interruption, at least for the duration of playout delay, since the playout buffer will need to be rebuilt. This is true when absence of video frame arrivals are dealt with a traditional playout buffer mechanism. On the other hand, the QoS scheme used in VT Room is designed to reduce the occurrences of video presentation interruptions. The sets of experiments being conducted allows the study of the proposed QoS scheme in reducing the number of interruptions and the load on the VT Distributor under varied extended playout delays and distribution source losses.

The first set of experiments studies the effects of the size of the extended playout delay in reducing the video presentation interruptions under different rates of premature user departures from the VT Room. Extended playout delay are varied from $1\times$ playout delay to $4\times$ playout delay. The probability of premature node departure, P, was varied from 0.1 to 0.4 and the duration of time a node may spend before its premature departure is uniformly distributed during the playback time of the entire video. The The total number of video presentation interruptions experienced by participating users are normalized to the total number of distribution source losses being detected in the VT Room. In Chaining, all distribution source losses being detected by the users resulted in the video presentation interruptions, as it uses the traditional playout buffer mechanism, regardless of the rate of user departures from the VT Room. This is depicted in Figure 7.6 by the horizontal line drawn at the 1.0 mark. In Sliding Batch and Restrained Sliding Batch, through the

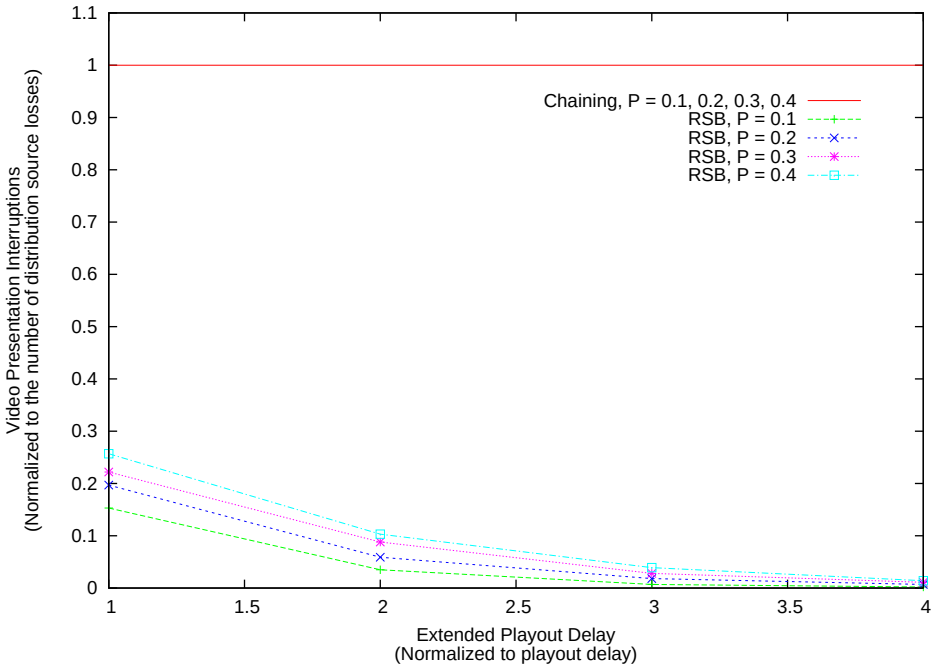


Figure 7.6: Effects of extended playout delay - I

implementation of proposed QoS scheme, not all distribution source losses being detected may result in the interruption of the video presentation. The user may potentially experience interruptions only if a downloading of a segment encounters a greater number of distribution source losses than all other segments in the same batch (i.e. maximum distribution source losses of a batch, $L_{\beta(e_i)}$). Further more, a video presentation interruption can occur only if the sum of the maximum distribution source losses of all batches results in the accumulated delay beyond the extended playout delay. Let Γ be the total number of video presentation interruptions being experienced by a user for the duration of the video playback. Γ is defined as:

$$\Gamma = \left\lceil \frac{\sum_i^N L_{\beta(e_i)} \times playouot\ delay}{extended\ playout\ delay} \right\rceil$$

In Restrained Sliding Batch (RSB), at 1× playout delay, 15% to 26% of distribution source losses being detected resulted in actual video presentation interruptions when 10% to 40% of nodes prematurely depart from the VT Room. At extended playout delay of 2×

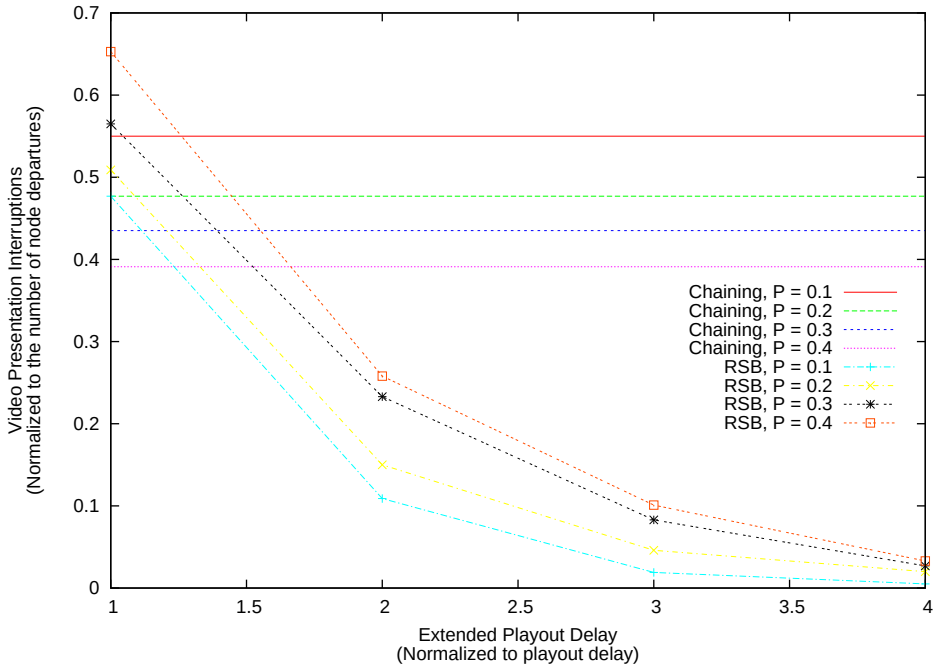


Figure 7.7: Effects of extended playout delay - II

playout delay, the rate of video presentation interruptions decreased to 4% to 10% when P is varied from 0.1 to 0.4. When the size of extended playout buffer size is increased to 4× playout delay, less than 1% of all distributions source losses being detected by users resulted in the actual video presentation interruptions when 10% of total nodes prematurely leave the VT Room. At 40% of premature node departure rate, roughly 1% of detected source losses resulted in video presentation interruptions.

Another way to express the effects of extended playout delay on the video presentation interruptions under different node departure rates is shown in Figure 7.7. In this figure, the total number of video presentation interruptions is normalized to the total number of premature node departures under the same settings as the previous set of experiments. Note, in Sliding Batch and Restrained Sliding Batch, a premature user departure may result in multiple instances of distribution source losses being experienced by other users. Figure 7.7 shows, on average, how many instances of video presentation interruptions are introduced when one node departs prematurely from the service. In Restrained Sliding Batch, every

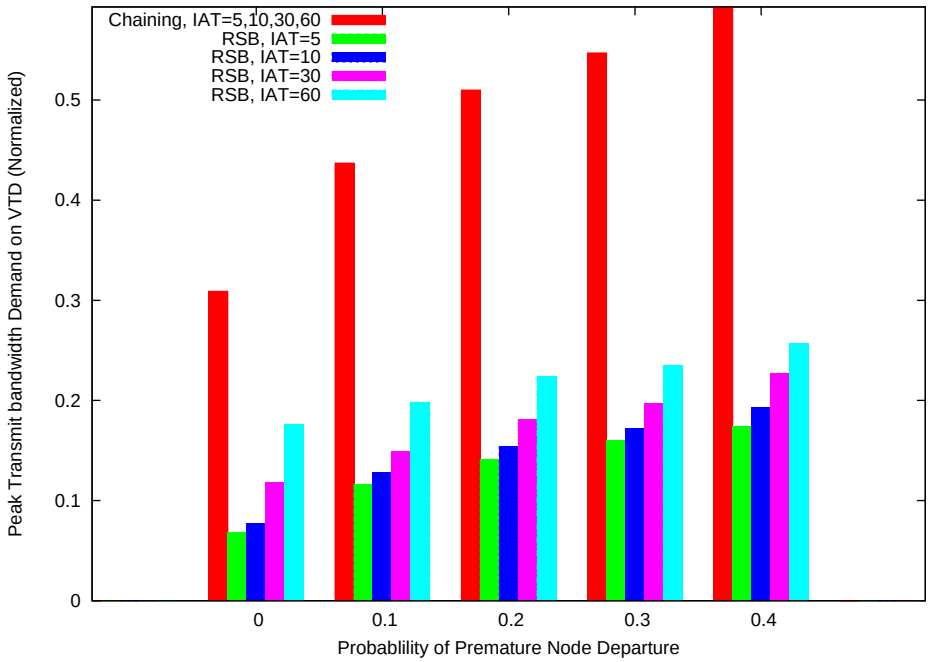


Figure 7.8: Effects of node departure rate

premature node departure resulted in a video presentation interruption at the rate of 48% to 65% at 1× playout delay when P is varied from 0.1 to 0.4. When the size of extended playout buffer is doubled, the interruption rates have halved at $P = 0.4$ and quartered at $P = 0.1$. When the extended playout delay is at 4× playout delay, less than 1% of premature node departure resulted in a video presentation interruption when $P = 0.1$ and roughly 3% of of premature node departure resulted in a video interruptions when 40% of users depart from the service prematurely.

In Chaining, every premature node departure resulted in a video presentation interruption at the rate of 55% to 39% when P is varied from 0.1 to 0.4. The first look of the simulation result is counter intuitive in that the rate of video interruptions decreases as the rate of node departure increases. This is because, in Chaining, as the rate of node departure increases, a large percentage of users begin relying on the central server for video distribution feeds, rather than their peers. The next set of experiments proves this point.

The last set of experiments studies the effects of premature node departure rate on the

load on the VT Distributor under varied user arrival rate to the VT Room. The results are shown as bar graphs in Figure 7.8. As before, the peak transmit bandwidth demand on VT Distributor is normalized to the peak transmit bandwidth demand on a traditional client-server based video distribution network. At $P = 0$, no node prematurely departs from the VT Room and the results gained this simulation are used as reference. For $0.1 \leq P \leq 0.4$, both Chaining and Restrained Sliding Batch have the similar rate of load increase on VT Distributor as the rate of premature node departure increases. The main difference between the two schemes is that Restrained Sliding Batch with QoS extension requires only a third or less of resources from the central server compared to the Sliding Batch. Another difference is that the Chaining did not exhibit statistical differences when varying the arrival rate of users where as Restrained Sliding Batch clearly performed better when more users arrive at the VT Room than otherwise.

This section presented the design and analysis of simulation studies. Seven sets of experiments were designed to study the behavior of users in a VT Room. The first four sets of experiments focused on the load on the VT Distributor while varying various parameters of Sliding Batch. The simulation results showed a trend in the load reduction in the VT Distributor as the rate of user arrival increased. This is an indication that a scalable stored video distribution network may be built on Virtual Theater Network and accompanying distribution and discovery schemes. The last three sets of experiments verified the effectiveness of QoS scheme in reducing the number of video presentation interruptions when users depart from the network prematurely. Linearly increase in the size of playout buffer resulted in logarithmic decrease in the rate of video presentation interruptions. Further more, relatively small increase in the VT Distributor load was observed when the probability of premature node departure was raised. These simulation results suggest that Sliding Batch QoS scheme has a potential in reducing the impacts of excessive delays on the presentation of video to receiving users.

8.0 CONCLUSION

This research work aims to enable distribution of high-volume, on-demand, stored-video distribution service to a large number of users with diverse system needs.

To address these challenges, we proposed the design of a new streaming video distribution network model, called *Virtual Theater Network*, which allows organization of peer-to-peer communities (*VT Rooms*) to support the distribution of streaming videos among the members in each VT Room. The model employs a *segmented* video stream reception scheme with its accompanying scheduling algorithm for an orderly and timely video segment retrieval that allows contribution from users with limited resource availability. QoS extension of the distribution scheme allows reduction in the number of video presentation interruptions when excessive delays are observed. The model also employs a video segment availability advertisement and discovery scheme which incorporates the parameters of segmented video reception scheduling algorithm and made possible the advertisement and query of ever changing segment availability information of each user in one advertisement and one query. The distribution source selection scheme allows incorporation of mobility information to select a source who could best satisfy the user needs.

The seven sets of experiments were conducted to study the feasibility of the proposed network architecture in supporting a large-scale stored-video streaming service and to verify the effectiveness of the proposed distribution scheme and its QoS extension. The first four sets of experiments focused on the scalability aspect of the network design and measured the peak demand on the transmit bandwidth usage at VT Distributor. The simulation results showed a decline in the load demand at the VT Distributor as the user arrival rate increased, which is a sign of scalability. The remaining experiments focused on the resiliency aspect of the network design and studied the effects of premature node departure from the service on

the number of video presentation interruptions at the receiving users. By increasing the size of playout buffer, a significant decrease in the number of video presentation interruptions were observed through simulation study, including cases where more than a third of the participating users were prematurely departing from the service.

8.1 FUTURE WORK

Several areas to which this research study can be extended as future work. Incorporation of security and digital right management features in the advertisement and distribution schemes to allow protection of network service from malicious users and copy righted materials from illegal usage. These are important issues in P2P based content distribution network as there has not been a practical solution available. Improvements in video distribution efficiency and scalability through VT Room merges/splits and load balancing among VT Distributors are another area of future study. Lastly, as there has been growing interests in the delivery of TV signals over IP based network (i.e. IPTV), a need exists that allows distribution of both the real-time and the stored video streams in a single network. Addition of a real-time streaming support on Virtual Theater Network is another area of future work.

APPENDIX A

NOTATIONS

Symbol	Meaning
V	Video
S_i	i_{th} segment
f_i	i_{th} frame
η	Nominal streaming rate
$\alpha(S_i)$	Starting playback time of S_i
$\delta(S_i)$	Playback duration of S_i
$\|x\|$	Size of x (e.g. segment, buffer) in number of frames
$\|\|y\|\|$	Size of y (e.g. segment, buffer) in number of bits
e_i	i_{th} epoch
$\alpha(e_i)$	Starting epoch time of e_i
$\delta(e_i)$	Epoch duration of e_i
$\beta(e_i)$	Batch associated with e_i
$A(S_i)$	Starting download time of S_i
$\Omega(S_i)$	Ending download time of S_i
$\overline{N}(e_i)$	Number of segments yet to be received at $\alpha(e_i)$
$\|\beta_R(e_i)\|$	Rate-limited batch size at $\alpha(e_i)$
$\|\beta_B(e_i)\|$	Buffer-limited batch size at $\alpha(e_i)$

Symbol	Meaning
R_T	Total receive bandwidth
$R_A(e_i)$	Available receive bandwidth at $\alpha(e_i)$
$R_U(e_i)$	Used receive bandwidth during e_i
B_T	Total buffer space
B_D	Buffer space allocated for segment downloading and playback
B_H	Buffer space allocated for segments that finished playing back
$B_A(e_i)$	Available buffer space at $\alpha(e_i)$
$B_U(e_i)$	Used buffer space during e_i
$T_A(e_i)$	Available transmit bandwidth at $\alpha(e_i)$
$\overline{d_{i,j}}(t_1, t_2)$	Mean distance between node i and j during time t_1 and t_2
$\Delta(S_i)$	Time it takes to download S_i
$d_{i,j}(t)$	Distance between node i and j at t
D_p	Playout delay (start-up latency)
r_i	Downloading rate of S_i
D_{En}	Start-up latency introduced by providing the n_{th} level of protection against excessive delay
r_i^{En}	Downloading rate of S_i by providing the n_{th} level of protection against excessive delay
L_i	Maximum number of distribution source losses a user may anticipate
P_i	Probability of distribution source loss when downloading S_i

APPENDIX B

VIDEO SEGMENT RECEPTION SCHEDULING ALGORITHM

```
// a scheduler template
typedef struct {
  int user_id;
  int seg_id;
  int batch_id;
  double seg_size;

  // targeted or nominal downloading times
  double t0;
  double start_dl_tm;
  double end_dl_tm;
  double dl_rate;

  // targeted or nominal playback times
  double start_pb_tm;

} segment_reception_info_t;

// data structure for the scheduling of segment receptions
segment_reception_info_t _seg_rcpt_info[MAX_MUM_OF_SEGMENTS];

// initialized at the beginning
double _RT;   // total receive bandwidth (bps)
double _RA;   // available receive bandwidth (bps)
double _BD;   // downloading buffer (in number of bits)
double _t0;   // the time a user joined the VT Room

User::_scheduling_algorithm()
{

  double RA, RU = 0.0, BA, BU = 0.0;
  double RU_in_this_batch, BU_in_this_batch;
```

```
33    int i, j = 0, m_i = -1;
34
35    // figure out the size of batch at each epoch
36    for (i = 0; i < num_of_epochs; i++) {
37
38      // determine the rate-limited batch size
39
40      // find RA
41      if (i == 0)
42        RA = _RT;
43      else {
44        RA = _RT - RU + _seg_rcpt_info[i-1].dl_rate;
45      }
46
47      // find RU
48      if (i == 0) {
49        for (j = 0; j < num_of_segments; j++) {
50          double dl_rate = nominal_rate / (j + 1);
51          if (RU + dl_rate > RA) {
52            break;
53          } else {
54            RU += dl_rate;
55          }
56        }
57      } else {
58        RU_in_this_batch = 0.0;
59        for (j = m_i + 1; j < num_of_segments; j++) {
60          double end_pb_tm = _t0 + seconds_per_segment * (j + 1);
61          double beg_dl_tm = _t0 + seconds_per_segment * i;
62          double dl_rate = bits_per_segment / (end_pb_tm - beg_dl_tm);
63          if (RU_in_this_batch + dl_rate > RA) {
64            break;
65          } else {
66            RU_in_this_batch += dl_rate;
67          }
68        }
69        RU = _RT - RA + RU_in_this_batch;
70      }
71
72      j--;
73      int rate_limited_batch_size = j - m_i;
74
75      // figure out the buffer-limited batch size
76
77      // find BA
78      if (i == 0)
79        BA = _BD;
```

```
80      else {
81        BA = _BD - BU + bits_per_segment;
82      }
83
84      // find BU
85      if (i == 0) {
86        for (j = 0; j < num_of_segments; j++) {
87          if (BU + bits_per_segment > BA) {
88            break;
89          } else {
90            BU += bits_per_segment;
91          }
92        }
93      } else {
94        BU_in_this_batch = 0.0;
95        for (j = m_i + 1; j < num_of_segments; j++) {
96          if (BU_in_this_batch + bits_per_segment > BA) {
97            break;
98          } else {
99            BU_in_this_batch += bits_per_segment;
100         }
101       }
102       BU = _BD - BA + BU_in_this_batch;
103     }
104
105     j--;
106     int buffer_limited_batch_size = j - m_i;
107
108     // min(rate_limited_batch_size, buffer_limited_batch_size)
109     int batch_size;
110     (rate_limited_batch_size < buffer_limited_batch_size)
        ? batch_size = rate_limited_batch_size : batch_size
        = buffer_limited_batch_size;
111
112     // min(N_bar, min(rate_limited, buff_limited))
113     if (num_of_segments < m_i + batch_size + 1)
114       batch_size = num_of_segments - m_i - 1;
115
116
117     // initialize RU and BU for this batch
118     RU_in_this_batch = 0.0;
119     BU_in_this_batch = 0.0;
120
121     // compute start downloading time of each segment
122     for (j = m_i + 1; j <= m_i + batch_size; j++) {
123
124       _seg_rcpt_info[j].user_id = _user_id;
```

```
125         _seg_rcpt_info[j].seg_id = j;
126         _seg_rcpt_info[j].batch_id = i;
127         _seg_rcpt_info[j].seg_size = bits_per_segment;
128         _seg_rcpt_info[j].t0 = _t0;
129         _seg_rcpt_info[j].start_dl_tm = _t0 + i * seconds_per_segment;
130         _seg_rcpt_info[j].start_pb_tm = _t0 + j * seconds_per_segment;
131         _seg_rcpt_info[j].end_dl_tm = _seg_rcpt_info[j].start_pb_tm +
            seconds_per_segment;
132         _seg_rcpt_info[j].dl_rate = bits_per_segment /
            (_seg_rcpt_info[j].end_dl_tm - _seg_rcpt_info[j].start_dl_tm);
133
134         // update RU_in_this_batch and BU_in_this_batch
135         RU_in_this_batch += _seg_rcpt_info[j].dl_rate;
136         BU_in_this_batch += _seg_rcpt_info[j].seg_size;
137       }
138
139       // compute RU and BU
140       RU = _RT - RA + RU_in_this_batch;
141       BU = _BD - BA + BU_in_this_batch;
142
143       // bump up m_i
144       m_i += batch_size;
145     }
146   }
```

BIBLIOGRAPHY

[1] A. Aggarwal and M. Rabinovich, "Performance of dynamic replication schemes for an internet hosting service," AT & T Labs, Tech. Rep., October 1998. [Online]. Available: citeseer.ist.psu.edu/aggarwal98performance.html

[2] C. Aggarwal, J. Wolf, and P. Yu, "A permutation-based pyramid broadcasting scheme for videoon -demand systems," in *IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[3] American National Standard for Telecommunications, "Telecom Glossary 2000 T1.523-2001," 2001.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," UMIACS TR-2002," Tech. Rep., 2002. [Online]. Available: citeseer.ist.psu.edu/banerjee02scalable.html

[5] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," in *Proceedings of IEEE INFOCOM*, 2003.

[6] T. C. Chiueh and C. H. Lu, "A periodic broadcasting approach to video-on-demand service," in *Proceedings of SPIE*, 1996, pp. 162–169.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science*, vol. 2009, pp. 46+, 2001. [Online]. Available: citeseer.ist.psu.edu/clarke00freenet.html

[8] E. Cohen, H. Kaplan, and A. Fiat, "Associative search in peer to peer networks: Harnessing latent semantics," in *Proceedings of IEEE INFOCOM*, Apr. 2003. [Online]. Available: citeseer.ist.psu.edu/562084.html

[9] B. Cooper and H. Garcia-Molina, "Ad hoc, self-supervising peer-to-peer search networks," Stanford University," Technical Report, Feb. 2003.

[10] K. Delgadill, "Distributed Director. White paper," Cisco Sytems, Inc. http://www.cisco.com/, 1999.

[11] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over peers," CS Dept., Stanford University, Tech. Rep. 2001-31, 2001.

[12] P. Ganesan, Q. Sun, and H. Garcia-Molina, "YAPPERS: A peer-to-peer lookup service over arbitrary topology," in *Proceedings of IEEE INFOCOM*, 2003. [Online]. Available: citeseer.ist.psu.edu/ganesan03yappers.html

[13] Gnutella, "http://gnuella.wego.com."

[14] L. Golubchik, J. C. S. Lui, and R. R. Muntz, "Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers," *Multimedia Systems*, vol. 4, no. 3, pp. 140–155, 1996. [Online]. Available: citeseer.ist.psu.edu/article/golubchik96adaptive.html

[15] K. Hua, D. Tran, and R. Villafane, "Caching multicast protocol for on-demand video delivery," in *ACM/SPIE Conference on Multimedia Computing and Networking*, 2000. [Online]. Available: citeseer.ist.psu.edu/hua00caching.html

[16] K. Hua and D. A. Tran, "Range multicast for video on demand," *Journal of Multimedia Tools and Applications*, 2003. [Online]. Available: citeseer.ist.psu.edu/hua03range.html

[17] K. Hua, D. A. Tran, and R. Villafane, "Overlay multicast for video on demand on the internet," in *Proceedings of ACM SIGAPP Symposium on Applied Computing (SAC 2003)*, March 2003. [Online]. Available: citeseer.ist.psu.edu/hua02overlay.html

[18] K. A. Hua, Y. Cai, and S. Sheu, "Patching : A multicast technique for true video-on-demand services," in *Proceedings of ACM Multimedia*, 1998, pp. 191–200. [Online]. Available: citeseer.ist.psu.edu/hua98patching.html

[19] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *SIGCOMM*, 1997, pp. 89–100. [Online]. Available: citeseer.ist.psu.edu/hua97skyscraper.html

[20] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in *INFOCOM*, 2001, pp. 31–40. [Online]. Available: citeseer.ist.psu.edu/jamin01constrained.html

[21] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., "Overcast: Reliable multicasting with an overlay network," in *In Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, October 2000, pp. 197–212. [Online]. Available: citeseer.ist.psu.edu/jannotti00overcast.html

[22] M. Kalman, E. Steinbach, and B. Girod, "Adaptive playout for real-time media streaming," in *IEEE International Symposium on Circuits and Systems, ISCAS-2002*, May 2002.

[23] KaZaA, "http://www.kazaa.com."

[24] P. Liu, S. Battista, F. Casalino, and C. Lande, "MPEG-4: a multimedia standard for the third millennium, part 1," *IEEE Multimedia*, October 1999.

[25] Morpheus, "http://www.morpheus.com/."

[26] Olson, Camarillo, and Roach, "Support for IPv6 in Session Description Protocol (SDP)," *RFC 3266*, June 2002.

[27] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of ACM/IEEE NOSSDAV*, 2002. [Online]. Available: citeseer.ist.psu.edu/padmanabhan02distributing.html

[28] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, pp. 40–48, Sep/Oct 1998. [Online]. Available: citeseer.ist.psu.edu/perkins98survey.html

[29] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *INFOCOM*, 2001, pp. 1587–1596. [Online]. Available: citeseer.ist.psu.edu/qiu01placement.html

[30] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proceedings of IEEE INFOCOM'02*, 2002. [Online]. Available: citeseer.ist.psu.edu/ratnasamy02topologicallyaware.html

[31] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proceedings of ACM SIGCOMM*, 2001. [Online]. Available: citeseer.ist.psu.edu/ratnasamy01scalable.html

[32] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001, pp. 329–350.

[33] Schulzrinne, Casner, Frederick, and Jacobson, "RTP: A transport protocol for real-time applications," *RFC 3550*, July 2003.

[34] S. Sheu and K. A. Hua, "Virtual batching: A new scheduling technique for video-on-demand servers," in *Database Systems for Advanced Applications*, 1997, pp. 481–490. [Online]. Available: citeseer.ist.psu.edu/sheu97virtual.html

[35] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-topeer systems," in *Proceedings of IEEE INFOCOM*, Apr. 2003. [Online]. Available: citeseer.ist.psu.edu/sripanidkulchai03efficient.html

[36] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001, pp. 149–160. [Online]. Available: citeseer.ist.psu.edu/stoica02chord.html

[37] W. tian Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Transactions on Multimedia*, vol. 1, no. 2, pp. 172–186, 1999. [Online]. Available: citeseer.ist.psu.edu/tan99realtime.html

[38] D. Tran, K. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *Proceedings of IEEE INFOCOM*, 2003. [Online]. Available: citeseer.ist.psu.edu/tran03zigzag.html

[39] A. Vetro, C. Christopoulos, and huifang Sun, "Video Trnascoding Architectures and Techniques: An Overview," *IEEE Signal Processing Magazine*, March 2003.

[40] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On peer-to-peer media streaming," Purdue Computer Science," Tech. Rep., April 2002. [Online]. Available: citeseer.ist.psu.edu/xu02peertopeer.html

[41] Z. Xu, M. Mahalingam, and M. Karlsson, "Turning heterogeneity into an advantage in overlay routing," in *Proceedings of IEEE INFOCOM 2003*, 2003. [Online]. Available: citeseer.ist.psu.edu/574255.html

[42] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," UC Berkeley, Tech. Rep. UCB/CSD-01-1141, Apr. 2001. [Online]. Available: citeseer.ist.psu.edu/zhao01tapestry.html