IMPROVING STREAMING CAPACITY IN TREE & MESHED BASED P2P LIVE STREAMING SYSTEMS

by

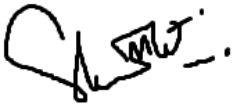
Shujjat Ahmed Khan

A Thesis
Presented to the School of Graduate Studies at
Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in the Program of Computer Networks
Department of Electrical and Computer Engineering
Toronto, Ontario, Canada, August 2012

Author's Declaration

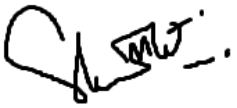
I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.



Author's Signature:_____

I further authorize Ryerson University to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.



Author's Signature:_____

Abstract

IMPROVING STREAMING CAPACITY IN TREE & MESHED BASED P2P LIVE STREAMING SYSTEMS

Master of Applied Science
Computer Networks Program
Department of Electrical and Computer Engineering
Ryerson University

The streaming capacity for a channel is defined as the maximum streaming rate that can be achieved by every user in the channel. In the thesis, we investigated the streaming capacity problem in both tree-based and mesh-based Peer-to-Peer (P2P) live streaming systems, respectively. In tree-based multi-channel P2P live streaming systems, we propose a cross-channel resource sharing approach to improve the streaming capacity. We use cross-channel helpers to establish the cross-channel overlay links, with which the unused upload bandwidths in a channel can be utilized to help the bandwidth-deficient peers in another channel, thus improving the streaming capacity. In meshed-based P2P live streaming systems, we propose a resource sharing approach to improve the streaming capacity. In mesh-based P2P streaming systems, each peer exchanges video chunks with a set of its neighbors. We formulate the streaming capacity problem into an optimal resource allocation problem. By solving the optimization problem, we can optimally allocate the link rates for each peer, thus improve the streaming capacity.

Acknowledgement

First of all I would like to thanks to almighty Allah for completing my thesis work, and everything I achieved in my whole life, and second I would like to convey my appreciation from the bottom of heart to my respected supervisor, Dr. Yifeng He, it was my pleasure and his greatness who accepted me to work with him, I am thankful for his continuous encouragement, tremendous technical guidance and supports throughout this research. It was a great and remarkable opportunity for me to work with him, if I would be given another chance in future I must prefer to work with him again.

I also would like to thank Dr. Bobby Ma for his advice and guidance in different steps of my work. His great and precious knowledge and experience was an enormous facilitated and privilege for me. I would like to thank other professors, namely Dr. Jaseemuddin, Dr. Woungang and Prof. Raj for their excellent teaching throughout my degree.

I would like to acknowledge the Computer Networks Department and the School of Graduate Studies at Ryerson University for their support in terms of financial aid, and work experience as a graduate assistant.

I can never find the words to thanks to my beloved mother and late father, without their struggle, hard work and prayers, I would have never reach my current stage in life, special thanks to my brothers and sisters, and last but not least my lovely wife and my cute three children without their continuous sacrifices, moral support and encouragement I would never complete my degree and this thesis.

Shujjat Ahmed Khan

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 P2P Streaming System

Today, Peer-to-Peer (P2P) has become one of the most extensively discussed technology in information technology. With the extensive availability of low-cost broadband Internet connections for home-users, a great number of bandwidth-intensive applications have now become realistic with P2P, which were not practical few years back. The term P2P refers to the concept that peers in a network using appropriate information and communication systems, two or more individuals are able to impulsively work together without necessarily needing central management. P2P networks promise improved scalability, lower cost of ownership, self-organized and decentralized coordination. The streaming services originally worked in the client/server architecture. A client/server arrangement involves multiple clients connecting to a single, central server. The client/server architecture enables the roles and responsibilities of a computing system to be distributed among several independent clients. Network congestion of the traffic is one of the problems related to the client/server model. As the number of simultaneous client requests to a given server increases, the server can become overloaded, and this centralized architecture cannot provide streaming to a large number of users due to the

limited and expensive upload bandwidth from the server. P2P technology has recently become a capable approach to provide live and on-demand video streaming services over the Internet at low cost. Commercial P2P live streaming and Video-On-Demand (VoD) systems, such as PPLive [1], PPStream [2], UUSee [3], have been successfully supporting tens of thousands of users. Apart from the large number of users, these systems have a common feature of providing a large number of channels for users to watch, and hence are referred to as multi-channel P2P streaming systems. P2P streaming systems can be categorized into two major branches, P2P VoD System and P2P live streaming system, in P2P VoD systems, where the users in the same channel may watch different positions of the video at any time, and in P2P Live streaming system, where the users in the same channel watch almost the same position of the video.

1.2 P2P VoD Streaming Systems

P2P systems has recently gained a lot of attractiveness and accomplishment in the commercial sector where they are now able to broadcast to multi-millions users at any given time. A VoD system is suitable for media files that can be reproduced at any moment, VoD system allows users to select and watch a video content on demand. VoD system provides the user a VCR (Video Cassette Recorder) functionality including pause, forward, rewind, jump to previous/further frame etc [47]. In P2P VoD system, an overlay is established among server and users/peers, liveness is merely inappropriate in VoD systems because the video stream is already pre-recorded, peers pull the video content from the server and neighboring peers, VoD systems are also appropriate for media files that can be frequently watch any time, for example Youtube.com, which is successfully supporting tens of thousands of users. In a P2P VoD system, users in the same channel may watch the same content from any special point, users can even choose any video they like and start to watch it at any time. A P2P VoD streaming system provides more flexibility and interactivity to users. Depending on the forwarding approaches, P2P VoD systems can be categorized into buffer-forwarding systems, storage-forwarding systems, and hybrid-forwarding systems.

Streaming capacity, which is our main topic has been studied in [40][41][42] for a single channel

in P2P VoD systems. In [40][42], the streaming capacity for a single channel is formulated into an optimization problem which maximizes the streaming rate under the peer bandwidth constraints. The throughput maximization problem viewed as a scalable P2P VoD system is studied in [43]. A Peer has an un-used available bandwidth is called a rich peer, rich peers in terms of available bandwidth helpers have been proposed in P2P systems to improve the system performance [41][44][45], a rich peer chosen by server for cross-channel resource sharing is called a helper peer. In P2P VoD systems, each additional helper increases the system upload capacity, thus offloading the server burden [45]. In [41], algorithms on helper assignment and rate allocation are proposed to improve the streaming capacity for P2P VoD systems.

In a VoD system, playbacks of the video watching by different users are actually not synchronized, synchronization refers to the continuation of the stream of the video, that means in VoD system a user can play and stop a video stream anytime. However in P2P live streaming system which is our main topic, users are synchronized by watching the same position of the stream almost at the same time.

1.3 P2P Live Streaming Systems

The traditional client/server model used to be the most suitable for live streaming purpose. In that client/server model, the server must directly provide the stream to all of its clients. However, this centralized model has not much capability to provide live streaming to a large number of users due to the limited and expensive upload bandwidth from the server. A P2P live streaming involves a continuous flow of information that is apparently being generated at that instant and is thus bound to be immediately consumed. In a P2P live streaming system, a live video is disseminated to all users in real time. The video playbacks on all users are synchronized, in this P2P live streaming system, each user watches almost the same position of the video, and any peer can be a parent peer of its neighbor peer/peers in a tree-based systems, a peer serves a stream to other peers is called a parent peer in a tree. In a P2P live streaming systems, user joining an on-going live streaming session, and starts watching the stream from his/her joining time.

P2P systems have been successfully supporting tens of thousands of users. Apart from the large number of users, P2P systems have a feature to provide a large number of channels for users to watch, and hence are referred to as multi-channel P2P streaming systems. In recent years, there has been several major industrial consumption of P2P live streaming systems, including PPLive [1], PPStream [2] and UUSee [3], these P2P streaming systems have begun to support multiple channels and a user in such systems is allowed to watch these live streams. Almost all P2P live streaming systems offer multiple channels. PPLive [1] and its contender each have over 100 channels, future systems with user-generated channels will likely have thousands of live channels. Based on its overlay structure, P2P live streaming system can be generally classified into tree-based P2P live streaming systems and mesh-based P2P live streaming systems.

1.3.1 Tree-based P2P Live Streaming System

In tree-based P2P live streaming systems, the video streams are delivered over a single application-layer tree or multiple application-layer trees.

1.3.1.1 Single Tree-Based P2P Live Streaming Systems

In a Single-Tree based P2P live streaming systems, at the application layer, peers are organized into a virtual topology, called an overlay, where each peer maintains a set of virtual links with other peers watching the same channel, similar to IP multicast.

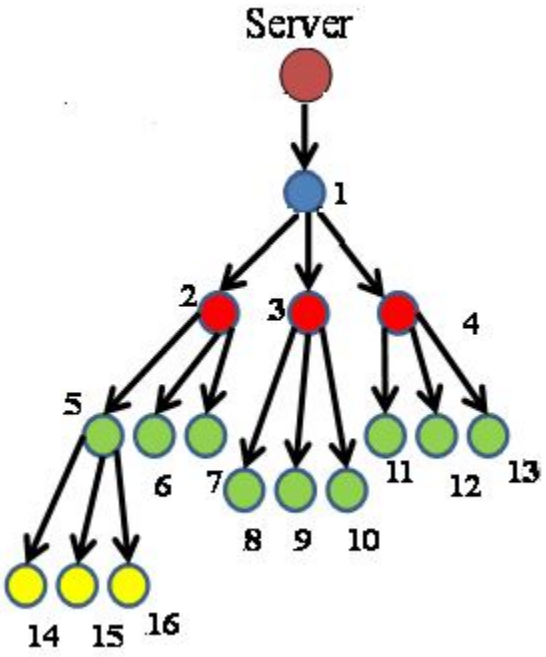


Fig. 1.1: Illustration of a single-tree based P2P live streaming system

In Single-Tree Based P2P live streaming systems, the root of the tree is the server. Each peer joins the tree at a certain level. It receives the video stream from its parent peer at the level above and uploads the received video to its child peers at the level below, a peer gets a stream from the parent peer or server is called a child peer in P2P live streaming system. The single tree-based topology has two setbacks:

1) The upload bandwidth of leaf nodes/ peers in an overlay cannot be utilized by others since they do not have their own child peers; and

2) A peer leaving an overlay network suffers a connection failure to its child peers.

1.3.1.2 Multiple-Tree Based P2P Live Streaming System

In Multiple-Tree based P2P live streaming systems, the video is encoded into multiple sub-streams, and each sub-stream is delivered over one tree. The quality received by a peer depends on the number of sub-streams that it receives.

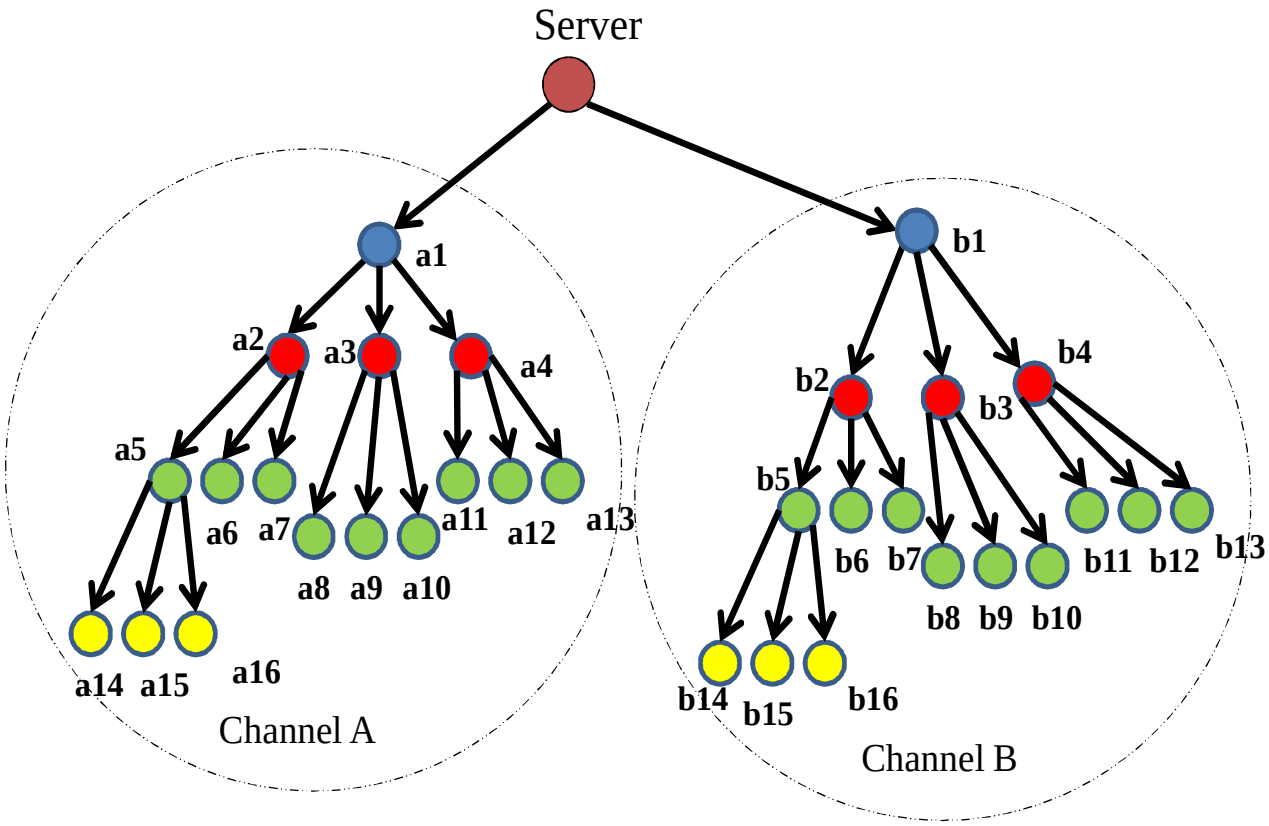


Fig. 1.2: Illustration of a multiple-tree based P2P live streaming

There are two major advantages for the multiple-tree based P2P live streaming systems. First, if a peer fails or leaves, all its children lose the sub-stream delivered from that peer, but they still receive the sub-streams delivered over the other trees. Therefore, all its children can watch the same video in case of a loss of a sub-stream. Second, a peer has diverse roles in different trees. It might be an internal node in one tree and a leaf node in another tree. When a peer is an internal node in a tree, its upload bandwidth will be utilized to upload the sub-stream delivered over that tree. To achieve high bandwidth utilization, a peer with a high upload bandwidth can supply sub-streams in more trees.

1.3.2 Mesh-based P2P Live Streaming System

In single-tree based P2P live streaming systems, a peer receives the streams from a single parent. If the peer's parent leaves, the peer will receive nothing until it is connected to a new parent. In multiple-tree based P2P live streaming systems, a peer receives a sub-stream from the parent in the corresponding tree. If the parent in the tree leaves, the received stream quality at the peer will be ruined. Therefore, peer dynamics make tree maintenance a challenging and costly task in tree-based P2P live streaming systems. In mesh-based P2P live streaming systems, each peer exchanges the data with a set of neighbors, each peer maintains several neighbors in the system and there is no strict parent child affiliation between any two connected peers, which specified that the two peers can download data from each other. Since a peer can download the data from multiple neighbors, the mesh-based systems are flexible to peer churn, the major advantage of mesh-based P2P live streaming systems is, if one neighbor leaves, the peer can still download the video stream from the remaining neighbors.

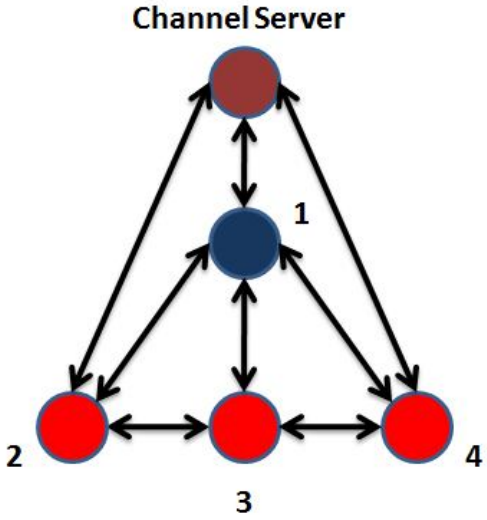


Fig. 1.3: Illustration of Data-driven P2P live streaming System

Meanwhile, the peers will add other peers into its neighbor set. However, this is not possible in single-tree based live streaming systems. Thus mesh-based streaming systems are robust against peer dynamics. The major challenges in mesh-based P2P live streaming systems are neighborhood formation and data scheduling.

1.4 Motivation and Contribution

The maximum streaming rate that can be received by every user in a channel is defined as the streaming capacity of the channel in a multi-channel P2P live streaming system [5]. The streaming capacity can served as the indicator of video quality for a channel. A higher streaming capacity for a channel means that the users in the channel can receive a higher video quality. Therefore, the objective of this thesis is to achieve a high streaming capacity for a channel in P2P live streaming systems in Tree-based and Meshed-based P2P live streaming systems.

In Chapter 3, we study the multi-channel P2P live streaming systems, where we proposed a cross-channel resource sharing approach to get the better streaming capacity. We utilized cross-channel helpers to create the cross-channel overlay links, with which the idle or unused upload bandwidths in a channel can be consumed to help the bandwidth-deficient peers in another channel, thus improving the streaming capacity. We simulated the proposed cross-channel resource sharing approach, and showed that it can result in a better streaming capacity in tree-based multi-channel P2P live streaming systems. We also compared the streaming capacity with/ without cross-channel resource sharing. The proposed cross-channel resource sharing better utilizes the upload bandwidths.

In Chapter 4, we extend our work to mesh-based P2P live streaming systems, where we developed another simulation to demonstrate the single channel meshed-based or data-driven P2P live streaming systems enhancement of the streaming capacity by selecting a better list of neighbor resource sharing. In that same chapter, we review the problem formulation of the streaming capacity problem introduced in [5], then we proposed a optimal allocation scheme to solve the problem.

1.5 Technical Challenges

It is challenging to achieve a high streaming capacity in multi-channel P2P live streaming

systems. The streaming capacity for a channel is dependent on the bandwidth capacity of each peer in the channel and the overlay structure of the channel. A higher streaming capacity can be achieved by optimizing the resources among the peers within the same channel, which is, however, a challenging task. Peers have heterogeneous bandwidths. Some peers may have deficient upload bandwidths, which limit the streaming capacity, while some other peers may have abundant upload bandwidths which have not been fully utilized. Resource sharing in a multi-channel P2P live streaming system is expected to improve the streaming capacity. However, resource sharing in multi-channel P2P live streaming systems is difficult to achieve due to the following reasons.

1) In a P2P live streaming system, each peer has different upload and download capacity, and each peer may join or leave the channel at any time. The heterogeneous characteristics and dynamic behaviors of the peers cause dynamic resource distribution among peers and channels.

2) It is difficult to shift the unused resources in a channel to improve the streaming capacity for another channel, since there is originally no overlay connection between the two channels.

In Mesh-based P2P Live streaming systems, neighbor selection is a challenging issue. A better neighbor list helps to improve the data exchange between the peer and its neighbors. The decision of the neighbor relationship is made based on the following considerations [34]:

1) The available resources on both peers, such as the number of connections, upload bandwidth, download bandwidth, CPU and memory usage;

2) The quality of the potential overlay link between the two peers, which can be characterized by the transmission delay and packet loss rate;

3) The accompaniment of the video content, which means that the video blocks available at a peer are needed by the other peer and vice versa.

1.6 Thesis Organization

The rest of the thesis is organized as follows: In Chapter 2, a review of related existing work in the literature is provided. Problem definition and formulation are presented in Chapter 3 and 4 with detailed explanations and performance evaluation is provided there. Finally, chapter 5 concludes our work and points to some future directions of the research.

Background knowledge, comparisons and overview with existing and proposed works are covered in each Chapter, when necessary, that's how each chapter can be read independently and understand by individual thoroughly without reading the whole paper work.

Chapter 2

Literature Review

2.1 Introduction

The work proposed in this thesis (Chapters 3 and 4) lie in two different arenas of P2P live streaming systems; Tree-based and Mesh-based P2P live streaming. In this chapter, we discussed others works that are related to the aforementioned research areas.

High user demand for these P2P systems has been shown by their increasingly large user support [48][49]. Not surprisingly, current studies shown that over 60% of Internet traffic is generated by P2P systems [54], with video accounting for more than one-third of all Internet traffic today [55][56]. P2P streaming can be divided into two main approaches, tree-based [52][50][57][58] and mesh-based [51][53] architectures. Tree-based overlays construct a tree, rooted at the source, which broadcasts the stream. Mesh-based overlays distribute data in a less structured manner, where nodes or peers exchange data with a subset of the nodes in the network without using any pre-defined tree.

2.2 Tree-Based P2P Live Streaming System

In a tree-based system, peers are connected and organized into a streaming tree where the server serves as the root. Each peer only downloads from its parent node and uploads to its children. Therefore, the tree-based systems inherently require a push-based data driven method [12] shows the structure of a single tree-based P2P streaming system. There are two major drawbacks of the single-tree based approach. First, the maintenance is complex and resource consuming. Moreover, it is difficult for leaf nodes to contribute their upload bandwidth. Therefore the multi-tree structure is introduced, in which a peer is enrolled in different streaming trees with different positions.

In early days, tree-based systems such as Overcast [21] and ESM [22] were proposed using application level multicast. However, a theoretical analysis of tree-based P2P video streaming model was proposed only after the year 2007 [11] [13] [20], deriving theoretical results for system performance with or without node degree bound. Due to its complexity and severity, tree-based approaches are then replaced by the mesh-based technique. Nowadays commercial P2P live streaming applications like PPLive [1] and PPstream [2] mainly adopt this approach for its robustness and simplicity to implement and maintain.

Cross-channel resource sharing has been recently studied in multi-channel P2P streaming systems [7][30][31][32][46]. In [7], a View-Upload Decoupling (VUD) scheme is proposed to decouple what a peer uploads from what it views, bringing stability to multi-channel P2P streaming systems and enabling cross-channel resource sharing. In [30], Wu et al. develop infinite-server queueing network models to methodically study the performance of multi-channel P2P live streaming systems. In [31], the bandwidth contentment ratio is used to compare three bandwidth allocation schemes, namely Naive Bandwidth allocation Approach (NBA), Passive Channel-aware bandwidth allocation Approach (PCA) and Active Channel-aware bandwidth allocation Approach (ACA), in multi-channel P2P streaming systems. In [32], Wu et al. proposed an online server capacity provisioning algorithm to adjust the server capacities available to each of the simultaneous channels, taking into account the number of peers, the

streaming quality, and the priorities of channels. In [46], Zhao et al. examined the streaming capacity in multi-channel P2P live streaming systems when each peer can only connect to a small number of neighbors.

In a multi-tree based system, the number of all distinct parents of a node is defined as in-degree, that of distinct children as out-degree. The streaming capacity of the entire system is defined as the maximum possible value of the summation of streaming rate of all multi-trees. While it is simple to compute this streaming capacity without degree bound [9], under certain degree bound it turns out to be a nontrivial job. As proposed in [20], other than streaming capacity, this model also seeks to find maximum streaming rate and transmission delay.

2.3 Mesh-Based P2P Live Streaming System

Mesh-based approach has been adopted by recent P2P video streaming systems. In a mesh-based system, topology is highly dynamic. Each peer maintains a list of neighbors and all peers form almost a complete graph. Transmission occurs only among a peer and its neighbors. If one of a peer's neighbors leaves the system, the peer can choose other candidates immediately to maintain its download rate. This attribute makes such kind of systems extremely robust against peer churn. Early papers mainly consider the push-based approach [8] [15], while the pull-based approach is applied in practice, mainly due to its simplicity. Nevertheless, because peers behaviors are highly dynamic and diverse, such model is very difficult to characterize, despite much work [9]–[11], [14], [16]–[19] published in this area.

Recently, a large number of mesh-based P2P systems have been widely deployed over the Internet [1][2][59] and studied in academic research [62][63]. These designs are based on unstructured overlays in which no parent-child relationships exist between nodes. As a result, the overlay is easier to maintain and is highly resistant to churn. Massouli´e et al. [61] have proposed an epidemic live streaming algorithm, proven to be rate-optimal for fully connected overlays and arbitrary node bandwidths. The algorithm, called DP/RU (Deprived Peer / Random Useful),

spreads chunks through the overlay in an epidemic, random like fashion. Although the optimality proof assumes a complete graph, simulations results shown that DP/RU also achieves near-optimal rates when using a small-degree unstructured overlay [59].

Capacity-oriented model [9][14][18][19]. This model considers P2P streaming from the perspective of network capacity. The idea is that total upload bandwidth of the entire system should be larger than or equal to total download rate of all peers. Assuming that a server with capacity *Us* want to support N peers with average capacity of *Up*, and there is a proper scheduling algorithm to fully utilize this total upload capacity, it is simple to derive performance bounds concerning maximum streaming rate and startup delay.

2.4 P2P Streaming Systems and Streaming Capacity

Streaming capacity in P2P live systems has been observed in the recent literature [5][13][38][39]. Most recent work were done to improve the efficiency and performance of P2P streaming systems. Lui et al [59] presented algorithms that find near-optimal streaming rates when nodes can only support a bounded number of children. Picconi et al [60] demonstrated that P2P live streaming systems can incorporate locality-awareness and thus be ISP-friendly. Other work [61] had also focused on utilizing network coding for improving download speeds and reducing the insufficiency of data.

In [5], the streaming capacity problem was formulated as an optimization problem, with the goal to maximize the streaming rate that can be supported by multi-tree based overlay. Sengupta et al. [5] provided a taxonomy of sixteen problem formulations on streaming capacity, depending on whether there is a single P2P session or there are multiple simultaneous sessions, whether the given topology is a full mesh graph or an arbitrary graph, whether the number of peers a node can have is bounded or not, and whether there are non-receiver relay nodes or not [38]. Liu et al. investigated the performance bounds for minimum server load, maximum streaming rate, and minimum tree depth in tree-based P2P live systems, correspondingly [13]. The streaming

capacity under node degree bound is inspected in [39].

In [6], P2P streaming systems were studied and classified into P2P live streaming systems and P2P VoD systems. In [4], the cross-channel resource sharing in correlated-channel P2P VoD systems were examined the server upload allocations was optimized among the channels to maximize the average streaming capacity. Additionally, bandwidth amplifiers to establish cross-channel links were introduced, hence enabling the peer upload sharing between channels. Using the correlated-channel P2P VoD systems with cross-channel resource sharing, it was proved that one can achieve a higher average streaming capacity compared to the independent-channel P2P VoD systems without cross-channel resource sharing.

A View-Upload Decoupling (VUD) scheme was proposed in [7] to decouple what a peer uploads from what it views, bringing stability to multichannel P2P streaming systems and enabling cross-channel resource sharing. In [7] Wu et al. presented a VUD framework for multi-channel P2P video systems. By fundamentally decoupling peer video uploading from viewing, VUD solves two primary performance problems: traditionally isolated-channel P2P streaming, and bad quality for small channels. In other papers [8][9][10], the performance and the efficiency of multi-channel P2P live or VoD streaming systems were investigated using a verity of techniques.

The maximum average streaming rate and average streaming rate are often derived to demonstrate how efficient the system is. The maximum streaming rate that can be received by every user in a channel is defined as the streaming capacity of the channel in a P2P live streaming system [5]. From the user's perception, this refers to its download rate. In [8][13][14][18], it was argued that that this streaming rate or throughput is not accurate in P2P streaming because there is no need to maximize such download rate. A download rate larger enough than the playback rate and this will be enough to guarantee the fluency of streaming. The transmission or diffusion means how long it takes for a chunk to be transmitted from the source

to a end peer. Users may see this as the difference of playback point between peers when they are watching the same live video. This is desired to be minimized [11][15] in live streaming applications in real time obligation. [10][14][18]. In order to maintain the smooth streaming without any delay, end systems must download enough chunks before they can start playback, but there is some startup delay or initial buffering delay. To overcome this issue, some systems set a fixed value for the number of chunks. Other work considers enough buffering as the buffer being filled up with chunks. The time to transmit these chunks is called startup delay. In [10][14][18] this startup and initial buffering delay was investigated and solutions were proposed to overcome this issue. As an alternative of maximize streaming rate, it was argued in [14][19] that it is worth scaling the system to support more peers, as when average streaming rate exceeds the inherent playback rate of a movie, all the peers are possibly be able to playback fluently. Thus extra bandwidth of the system can be used to provide support to new peers to make the system scalable. The total number of all peers within the system is system scale.

Sustainable streaming rate is the key quality of a P2P streaming system. It is defined as the rate that video can be played without skips or pauses. Different models may have different definition of this. Zhou et al [10] and Zhao et al [16] developed probabilistic model, to characterize the continuity. In [9][10][15][16][24], throughput was defined for VoD systems as the maximum slope of a line that does not exceed the y-coordinate at any time of the curve of consecutive arrived chunks. In [9] it was stated that the maximum streaming rate of a system cannot exceed download bandwidth or upload bandwidth. No matter what algorithm is applied, upload bandwidth or the download bandwidth of the entire system is not infinite. The average download rate or streaming rate of peers provides an upper bound for the average download rate.

A promising technique that may facilitate P2P video streaming is network coding. Network coding was initially proposed to improve the throughput by making the optimal use of bandwidth resources in a network for content distribution [7][26][28]. Its effects have been studied in [24] and [29], which demonstrate that random linear network coding is easy and feasible for both P2P live streaming and VoD systems.

2.5 Chapter Summary

In this chapter, we have discussed some related works on streaming capacity and peers remaining bandwidth usage and maximize the streaming rate. The results of their researches encouraged and motivated us to work on this sustainable bandwidth by using the peers unused available bandwidth and resources. The following chapter we proposed several schemes that allow the available bandwidth of rich peers to be shared with poor peers, a peer has smaller download bandwidth is called a poor peer. By doing this the streaming capacity of the channel is increased, the rich and poor peer terms we used for peers in terms of available bandwidth. The objective of our work is to reach a high streaming capacity for a channel in P2P live streaming systems in Tree-based and Meshed-based live streaming systems.

Chapter 3

Improving Streaming Capacity in Tree-based Multi-Channel P2P Live Streaming Systems via Cross-Channel Resource Sharing

Tree-based P2P streaming approaches explicitly construct multiple trees connecting the source to all peers, divide the stream into multiple sub-streams, and route one sub-stream per tree. The streaming capacity for a channel is defined as the maximum streaming rate that can be received by every user in the channel. In tree-based multi-channel P2P live streaming systems, the streaming capacity is limited by the internal peer with a low upload bandwidth. In this chapter, we propose a cross-channel resource sharing approach to improve the streaming capacity. We employ cross-channel helpers to establish the cross-channel overlay links, with which the unused upload bandwidths in a channel can be utilized to help the bandwidth-deficient peers in another channel, thus improving the streaming capacity. We demonstrate in the simulations that the proposed cross-channel resource sharing approach can significantly improve the streaming capacity in tree-based multi-channel P2P live streaming systems.

3.1 Overview

Though network-layer multicast is efficient in broadcasting a video to all users, it is not widely accepted largely due to the router overhead of managing multicast groups and the complexity of transport control for multicast sessions. Therefore the multicast function is moved from the network layer to the application layer. In a tree-based P2P live streaming systems, a single application-layer tree or multiple application-layer trees are constructed to deliver the video streams. P2P technology has recently become a capable approach to provide live and on-demand video streaming services over the Internet at low cost. There are two kinds of tree-based P2P live streaming systems, single-tree and multiple-tree. In a single-tree based P2P live streaming system, users participating in a live video streaming session can form a tree at the application layer. The root of the tree is the server. Each user joins the tree at a certain level. It receives the video from its parent peer at the level above and forward the received video to its child peers at the level below it. On the other hand, in multiple-tree based P2P live streaming systems, the video is encoded into multiple sub-streams, and each sub-stream is delivered over one tree. In this chapter, we have investigated and focused on multiple Tree-based multi-channel P2P live streaming systems.

In a live streaming session, a live video content is disseminated to all users in real-time. The video playbacks on all users are synchronized. To the contrary, VoD users enjoy the flexibility of watching whatever video clips whenever they want. The playbacks of the same video clip on different users are not synchronized.

In early days of the Internet, IP level multicast was projected as a proficient way to stream audio and video to a group of users. In an IP multicast session, the video source server is connected to all users participating in the session by a multicast tree formed by IP routers in the network. Unfortunately, largely due to the router overhead of managing multicast groups and the complexity of transport control for multicast sessions, it is difficult to widely deploy the IP level multicast over the Internet cloud. Instead, the multicast function has been implemented recently at application layer. Video servers and users form an application level overlay networks to

distribute video content.

Single-Tree Streaming

Similar to an IP multicast tree formed by routers at the network level, users participating in a video streaming session can form a tree at the application layer that is rooted at the video source server. Each user joins the tree at certain level. It receives the video from its parent peer at the level above and forward the received video to its children peers at the level below.

Multi-Tree Streaming

In multi-tree streaming, the server divides the stream into multiple sub-streams. Instead of one streaming tree, multiple sub-trees are constructed, one for each sub-stream. Each peer joins all subtrees to retrieve sub-streams. Within each sub-tree, the corresponding sub-stream flows down level by level from the source server to all the leaf nodes. A peer has different positions in different sub-trees. It might be positioned on an internal node in one subtree and on a leaf node in another subtree. A peer's uploading bandwidth will be utilized to upload a sub-stream whenever it is placed on an internal node in some sub-tree. To achieve high bandwidth utilization or maximum streaming capacity, the number of sub-trees in which a peer is placed on an internal node can be set to be proportional to its uploading bandwidth.

3.2 The Problem

P2P Streaming Rate

The fundamental question in P2P streaming is; what is the maximum streaming rate that can be sustained for all peers in a channel for all the peers? This question in P2P streaming capacity is often challenging because of the constraints imposed on the overlay topology. In this chapter, we demonstrate in the simulations that in tree-based multi-channel P2P live streaming systems,

cross-channel resource sharing approach can extensively improve the streaming capacity of the channel.

Streaming Capacity

P2P systems provide a scalable way to stream the content to multiple peers. The maximum rate achievable by all peers is the capacity of a P2P streaming session. The maximum streaming rate that can be received by every user in a channel is defined as the streaming capacity of the channel in a multi-channel P2P live streaming system [5]. The streaming capacity can serve as the indicator of video quality for a channel. A higher streaming capacity for a channel means that the users in the channel can receive a higher video quality. Therefore, the objective of this chapter is to achieve a high streaming capacity for a channel in P2P live streaming systems.

Maximum Streaming Capacity

The capacity is defined as the maximum streaming rate to be achievable by all the peers in a P2P streaming session. It depends on system configurations and constraints, such as; the available bandwidth and degree bound of each node, the network topology whether it is fully meshed or not, the topology approach whether it is tree-based or mesh-based, the constraints of the nodes whether they are heterogeneous or not, etc. Finding the maximum streaming rate in a tree-based P2P streaming system is a difficult task. However, it becomes more complicated in a mesh-based P2P system, where a node can connect to any subset of the nodes in the system. It is challenging to achieve a high streaming capacity in tree-based multi-channel P2P live streaming systems [5]. The streaming capacity for a channel is dependent on the bandwidth capacity of each peer in the channel and the overlay structure of the channel. A higher streaming capacity can be achieved by optimizing the resources among the peers within the same channel or cross-channel, which is, however, a challenging task. Peers may have heterogeneous bandwidths, some peers may have deficient upload bandwidths, which limit the streaming capacity, while some other peers may have abundant upload bandwidths which have not been fully utilized. Resource sharing in a multi-channel P2P live streaming system is expected to improve the streaming capacity.

However, resource sharing in multi-channel P2P live streaming systems is difficult to achieve due to the following reasons.

1) In a P2P live streaming system, each peer has different upload and download capacity, and each peer may join or leave the channel at any time. The heterogeneous characteristics and dynamic behaviors of the peers cause dynamic resource distribution among peers and among channels.

2) It is difficult to shift the unused resources in a channel to improve the streaming capacity for another channel, since there is originally no overlay connection between the two channels.

The related work on this topic has already been discussed in detail in chapter 2. A previous work investigated the cross-channel resource sharing in correlated-channel P2P VoD systems [5], in which the server upload allocations among channels are optimized to maximize the average streaming capacity. Wu et al. [7] presented a View-Upload Decoupling (VUD) framework for multi-channel P2P video systems. By fundamentally decoupling peer video uploading from viewing, VUD solves two primary performance problems: traditionally isolated-channel P2P streaming, and bad quality for small channels [7]. In other papers [8][9][10], the performance and the efficiency of multi-channel P2P live or VoD streaming systems were investigated using different techniques.

In this chapter, unlike the above discussed works, our approach is a cross-channel resource sharing in tree-based multi-channel P2P live streaming systems to improve the streaming capacity. The proposed approach employs cross-channel helpers to establish cross-channel overlay links, via which the unused resources in a channel can be utilized to help the bandwidth-deficient peers in another channel. The proposed scheme can significantly improve the streaming capacity compared to the case without cross-channel resource sharing.

3.3 System Overview

In this chapter, we worked on the streaming capacity in tree-based P2P live streaming systems. We will extend our study to mesh-based P2P live streaming systems in the 4th chapter. In a tree-based multi-channel P2P live streaming system, the peers in the same channel are organized into a tree for delivering media streaming, with the media source (e.g., the streaming server) as the root of the tree. Any P2P streaming system consists of two diverse but correlated components that are overlay construction and content delivery.

Overlay Construction

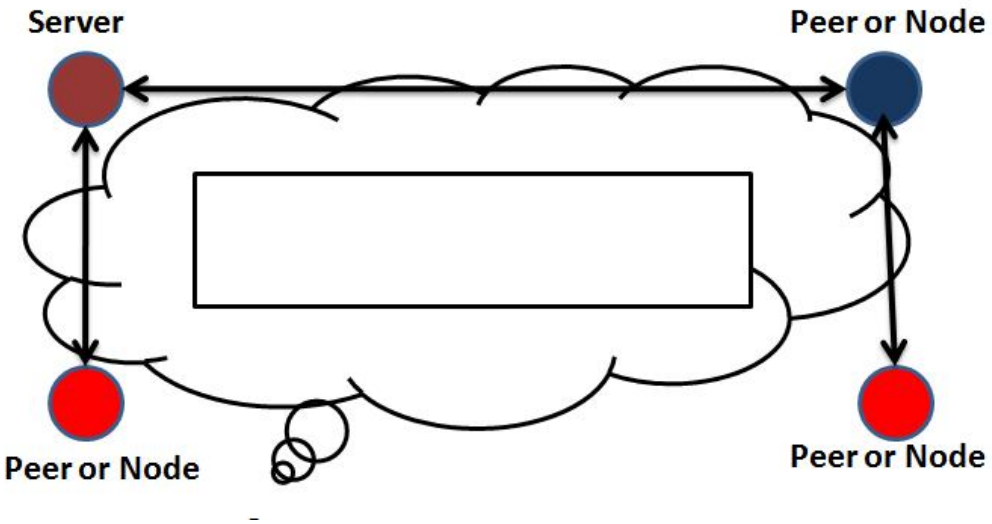


Fig. 3.1: Illustration of an Overlay on top of existing network

Tree-based overlays employ a tree distribution graph rooted at the source of content, each peer receives data from a parent peer, which may be the source or a peer node.

The peers watching the same channel form a tree-based overlay, respectively, with the common root, the streaming server. For example the outgoing degree of the tree is 3, which means that

each peer can have at most three child nodes. The video content is delivered from the root to each peer along the tree.

3.4 Streaming Capacity for Multi-Channel P2P Live Streaming Systems

The streaming capacity for a channel in the tree-based P2P live streaming system is dependent on the upload bandwidths of the peers. For example, suppose that peer a4 in Fig. 3.2 has an upload bandwidth of 900 Kbps and it supplies the stream to three child nodes. In this case, each child node of peer a4 can receive a streaming rate of 300 Kbps. In other words, the streaming capacity for a channel is limited by the internal peer (e.g., the peer who has child node) who has a low upload bandwidth. On the other hand, many other peers such as peers who have no child, do not use their upload bandwidth. Moreover, some internal nodes may have large remaining upload bandwidths which have not been used.

The streaming capacity for each channel is expected to be improved if the resources in each channel can be re-organized. To balance the utilization of the upload bandwidth among peers. Here, we propose a cross-channel resource sharing approach to improve the streaming capacity in the tree-based multi-channel P2P live streaming system. The essential idea of the proposed approach is to utilize the unused upload bandwidths of the peers in a channel (e.g., channel A) to help the bandwidth-deficient peers in another channel (e.g., channel B).

Cross-Channel Resource Allocation

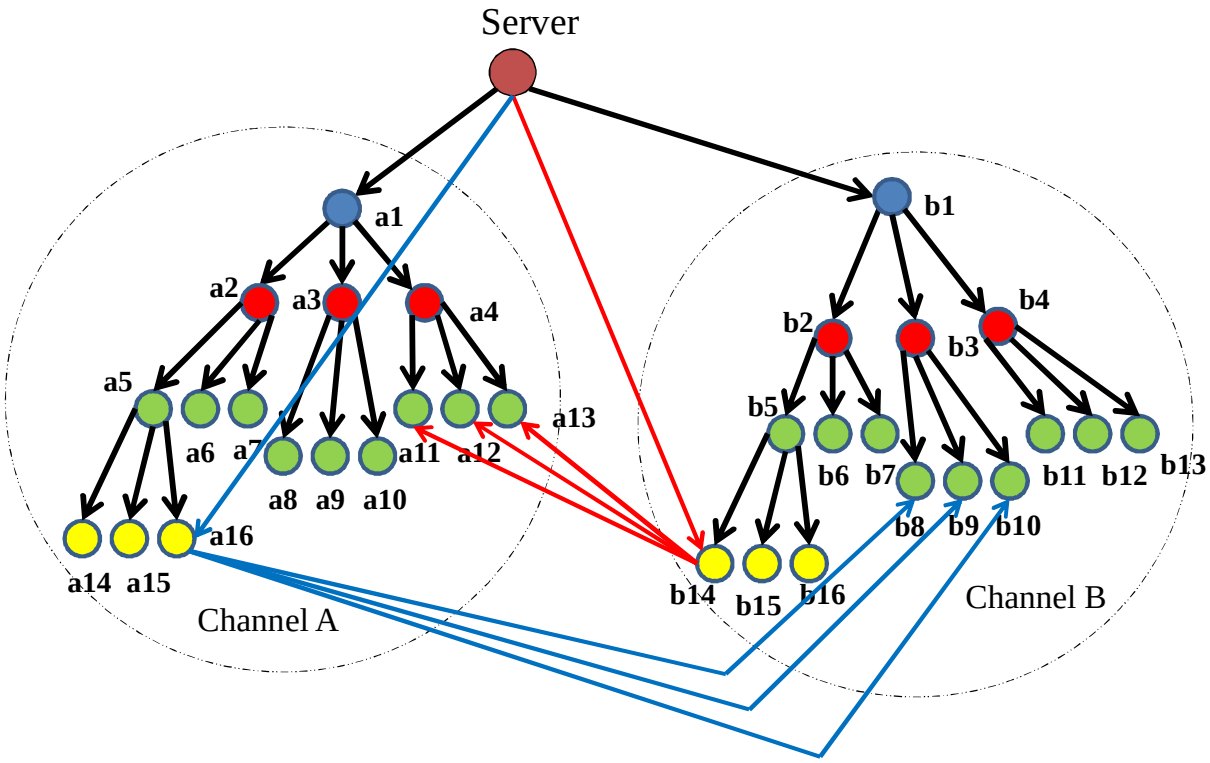


Fig. 3.2: Illustration of proposed cross-channel resource sharing approach

The proposed approach is illustrated in Fig. 3.2 Channels A and B are a pair of channels which help with each other. Suppose peer a4 is a bandwidth-deficient peer in channel A, peer b14 is bandwidth-abundant peer in channel B. We define a cross-channel helper as the peer who will contribute its remaining upload bandwidth to help the bandwidth-deficient peer in the partner channel. For example, peer b14 is a cross-channel helper, who downloads a segment of the stream from the server and then serves the segment to the child nodes of peer a4 in channel A. The advantage of cross-channel helper is to utilize the unused, bandwidth of a channel by the other channel. For example, peer b14 can download a segment of channel-A video at a rate of 50 Kbps, and serve the video to peers a11, a12, and a13 at a rate of 50 Kbps, respectively. In the same way, the bandwidth-abundant peer in channel A (e.g., peer a16) can help the bandwidth-deficient peer (e.g., peer b3) in channel B. The streaming capacity of channels A and B can both be improved by such cross-channel resource sharing.

3.5 Practical Protocol

Tree Constructions

Paper [64] described the procedure of creating the tree among the peers in a P2P live streaming systems in a centralized approach. Fig. 3.3 shows how a peer gets a stream from the other peer as child/parent relationship in a P2P live streaming system.

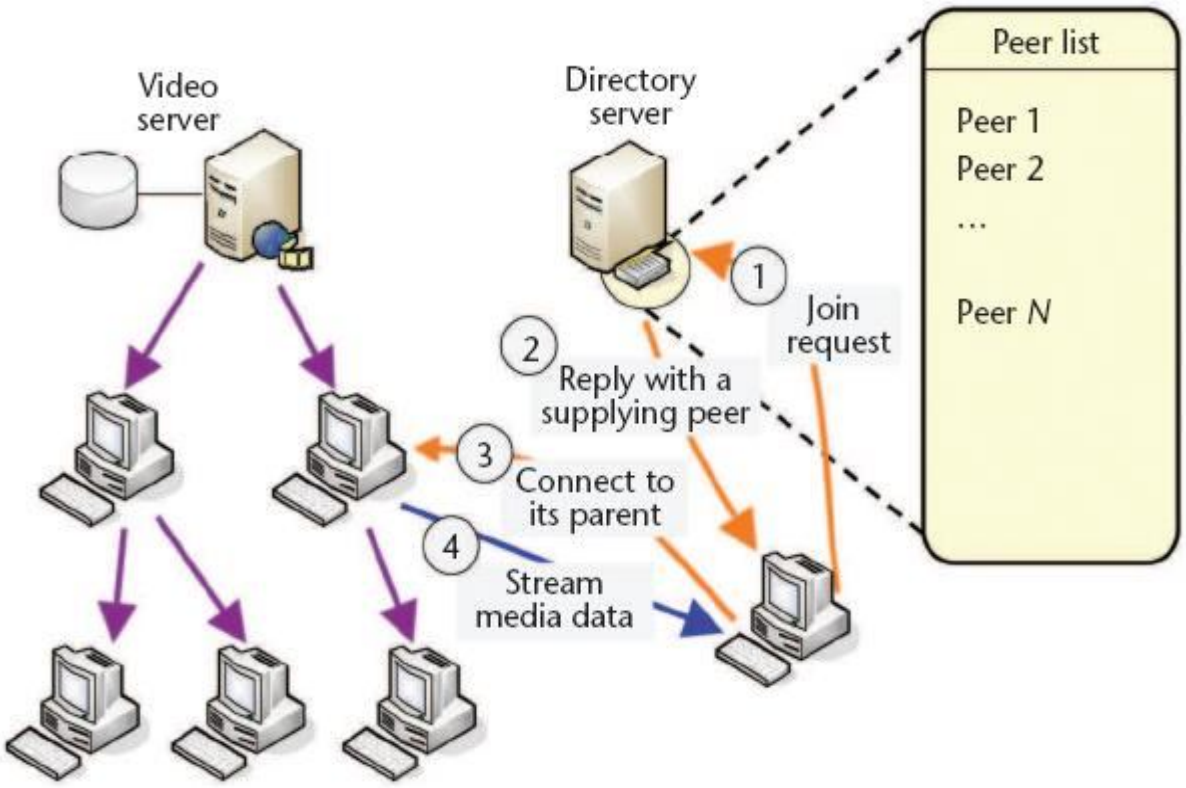


Fig. 3.3: Illustration of creating the tree in a channel of P2P live streaming system[64]

In [64], four steps are used to make the tree in a P2P live streaming system in a centralized approach. At the first step, peer *x* sends a join request to the directory server. At the second step, the directory server replies peer *x* with peer information and adds peer *x* into its peer list. At the third step, peer *x* connects itself to its parent peer with the help of the information that peer *x* gets from the directory server. At the fourth step, the parent peer starts sending the stream to peer *x*.

In a centralized approach [6], a central server controls the tree construction and its recovery. When a peer joins the stream, it first contacts the server. Based on the existing topology and the characteristics of the newly joined peer, such as its location and network access capacity, the server decides the position of the new peer in the tree and notify it which parent peer to connect to. A peer might leave the session at any time either gracefully or unexpectedly (e.g., machine crashes). After a peer leaves, all its children in the tree get inaccessible from the server and cannot receive the video. In such a case, the inaccessible peers can contact the central server, who then assigns each of the inaccessible peers to a new parent peer in the tree. In our proposed approach we used the same [6] principle to construct the tree in P2P live streaming systems.

Proposed Approach

The proposed cross-channel resource sharing approach is centralized which consists of five steps as follows.

1) Determine the channel pair: In each channel, server selects a partner channel to help with each other. All channels are first ordered based on the channel upload capacity, which is defined as the sum of upload bandwidths of the peers in the channel. The channels are then grouped in pairs based on a resource-balancing criterion, with which the channel with a larger channel upload capacity is paired with the one with a smaller channel upload capacity.

2) Determine cross-channel helpers in each channel: The server of a channel chosen the rich peer as the cross-channel helper peers which has an unused upload bandwidth larger than a pre-set threshold.

3) Determine bandwidth-deficient peers in each channel: The server of a channel chooses the poor peers or child peers. The average upload bandwidth per child is defined as the ratio between the upload bandwidth of the internal peer and the number of its child nodes. The internal peers with an average upload bandwidth per child smaller than a pre-set threshold are chosen as bandwidth-deficient peers.

4) Establish the cross-channel overlay links: The cross-channel overlay links are established by enabling the cross-channel helper to download a segment of the stream from the server or a peer

and forward the segment to the child nodes of the bandwidth-deficient peer in the partner channel.

5) Determine the streaming capacity for each channel: The streaming capacity for each channel can be found based on the upload bandwidths of the peers and the contributed rates from the cross-channel helpers.

Peer dynamics have an impact on the streaming capacity in tree-based multi-channel P2P live streaming systems. First, the peers may leave or join a channel dynamically. Second, the cross-channel helpers may leave the channel, which causes the disconnection of the cross-channel overlay links. To handle the dynamic conditions, the proposed cross-channel resource sharing approach needs to be performed in a discrete-time manner, considering the time-varying status of the peers.

3.6 Simulations

We perform simulations for a tree-based multi-channel P2P live streaming system with two channels (channel A and channel B). The numbers of peers in channels A and B are 100 and 80, respectively.

The peers have heterogeneous bandwidths. The download bandwidths of the peers are uniformly distributed between 2.5 Mbps and 4.0 Mbps, and the upload bandwidths of the peers are uniformly distributed between 1.5 Mbps and 3.0 Mbps. The tree overlay for each channel is constructed based on the startup time of the peer. The newly coming peer chooses an existing peer as parent and connects itself with the parent node. The outgoing degree is defined as the maximum number of the child nodes that an internal peer can have. The outgoing degree is set to 5 in the default setting.

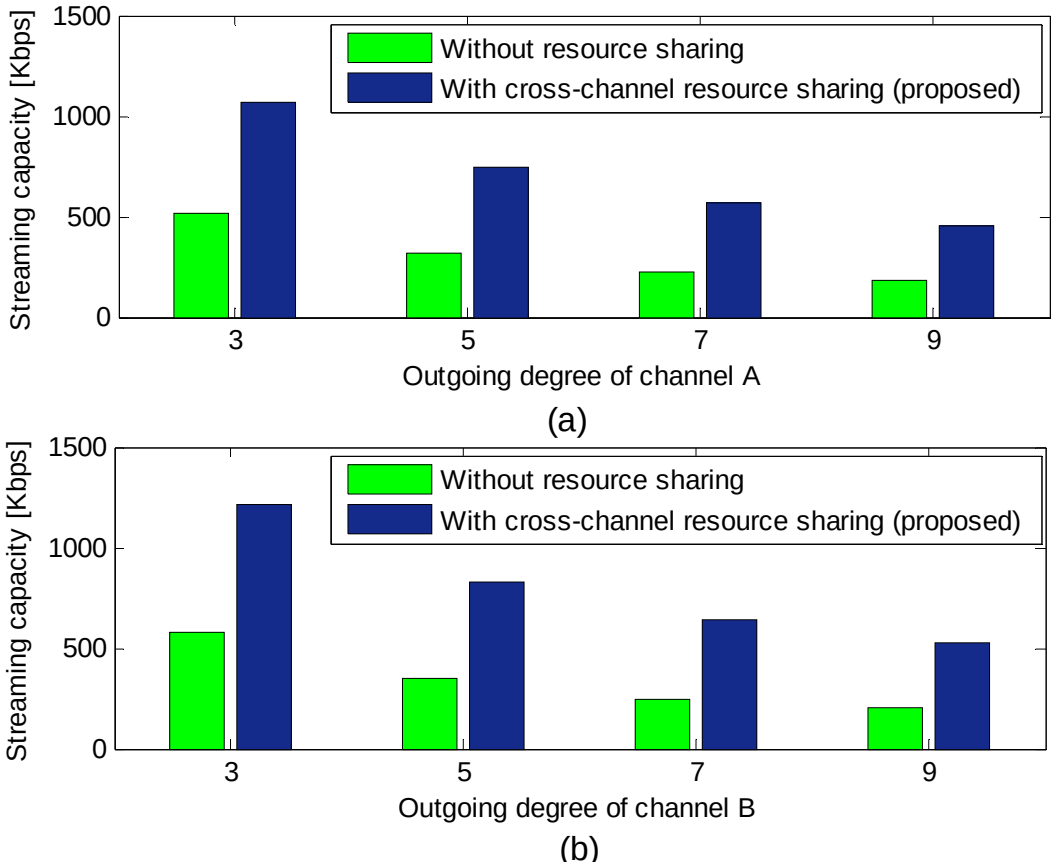


Fig. 3.4: Comparison of streaming capacity with different outgoing degree: (a) channel A, and (b) channel B

Fig. 3.4 compares the streaming capacity between the case without cross-channel resource sharing and the case with the proposed cross-channel resource sharing. The proposed cross-channel resource sharing better utilizes the upload bandwidths, thus significantly improving the streaming capacity compared to the case without resource sharing. The average improvements of streaming capacity are 135% for channel A and 143% for channel B respectively, as shown in the following table as well.

	Channel A		Channel B	
outgoing degree	without helper	with helper	without helper	with helper
3	517.433	1067.35	579.524	1213.79
5	320.167	739.655	347.714	828.7
7	228.691	563.761	248.367	641.547
9	177.87	454.218	199.367	527.242

Table 3.1: streaming capacity vs. outgoing degree (vary out_degree)

The total helping rate is distinct as the addition of the outgoing rates of all cross-channel helpers in a channel, the streaming capacity is improved when the total helping rate is increased.

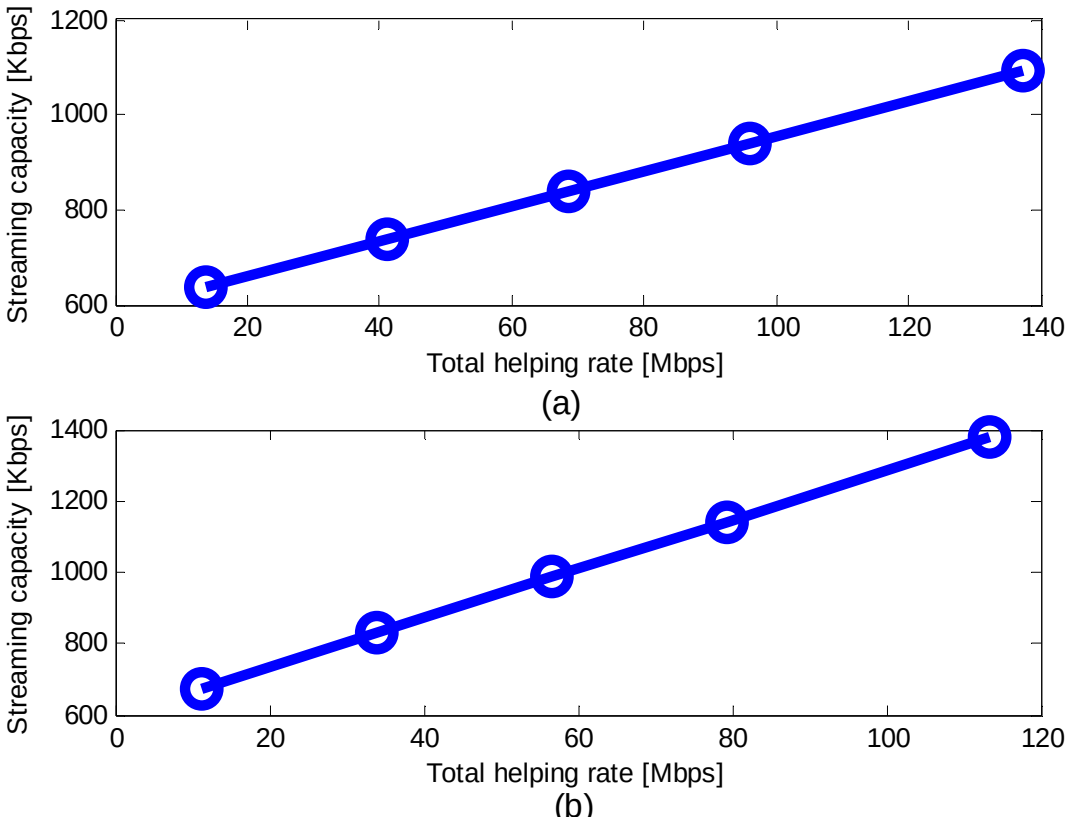


Fig. 3.5: Comparison of streaming capacity with different total helping rate:

(a) channel A, and (b) channel B

Fig. 3.5 compares the streaming capacity with different total helping rate, which is defined as the sum of the outgoing rates of all cross-channel helpers in a channel. We can see in Fig. 3.5 and the following table 3.2 that the streaming capacity is almost linearly increased when the total helping rate is increased from 13.7 Mbps to 137.2 Mbps for channel A and from 11.3 Mbps to 113.4 Mbps for channel B.

Channel A			Channel B		
total help rate [Kbps]	without helper	with helper	total help rate [Kbps]	without helper	with helper
13723	320.167	638.817	11344.3	347.714	671.866
41169.1	320.167	739.655	34032.8	347.714	828.7
68615.1	320.167	840.493	56721.3	347.714	985.535
96061.2	320.167	941.33	79409.8	347.714	1142.37
137230	320.167	1092.59	113443	347.714	1377.62

Table 3.2: streaming capacity vs total help rate

The standard deviation is defined as the heterogeneity of peer upload bandwidths calculated by peer upload bandwidths.

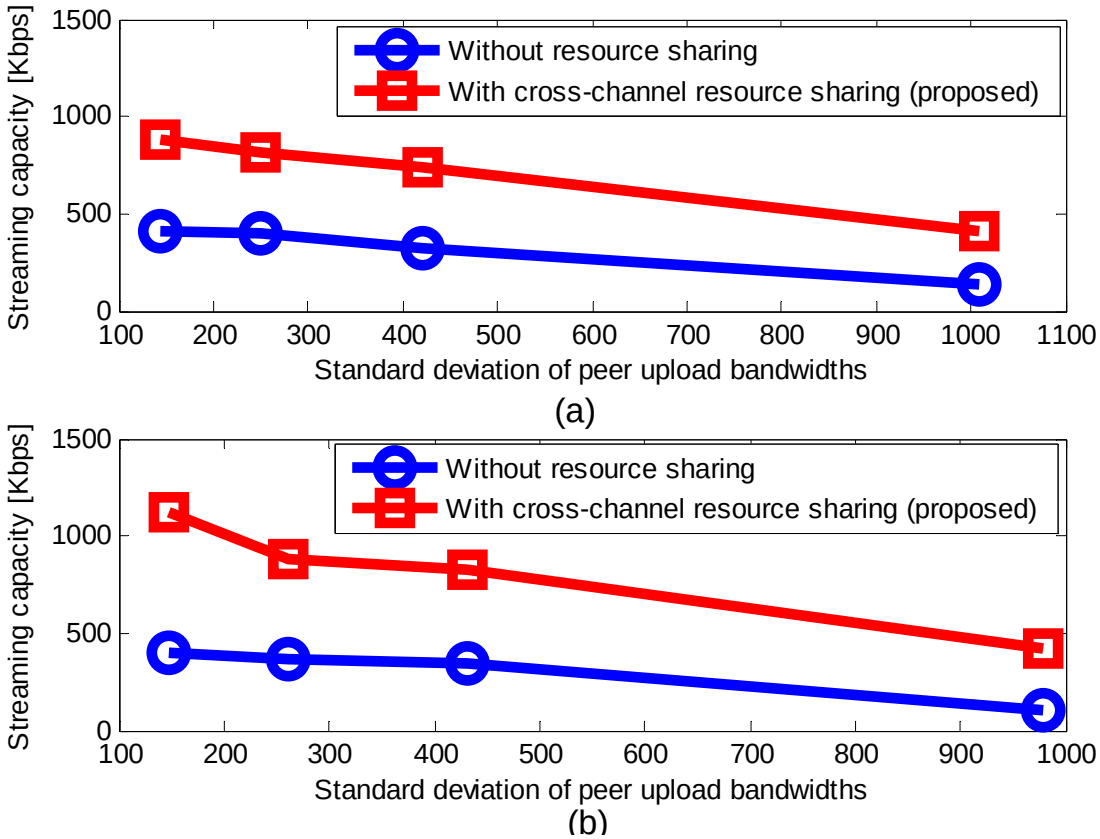


Fig. 3.6: Comparison of streaming capacity with different standard deviation of peer upload bandwidths: (a) channel A, and (b) channel B

Fig. 3.6 evaluates the impact of the heterogeneity of peer upload bandwidths to the streaming capacity. The heterogeneity of peer upload bandwidths can be measured by the standard deviation of peer upload bandwidths. A higher standard deviation indicates a higher heterogeneity of peer upload bandwidths. From Fig. 3.6, we can see that a higher heterogeneity of peer upload bandwidths leads to a lower streaming capacity. As shown in Fig. 3.6 and table 3.3, the proposed cross-channel resource sharing approach obtains a significantly improved streaming capacity, with an average improvement of 141% for channel A and 188% for channel B.

Channel A			Channel B		
standard deviation	without helper	with helper	standard deviation	without helper	with helper
1009.9	133.288	416.027	979.301	108.177	424.643
419.885	320.167	739.655	431.257	347.714	828.7
249.719	399.26	812.804	260.268	365.929	885.601
144.836	407.818	883.848	148.432	402.867	1125.41

Table 3.3: streaming capacity vs standard deviation of peer upload bandwidth

3.7 Chapter Summary

In this chapter, we proposed a cross-channel resource sharing approach to get better streaming capacity in tree-based multi-channel P2P live streaming systems. In tree-based multi-channel P2P live streaming systems, the streaming capacity is restricted by the internal peer with a low upload bandwidth, while the upload bandwidths of some other peers may have not been fully utilized. Our approach utilizes the unused upload bandwidths to improve the streaming capacity. We use cross-channel helpers to establish the cross-channel overlay links. The cross-channel helpers enable the unused upload bandwidth in a channel to be utilized in the partner or other channel, thus improving the streaming capacity of the partner or other channel. We demonstrated in the simulations that the proposed cross-channel resource sharing approach can significantly improve the streaming capacity compared to the case without cross-channel resource sharing.

Chapter 4

Improving Streaming Capacity in Mesh-based Single-Channel P2P Live Streaming Systems

In mesh-based (also called data-driven) P2P streaming approaches, peers dynamically exchange video packets with several of their neighbor peers. No tree is constructed. We study the streaming capacity problem in tree-based P2P live systems in chapter 3, and the same, we will discuss in mesh-based single-channel P2P live systems in this chapter, we first review the problem formulation of the streaming capacity problem in [5], and then we proposed optimal resource sharing approach which increased the streaming capacity as compare to equal utilization scheme.

4.1 Overview

Video streaming has become a popular service in the Internet. A large number of emerging applications, such as Internet TV, requires support for simultaneous video delivery to a large number of receivers. Meshed-based overlays implement a mesh distribution graph, where each peer contacts a subset of peers to obtain a number of video chunks. Every peer needs to know

which video chunks are owned by its peers and explicitly pulls those video chunks as per it needs. This type of method involves overhead, due in part to the exchange of buffer maps between peers that advertises the set of video chunks they hold and in part to the delivery process that each peer sends a request in order to receive the video chunks. Each peer relies on multiple peers to retrieve content, mesh-based systems offer good flexibility to peer failure. On the negative side they require large buffers to support the chunk pull, hence they needed large buffers to increase the chances of finding a video chunk, every peer keeps an active path for data, and a set of backup paths, in case the active path fails to deliver within certain time constraints. In this chapter we have investigated and focused on single-channel meshed-based P2P live streaming systems.

Due to bandwidth constraint, most of the current P2P live systems provide the video at a low bit rate. For example, the source video rate in PPLive system is usually between 381 to 450 Kbps [33]. Even if users would like to watch the video at a higher quality. The P2P network may not be able to deliver the video at a high rate. Therefore, determination of the upper bound of the streaming rate in a P2P system becomes an attractive topic. Streaming capacity is defined as the maximum supported streaming rate that can be received by every receiver [5]. This chapter focuses on the following problems:

1) What is the streaming capacity in the P2P live streaming systems? and

2) How to achieve such streaming capacity in a distributed manner?

4.2 Streaming Capacity in Data-Driven P2P Live Streaming systems

In a data-driven P2P live system, each peer obtains a set of neighbors from the server. The peer periodically exchanges data availability information with the neighbors, and then retrieves unavailable data from one or more partners, or supplies available data to neighbors. Fig. 4.1 illustrates a data-driven P2P live system, in which almost each peer exchanges the video content with its neighbors.

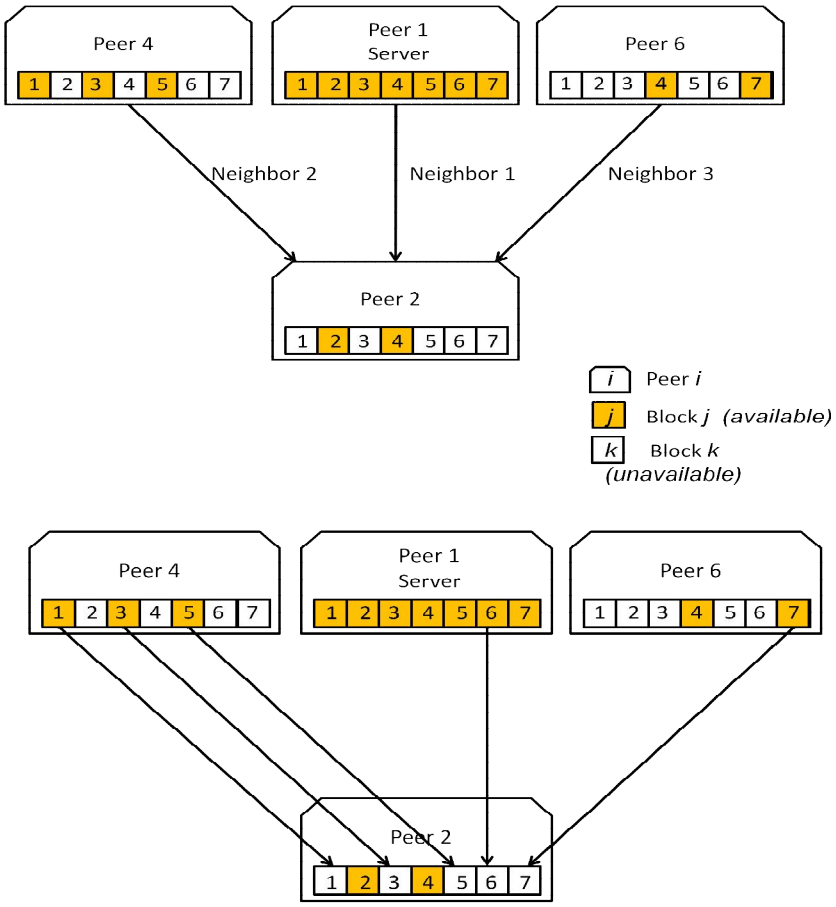


Fig.4.1: Illustration of block request at a peer: (a) the sliding window at each peer, and (b) the block request by peer 2

Following we have discussed the neighborhood formation in a data-driven P2P live streaming system.

4.3 Practical Protocol

In mesh-based P2P streaming systems, each peer downloads the video data from its neighbors. Let's first look at how a peer finds its neighbors. The process for finding the neighbors consists of 3 steps as shown in Fig. 4.2 At the beginning, the peer (e.g., peer 1 in Fig. 4.2) sends out a query message to the channel server to obtain an updated channel list, from which peer 1 chooses

a channel to watch (step 1). Next, peer 1 reports its own information, such as IP address and port number, to the tracker, who then provides peer 1 a peer list that contains a random subset of active peers in the channel (step 2). The tracker in a P2P streaming system is responsible for keeping track of the active peers for each channel. After receiving a peer list, peer 1 will send connection requests to some remote peers on the peer list (step 3). If a connection request is accepted by a remote peer, peer 1 will add the remote peer into its neighbor list. After obtaining sufficient neighbors in the neighbor list, the neighborhood peer 1 is formed, and then peer 1 starts to exchange the video content with its neighbors.

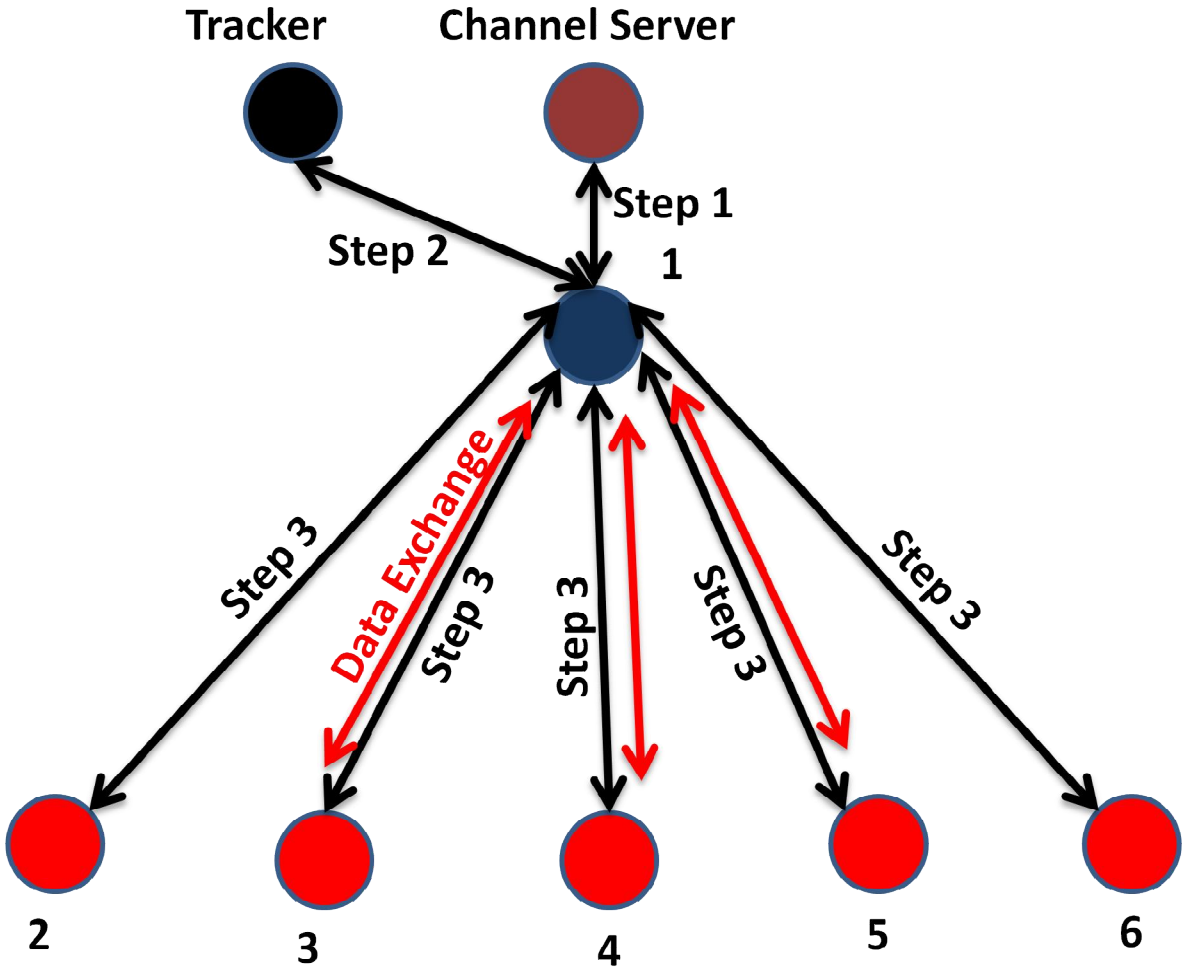


Fig. 4.2: Neighborhood formation in mesh-based P2P live streaming systems

Due to peer dynamics, the neighbor list of a peer may be stale or out of date. Therefore, a peer needs to periodically update its neighbor list. A peer can retrieve a new peer list from the tracker

periodically, and then find the neighbors from the peer list. It can also exchange the peer list with its neighbors, and then find the new neighbors from the updated peer list.

If a peer leaves the session gracefully, it will notify the tracker and its neighbors of its departure such that its information can be removed from their peer lists immediately. If a peer leaves unexpectedly such as computer crashes, the failure of the peer can be detected by regular heartbeat messages. A peer will remove the information of a neighbor from its neighbor list if it has not received a heartbeat message from the neighbor within a timeout period.

If the peer has more neighbors in its neighbor list, it can download the missing blocks from more sources simultaneously, thus increasing the throughput. However, connections with more neighbors introduce a higher communication overhead. In order to achieve a higher streaming performance, a peer can periodically update its neighbor list by rejecting the old neighbors with a low performance (e.g., a low upload bandwidth) and admitting the new neighbors with a higher performance (e.g., a high upload bandwidth). There is a tradeoff on the frequency of neighbor update. If a peer updates its neighbor list frequently, it can achieve a higher video quality at the price of a higher communication overhead.

4.4 Problem Formulation

The overlay of the mesh-based P2P live system can be modeled as a directed graph G = (N, L), where N is the set of nodes and L is the set of directed overlay links. Peer 1 is defined as the server. The neighbor set of peer i is denoted by B_i. The server, peer 1, is a neighbor peer of each receiving peer, such that each receiving peer can request the blocks that cannot be obtained from the other neighbors to guarantee the continuous playback. The streaming rate that can be received by every peer is denoted by r.

The relationship between a node and its outgoing links is represented with a matrix A^+, whose elements are given by

$$a_{il}^{+} = \begin{cases} 1, \text{if link } l \text{ is an outgoing link from node } i, \\ 0, \text{otherwise.} \end{cases} \quad (1)$$

The relationship between a node and its incoming links is represented with a matrix A^{-}, whose elements are given by

$$a_{il}^{-} = \begin{cases} 1, \text{if link } l \text{ is an ingoing link from node } i, \\ 0, \text{otherwise.} \end{cases} \quad (2)$$

In order to distinguish the server from the other peers, we define a *node-filtering element* f_i as follows.

$$f_i = \begin{cases} 0, \text{if } i = 1, \\ 1, \text{otherwise.} \end{cases} \quad (3)$$

A video with a length of L is evenly divided into a set of blocks, denoted by M. Each block has the same time duration, denoted by d. Each block is labeled with a playback time stamp T_j for $j \in$ M. Users in P2P live applications watch almost the same position of the video. We ignore the variation of the playback times of the users, and assume that each user has the same playback time t_t^p at time t where $0 \leq t \leq L$. Each peer maintains a length of the buffer, called the *sliding window*, which contains the emergent blocks after and close to the playback time. The start time of the sliding window at time t is the playback point of the video, denoted by t_t^p. The length of the sliding window is denoted by L_{sw}. Then the end time of the sliding window is given by $t_t^p + L_{sw}$. The sliding window moves forward at the same speed of the playback progress. The blocks following within the current sliding window are denoted by a set S_t at time t. At peer i, a bit vector H_i are given by

$$h_{ij} = \begin{cases} 1, \text{if link } j \text{ is } S_t \text{ is available at peer } i, \\ 0, \text{otherwise.} \end{cases} \quad (4)$$

The block request is performed in a discrete-time manner with an increment of τ with $0 < \tau < L_{sw}$. At time *t* peer *i* requests block *j* where $j \in S_t$ and $h_{jk} = 1$ to the peers who are requesting the blocks from it. The block request at a peer is illustrated in Fig. 4.1(a) and 2(b).

Each unavailable block is requested with a prioritized way. For a requesting peer (peer *i*), block $j(j \in S_t)$ is assigned with a priority weight P_{ij}, which is determined by the scheduling policy. For example, the scheduling algorithm that places a higher priority to the dissemination of the blocks in the P2P network will request the rarest block in the neighborhood first, while the scheduling algorithm that pays more attentions to the playback continuity of the requesting peer will request the block closest to the playback position first. For a supplying peer (peer *j*), it supplies the available blocks to peer *i* for $j \in B_i$. Block $j(j \in S_t)$ has a priority weight p_{ij} when it is supplied to peer *i.*

The streaming capacity problem is a time-variant problem due to peer dynamics. At the current time, peer *i* performs the following steps.

1) Peer *i* exchanges the information of block availability in the sliding window with the peers in the neighbor set B_i of peer *i*;

2) Peer *i* requests the block b_i, which has the highest priority weight p_i^m in the current sliding window, from the neighbor peers;

3) Peer *i* establishes an incoming link from peer $j(j \in B_i)$ to peer *i* if peer *j* owns the block b_i in its current sliding window;

4) If all neighbor peers of peer *i* do not own the block b_i in their current sliding window, peer *i* establishes an incoming link from the server and requests the block b_i from it.

Therefore, the streaming capacity at the current moment in the data-driven P2P live system can be described as to maximize the streaming rate r by optimizing the streaming rate r and the link rate $x_l(\forall l \in \mathrm{L})$, subject to the equality constraint that each receiving peer has to receive the same streaming rate, the upload bandwidth constraint and the download bandwidth constraint.

Mathematically, the problem can be formulated as follows.

$$
\begin{aligned}
&\text{minimize}_{(\mathrm{x},r)}\ r \\
&\text{subject to } \Sigma_{l\in\mathrm{L}}\, a_{il}^{-}\, x_l = f_i r, \qquad \forall i \in \mathrm{N}, \\
&\qquad \Sigma_{l\in\mathrm{L}}\, a_{il}^{+}\, x_l \le O_i, \qquad \forall i \in \mathrm{N}, \\
&\qquad 0 \le r \le \min_{i\in\mathrm{N}} \{I_i\}, \\
&\qquad 0 \le x_l \le x_l^m, \qquad \forall l \in \mathrm{L},
\end{aligned}
\tag{5}
$$

In the optimization problem (5), the objective function is the sum of the priority weights of the scheduled blocks, the first constrain $\Sigma_{l\in\mathrm{L}}\, a_{il}^{-}\, x_l = f_i r, \forall i \in \mathrm{N}$ represents that each receiving peer has to receive the same streaming rate r, the second constraint $\Sigma_{l\in\mathrm{L}}\, a_{il}^{+}\, x_l \le O_i, \forall i \in \mathrm{N}$ represents that the outgoing rate from each peer needs to be no larger than the upload capacity of the peer, and the third constraint $0 \le r \le \min_{i\in\mathrm{N}} \{I_i\}$ represents that the received streaming rate at each peer needs to be nonnegative and no larger than the download capacity of the peer.

The optimization problem in (5) is a LP. It can be solved in a centralized way using the interior point method [37]. The centralized solution for the optimization problem (5) provides the maximal streaming capacity supported by the P2P live system.

4.5 Simulations

In the numerical simulations, there are two classes of peers: cable/DSL peers and Ethernet peers. Cable/DSL peers take 85% of the total population with download capacity uniformly distributed

between 0.6 Mbps and 1.0 Mbps and upload capacity uniformly distributed between 0.2 Mbps and 0.4 Mbps. Ethernet peers take the remaining 15% of the total population with both upload and download capacities uniformly distributed between 1Mbps and 2 Mbps. The length of the video is 120 minutes, which is evenly divided into 120 blocks.

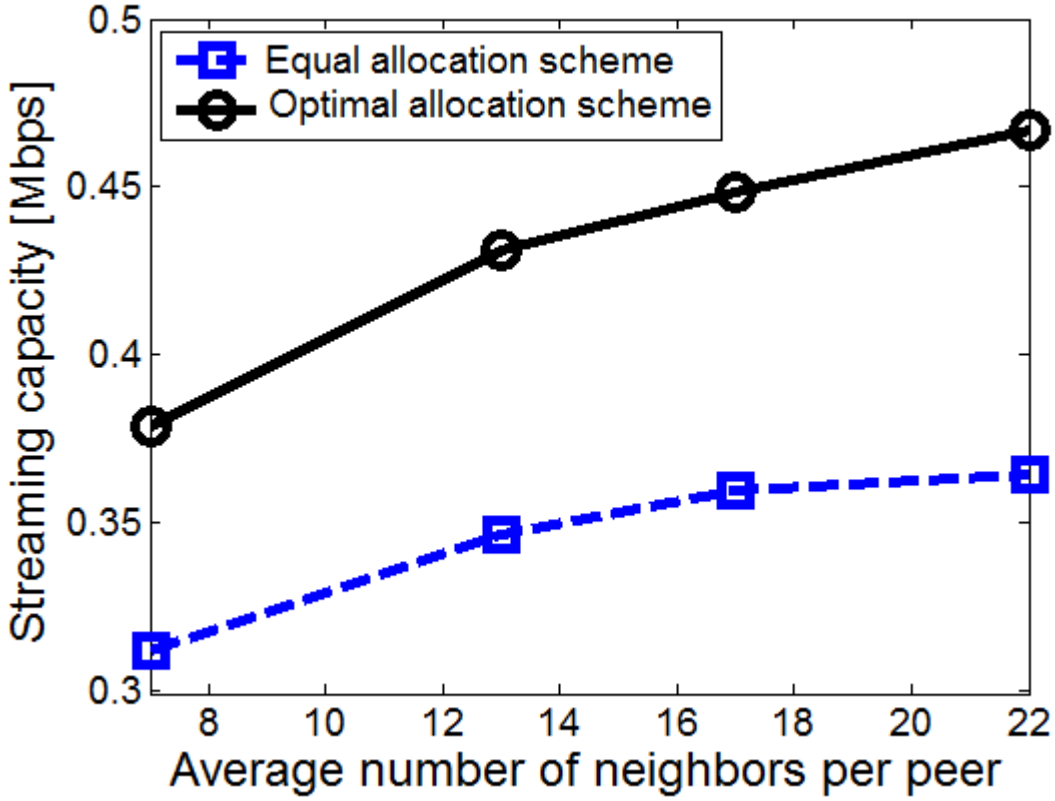


Fig: 4.3 Comparison of Streaming Capacity in two schemes

In Fig. 4.3, we compare the streaming capacity between the two schemes: 1) optimal allocation scheme (the proposed scheme), in which the link rates are optimized by solving the streaming capacity maximization problem; and 2) the equal allocation scheme, in which the link rates from each peer are equally allocated. As shown in Fig. 4.3, when the number of neighbors per peer is increased, each peer can download the missing block from more neighbors, thus increasing the streaming capacity. By optimally utilizing the peer upload bandwidths, the proposed optimal

allocation scheme improves the streaming capacity by 24.7% in average, compared to the equal allocation scheme.

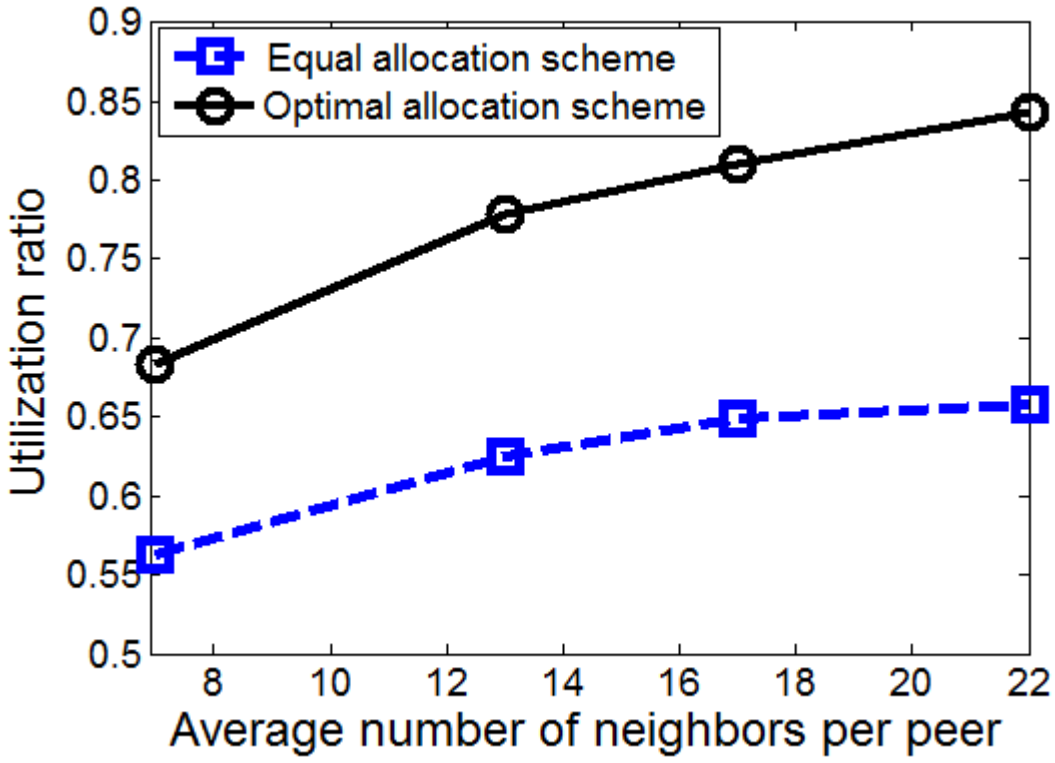


Fig: 4.4 Streaming Capacity v/s Average Peer Upload Bandwidth

In Fig. 4.4, we compare the utilization ratio of peer upload bandwidth, which is defined as: utilization ratio = streaming capacity/(total system upload bandwidth/number of peers). As shown in Fig. 4.4, the utilization ratio is increased as the number of neighbors is increased. The proposed optimal allocation scheme achieves a much higher utilization ratio than the equal allocation scheme.

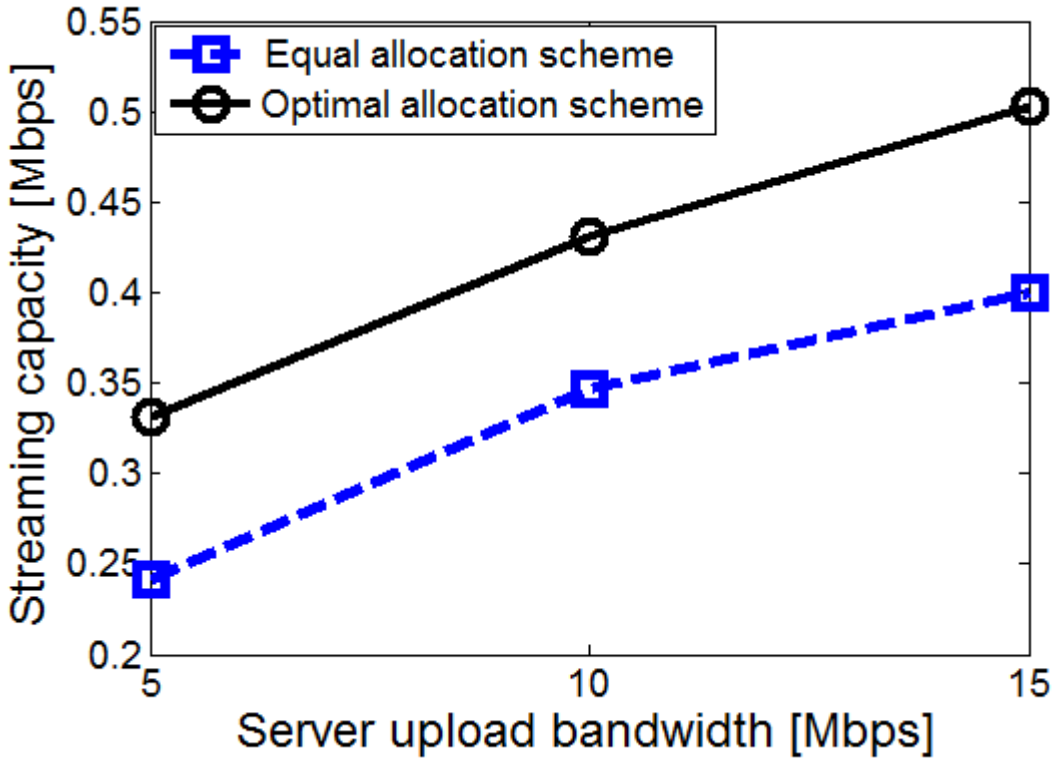


Fig: 4.5 Streaming Capacity v/s Server upload Bandwidth

In Fig. 4.5, we compare the streaming capacity with different server upload bandwidth. When the server upload bandwidth is increased from 5 Mbps to 15 Mbps, then the streaming capacity in the proposed optimal allocation scheme is increased from 0.331 Mbps to 0.503 Mbps. The proposed optimal allocation scheme improves the streaming capacity by 29.1% in average, compared to the equal allocation scheme.

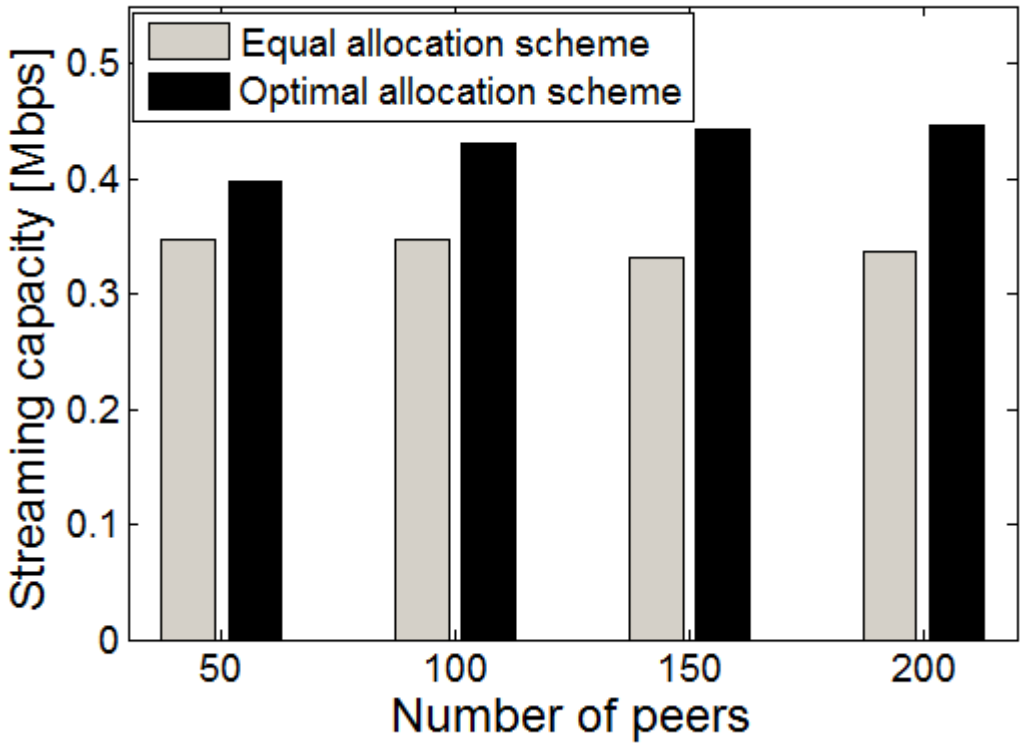


Fig: 4.6 Streaming Capacity v/s Different Number of Peers

In Fig. 4.6, we compare the streaming capacity with different number of the peers. We vary the number of peers from 50 to 200. The server upload bandwidth is set to 0.1* (number of peers) Mbps. The proposed optimal allocation scheme improves the streaming capacity by 26.5% in average, compared to the equal allocation scheme.

4.6 Chapter Summery

In this chapter, we have proposed a meshed-based (also called data-driven) resource sharing approach which we called it optimal allocation scheme to improve the streaming capacity, we formulate the problem in mesh-based P2P live systems. In mesh-based P2P streaming systems,

peers dynamically exchange video packets with several of their neighbor peers without constructing the explicit tree. In our simulation results, we demonstrated that proposed optimal resource sharing approach increased the streaming capacity as compare to equal utilization scheme.

Chapter 5

Conclusion

5.1 Summary

The proposed work in this thesis majorly lie in two different pitches of P2P live streaming systems, one is tree-based and other one is mesh-based. The maximum streaming rate that can be received by every user in a channel is defined as the streaming capacity of the channel in a P2P live streaming system [5]. A higher streaming capacity for a channel means that the users in each channel can receive a higher stream quality. Therefore, the objective of this paper is to achieve a high streaming capacity in Tree-based multi-channel and Meshed-based single-channel P2P live streaming systems.

In chapter 3, we proposed a cross-channel resource sharing approach to improve the streaming capacity in tree-based multi-channel P2P live streaming systems. In tree-based multi-channel P2P live streaming systems, the streaming capacity is restricted by the internal peer with a low upload bandwidth, although the upload bandwidths of some other peers may have not been fully

consumed. Our approach consumed the unemployed upload bandwidths to improve the streaming capacity. We employ cross-channel helpers to set up the cross-channel overlay links. The cross-channel helpers enable the unused upload bandwidth in a channel to be utilized in the partner channel, consequently improving the streaming capacity of the partner channel. We demonstrated in the simulations that the proposed cross-channel resource sharing approach can drastically improve the streaming capacity contrast to the case without cross-channel resource sharing.

In chapter 4, we proposed a Meshed-based (also called data-driven) resource sharing approach to improve the streaming capacity, we formulate the problem in mesh-based P2P live systems, and then solve the problem using optimal resource sharing approach. In mesh-based P2P streaming approaches, peers dynamically exchange video packets with several of their neighbor peers. No explicit tree is constructed, but the path of each packet still forms multi-tree, originating from the source and reaching each peer exactly once. Meshed-based overlays implement a mesh distribution graph, where each peer contacts a subset of peers to obtain a number of video chunks. Every peer needs to know which video chunks are owned by its peers and explicitly pulls those video chunks as per it needs.

5.2 Future Directions

In chapter 3, we have demonstrated the multi-channel tree-based P2P live streaming system resource sharing, the other researchers can extend this work by maintaining the maximum streaming rate and capacity if any of the peer in the single-channel tree-based P2P leaves the tree.

In chapter 4, we have demonstrated the single-channel meshed-based P2P live streaming work by using optimal allocation scheme and that's how we proved that streaming capacity is increased, this work can be extended to multi-channel meshed-based P2P live streaming systems.

References

[1] PPLive, http://www.pplive.com

[2] PPStream, http://www.pps.tv

[3] UUSee, http://www.uusee.com

[4] Y. He and L. Guan, “Streaming capacity in multi-channel P2P VoD systems”, in Proc. of IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1819-1822, May 2010.

[5] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, “Streaming Capacity in Peer-to-Peer Networks with Topology Constraints,” Microsoft Research Technical Report, 2008.

[6] Y. He and L. Guan, “Peer-to-peer streaming systems”, in Intelligent Multimedia Communication: Techniques and Applications, Springer-Verlag Berlin Heidelberg, pp. 195–215, May 2011.

[7] D. Wu, C. Liang, Y. Liu and K. W. Ross, “View-upload decoupling: a redesign of multi-channel P2P video systems,” in Proc. of IEEE INFOCOM, pp. 2726–2730, Apr. 2009.

[8] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez. Randomized decentralized broadcasting algorithms. trees, 1:5.

[9] R. Kumar, Y. Liu, and K.W. Ross. Stochastic fluid theory for P2P streaming systems. In Proc. of IEEE Infocom. Citeseer, 2007.

[10] Y. Zhou, D.M. Chiu, and J.C.S. Lui. A simple model for analyzing p2p streaming protocols. In Proc. of IEEE International Conference on Network Protocols (ICNP). Citeseer, 2007.

[11] Y. Liu. On the minimum delay peer-to-peer video streaming: how realtime can it be? In Proceedings of the 15th international conference on Multimedia, page 136. ACM, 2007.

[12] Y. Liu, Y. Guo, and C. Liang. A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications, 1(1):18–28, 2008.

[13] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance bounds for peer-assisted live streaming. In Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pages 313–324. ACM, 2008.

[14] C. Feng and B. Li. On large-scale peer-to-peer streaming systems with network coding. ACM New York, NY, USA, 2008.

[15] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: optimal performance trade-offs. In Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pages 325–336. ACM, 2008.

[16] B.Q. Zhao, J.C.S. Lui, and D.M. Chiu. Exploring the Optimal Chunk Selection Policy for Data-Driven P2P Streaming Systems. In The 9th International Conference on Peer-to-Peer Computing, 2009.

[17] D. Wu, Y. Liu, and K.W. Ross. Queueing Network Models for Multi-Channel P2P Live Streaming Systems. In Proceedings of IEEE Infocom, 2009.

[18] C. Feng, B. Li, and B. Li. Understanding the Performance Gap between Pull-based Mesh Streaming Protocols and Fundamental Limits. In Proc. of IEEE INFOCOM, 2009.

[19] F. Liu, B. Li, L. Zhong, B. Li, and D. Niu. How P2P Streaming Systems Scale Over Time Under a Flash Crowd? 2009.

[20] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. Chou. P2P Streaming Capacity under Node Degree Bound. In Proc. of IEEE INFOCOM, 2010.

[21] J. Jannotti, D.K. Gifford, K.L. Johnson, M.F. Kaashoek, and J.W. O'Toole Jr. Overcast: reliable multicasting with on overlay network. In Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4, pages 14–14. USENIX Association Berkeley, CA, USA, 2000.

[22] Y. Chu, SG Rao, S. Seshan, and H. Zhang. A case for end system multicast, volume 20. 2002.

[23] B. Li, Y. Keung, S. Xie, F. Liu, Y. Sun, and H. Yin. An Empirical Study of Flash Crowd Dynamics in a P2P-based Live Video Streaming System. In Proc. of IEEE Globecom, 2008.

[24] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P.R. Rodriguez. Is high-quality vod feasible using P2P swarming? pages 903–912, 2007.

[25] Y. Zhou, D.M. Chiu, and J.C.S. Lui. A Simple Model for Analyzing P2P Streaming Protocols.

[26] R. Ahlswede, N. Cai, S.Y.R. Li, and RW Yeung. Network information flow, 2000.

[27] P.A. Chou, Y. Wu, and K. Jain. Practical network coding. 41(1):40–49, 2003.

[28] C. Gkantsidis and PR Rodriguez. Network coding for large scale content distribution. In Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, volume 4, 2005.

[29] M. Wang and B. Li. Rˆ 2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming. IEEE Journal on Selected Areas in Communications, 25(9):1655, 2007.

[30] D. Wu, Y. Liu, and K.W. Ross, "Queuing Network Models for Multi-Channel P2P Live Streaming Systems," in Proc. of IEEE INFOCOM, pp. 73-81, 2009.

[31] M. Wang, L. Xu, and B. Ramamurthy, "Linear Programming Models for Multi-Channel P2P Streaming Systems," in Proc. of IEEE INFOCOM, 2010.

[32] C. Wu, B. Li, and S. Zhao, “Multi-channel Live P2P Streaming: Refocusing on Servers,” in Proc. of IEEE INFOCOM, pp. 1355–1363, Apr. 2008.

[33] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang, “Challenges, design and analysis of a large-scale p2p-vod system,” in Proc. of ACM SIGCOMM, vol. 38, no. 4, pp. 375-388, Oct. 2008.

[34] Liu, J., Rao, S.G., Li, B., Zhang, H. "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast". Proceedings of the IEEE 96(1), 11–24 (2008)

[35] Bonald, T., Massoulie, L., Mathieu, F., Perino, D., Twigg, A.: Epidemic live streaming: optimal performance trade-offs. In: Proc. of ACM SIGMETRICS, pp. 325–336 (2008)

[36] Zhang, M., Xiong, Y., Zhang, Q., Sun, L., Yang, S.Q.: Optimizing the Throughput of Data-Driven Peer-to-Peer Streaming. IEEE Transactions on Parallel and Distributed Systems 20(1), 97–110 (2009)

[37] R. J. Vanderbei, Linear programming: foundations and extensions, 2nd Edition, Springer Press, 2001.

[38] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P.A. Chou, “Peerto-Peer Streaming Capacity,” IEEE Transactions on Information Theory,vol. 57, no. 8, pp. 5072-5087, Aug. 2011.

[39] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P.A. Chou, “P2P Streaming Capacity under Node Degree Bound,” in Proc. of IEEE ICDCS, pp. 587-598, 2011.

[40] Y. He and L. Guan, “Streaming capacity in P2P VoD systems,” in Proc. of IEEE ISCAS, pp. 742-745, May 2009.

[41] Y. He and L. Guan, “Improving the streaming capacity in P2P VoD systems with helpers,” in Proc. of IEEE ICME, pp. 790-793, Jul. 2009.

[42] Y. He and L. Guan, “Solving streaming capacity problems in P2P VoD systems,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 11, pp. 16381642, Nov. 2010.

[43] Y. He, I. Lee, and L. Guan, “Distributed throughput maximization in P2P VoD applications,” IEEE Transactions on Multimedia, vol. 11, no.3, pp. 509-522, Apr. 2009.

[44] J. Wang, C. Yeo, V. Prabhakaran, and K. Ramchandran, “On the Role of Helpers in Peer-to-Peer File Download Systems: Design, Analysis and Simulation,” in Proc. of IPTPS, Feb. 2007.

[45] P. Garbacki, D. Epema, J. Pouwelse, M. van Steen, “Offloading Servers with Collaborative Video on Demand,” in Proc. of IPTPS, Feb. 2008.

[46] C. Zhao, X. Lin, and C Wu, “The streaming capacity of sparsely connected P2P systems with distributed control,” in Proc. of IEEE INFOCOM, pp. 1449-1457, 2011.

[47] Yang Guo, Shengchao Yu†, Hang Liu, Saurabh Mathur, and Kumar Ramaswamy "Supporting VCR Operation in a Mesh-based P2P VoD System".

[48] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, C. Huang, Challenges, design and analysis of a large-scale p2p-vod system, in:SIGCOMM, 2008.

[49] X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, A measurement study of a large-scale p2p iptv system, IEEE Trans. on Multimedia 9 (2007) 1672 – 1687.

[50] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, Splitstream: High-bandwidth multicast in cooperative environments, in: SOSP, 2003.

[51] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, A. Mohr, Chainsaw: Eliminating trees from overlay multicast, in: IPTPS, 2005.

[52] V. Venkataraman, K. Yoshida, P. Francis, Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast, in: ICNP, 2006.

[53] X. Zhang, J. Liu, B. Li, T.-S. P. Yum, CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming, in: IEEE INFOCOM, 2005.

[54] K. Cho, K. Fukuda, H. Esaki, A. Kato, Observing slow crustal movement in residential user traffic, in: CONEXT, 2008.

[55] M. Meeker, D. Joseph, The state of the internet, part 3, in: Web 2.0, 2006.

[56] H. Schulze, K. Moschalski, Ipoque internet study 2008/2009,http://www.ipoque.com (2009).

[57] V. N. Padmanabhan, H. J. Wang, P. A. Chou, Resilient peer-to-peer streaming, in: ICNP, 2003.

[58] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, J. O. Jr., Overcast: Reliable multicasting with an overlay network, in: OSDI, 2000.

[59] T. Bonald, L. Massouli´e, F. Mathieu, D. Perino, A. Twigg. Epidemic Live Streaming: Optimal Performance Trade-Offs. In Proc. of SIGMETRICS, 2008.

[60] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. DONet/CoolStreaming: A data-driven overlay network for live media streaming. In Proc. of INFOCOM, 2005.

[61] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez. Randomized decentralized broadcasting algorithms. In Proc. of INFOCOM, 2007.

[62] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In IPTPS, 2005.

[63] F. Pianese, D. Perino, J. Keller, and E. Biersack. Pulse: an adaptative, incentive-based, unstructured p2p live streaming system. In IEEE Transaction on Multimedia, 2007.

[64] Yiu, W.-Pk.; Xing Jin; Chan, S.-H.G., "Challenges and Approaches in Large-Scale P2P Media Streaming," Multimedia, IEEE , vol.14, no.2, pp.50-59, April-June 2007