Overlay Management Strategies to Improve QoS in Peer-assisted Live Streaming Systems

Shilpa Budhkar

Overlay Management Strategies to Improve QoS in Peer-assisted Live Streaming Systems

Thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Shilpa Budhkar

Under the Supervision of

Dr. T. Venkatesh

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
Guwahati 781039, India
December 2019

In the Memory of

Aaji and Aazoba

Dedicated to

My Parents, Husband and Sister

For their blessings, encouragement and love

Declaration

I certify that

a. The work contained in this thesis is original, and has been done by myself under the general supervision of my supervisor.

b. The work has not been submitted to any other institute for any degree or diploma.

c. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

d. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT Guwahati
Date:

Shilpa Budhkar
Research Scholar
Department of Computer Science
and Engineering,
Indian Institute of Technology Guwahati,
Guwahati 781039, India.

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781039, India

Dr. T. Venkatesh
Associate Professor
Email : t.venkat@iitg.ac.in
Phone : +91-361-258-2366

CERTIFICATE

This is to certify that this thesis entitled "**Overlay Management Strategies to Improve QoS in Peer-Assisted Live Streaming Systems**" being submitted by **Shilpa Budhkar** to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India, is a record of the bonafide research work carried out by her under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Place: IIT Guwahati **Dr. T. Venkatesh**
Date:

Acknowledgements

I take this opportunity to thank all those individuals who directly or indirectly contributed in my PhD journey to make it a success.

First, I would like to express my eternal gratitude to my supervisor *Dr. T. Venkatesh* for his consistent support, expert advice and words of wisdom. He has been the one who nurtured me from being a mere novice to a mature researcher. I am blessed to have learnt a lot from a intellectual person like him. He has always encouraged me to pursue research with a strong stand on ethics. I am indebted to him for motivating me and believing in my abilities during the difficult phases of PhD. I have always enjoyed and benefited from countless hours of discussions with him on various topics, that has widened my perspective to become a wiser person.

I am highly grateful to the chairman of my Doctoral Committee *Prof. Diganta Goswami* for his valuable piece of advice and encouragement during my PhD journey. I would also like to thank the other members of my Doctoral Committee, *Dr. Sonali Chouhan* and *Dr. Sanasam Ranbir Singh* for their constructive feedback on my research work. It helped me to improve the quality of work and deepen the understanding of problem statements. I owe a sincere thanks to *Prof. Deep Medhi* for his support and having a few interesting technical discussions. I extend my gratitude to the anonymous reviewers of my research articles for their valuable review comments.

I am obliged to the present Director- *Prof. Gautam Biswas*, the former Director- *Prof. Gautam Barua*, all the Deans, and other management of IIT Guwahati for their efforts to make this institute one of the best across the globe and to provide us with great facilities to carry out the research. I am thankful to the current and former heads of the department of Computer Science and Engineering at IIT Guwahati, *Prof. S. V. Rao Prof. Purandar Bhaduri*, *Prof. Shivashankar B. Nair*, *Prof. Diganta Goswami* for their cooperation and efforts to provide us with departmental resources. I would like to sincerely acknowledge *Prof. Sukumar Nandi* and *Prof. Hemangee K. Kapoor* for their words of wisdom during my conversations with them.

I would like to appreciate the staff of the department of Computer Science and Engineering, *Raktajit Pathak, Nanu Alan Kachari, Bhriguraj Borah, Nava Kumar Boro, Monojit Bhattacharjee, Gauri Khuttiya Deori* and *Prabin Bharali* for their assistance, whenever I sought for it. I thank the security persons of the institute for maintaining a secure and peaceful environment to carry out research work in labs throughout the day.

I express my cordial gratitude to my teacher *Dr. Nilay Khare*, who wonderfully taught the basics of computer science during my engineering days and motivated me to pursue research from a world-class institute like IIT Guwahati. I am also grateful to my colleagues *Dr. Sandip Chakraborty* and *Dr. Ferdous Ahmed Barbhuiya* for their fruitful suggestions and discussions on various topics.

Next, I am wholeheartedly grateful to my friends at IIT Guwahati for their love, encouragement and life changing literature they shared with me during my PhD journey. The unconditional love and care given by *Anshita Mishra* and *Mahasweta Mitra* helped me to have a comfortable stay away from my family for the very first time in my life. I am profoundly beholden to my best buddies cum batchmates, *Shirshendu Das, Shashi Shekhar Jha, Nilkanta Sahu, Basant Subba* and *Mayank Agarwal* for being by my side in the happy moments and tough days of my life. They have been always a call away whenever I needed them during my sickness or the moments of distress. Their suggestions and psychological support have always encouraged me to complete my endeavors, whenever I lost track. I am blessed to have wonderful friends cum juniors *Nayantara Kotoky, Hema K. Yarnagula, Dipika Deb* and *Sukarn Agarwal*, with whom I had many exciting discussions on various aspects of life during evening tea or long walks. They helped me in evolving as a person and developing diverse perspective. A hearty acknowledgement goes to *Satish, Sibaji, Durgesh, Deepak, Debanjan* and *Shuvendu* for their timely assistance, whenever I sought for it. I appreciate *Achyutmani, Prateek, Rupesh, Swarup, Saroj, Ashutosh, Trishna, Maharishi, Smrati* and *Atanu* for their concern and amusing conversations. I would also like to acknowledge the love and respect given by kids of Shishugram orphanage and their caretakers. The smiles on their faces and their optimistic attitude towards life used to give me strength, inspiration and positivity to move ahead in my life.

I fall short of words to express my gratitude for my family. The prayers, love, hard work and sacrifices of my mother- *Mrs. Snehal Budhkar* and father- *Mr. Shirish Budhkar* have been my strength and a source of inspiration during my PhD journey. They dealt with the difficult moments with my least assistance to ensure that I could focus on my work. They always emboldened me to stand strong by truth, honesty and hard work. My deepest gratitude goes to my husband- *Mr. Gaurav*

Bhandari, for his encouragement, understanding and support towards my endeavors. He took care of all the family responsibilities alone and assured nothing bothers me during my work. I thank my brother- *Gunjan Budhkar* for his love and inquisitive technical questions that made me think differently. I am heartily grateful to my in-laws for their backing and understanding towards my work commitments. I also appreciate the best wishes, love and concern of the other members of my joint family and close relatives, At last, a special thanks goes to my cousin *Ananya Sawe* for her encouragement and trust on my calibre that helped me to overcome the tough phases of my PhD life. The fun filled long conversations with her used to refresh me on stressful days and kept me connected with my family though staying at a distance.

Place: IIT Guwahati **Shilpa Budhkar**
Date:

Abstract

With the increase in demand for live streaming services over Internet, Content Delivery Networks (CDN) are overloaded resulting in poor Quality of Service (QoS). To alleviate this problem, peer-assisted live streaming systems are being deployed to take advantage of self-scalability of peer-to-peer (P2P) overlay networks. In peer-assisted live streaming systems, peers retrieve stream content from other peers in the overlay apart from the servers. A major challenge in deploying these systems is to ensure QoS, while dealing with heterogeneity in lifetime and bandwidth of peers. This dissertation presents a few overlay management strategies to improve the QoS of peer-assisted live streaming systems.

First, we study the limitations of different overlay topologies in providing QoS under flash crowd, churn and heterogeneous bandwidth of peers. We found that the multi-tree overlay shows poor streaming quality, startup delay and resilience during churn, while the mesh overlay has larger playback delay and jitter. The hybrid tree-mesh overlay requires to minimize the delay of the mesh sub-overlay and precise prediction of stability of peers to organize them in the tree sub-overlay. Considering these issues, we propose a three-stage peer selection strategy to build a minimum delay mesh overlay. In the first stage, the tracker suggests some peers as prospective partners to a new peer. In the second stage, the new peer selects its partners out of these peers to retrieve stream content. The third stage is the topology adaptation phase of peers, where peers reposition themselves in the overlay to minimize the delay during peer churn. In all the stages, peers are selected considering upload capacity, propagation delay, buffering level and per-hop chunk buffering duration. The proposed peer selection strategy is compared with two existing strategies and the results show that the playback delay is reduced significantly along with a marginal improvement in startup delay.

Next, we propose an overlay management strategy for hybrid tree-mesh overlays to improve streaming quality, startup delay and upload capacity utilization despite heterogeneity in lifetime and bandwidth of peers. It organizes peers based on their serviceability, defined in terms of stability, streaming quality, upload and download capacities. The peers with higher upload capacity are part of an extended CDN tree to facilitate stable seeders. A part of the upload capacity of the existing peers is reserved to form *virtual sources* that provide startup chunks for quick playback. The peers joining the mesh overlay select partners with the highest serviceability to ensure better streaming quality. They also adapt the topology by replacing partners based on serviceability and upload capacity utilization to maintain QoS during churn. A comparison with existing strategies shows that streaming quality and startup delay is improved significantly due to better upload capacity utilization of peers. It also results in offloading the CDN servers. Overall, the proposed overlay management strategies are found to improve the QoS in mesh and hybrid overlays that can further improve the scalability of peer-assisted live streaming systems.

Contents

List of Figures

List of Symbols

Symbol	Description
N	Number of peers
l	Length of streaming session
C	Streaming rate or Chunk rate (chunks per unit time) generated by the source
$C_s(t)$	Chunk number of the latest chunk generated by source at time t
$C_i(t)$	Chunk number of the latest chunk available at i^{th} peer at time t
P	Set of peers suggested as prospective partners to the new peer
K	Number of peers suggested as prospective partners
n	New peer
U_i^a	Aggregate upload capacity of a peer i
U_i^r	Residual upload capacity of a peer i
D_i^a	Aggregate download capacity of a peer i
B_i	Buffering level of a peer i
$D_{n,i}$	Propagation delay from peer i to the new peer n
L_i	Playback lag experienced by a peer i
T	Set of ancestors of i^{th} peer with lesser aggregate upload capacity
H_j	Highest chunk number available at an ancestor j
TTL	Time-to-live of a message sent by a peer (in number of hops)
Z	Set of all CDN servers
d_i	Geographical distance of the CDN server i from the new peer n
$subA$	Sub-overlay of CDN server A
F_i	Fraction of peers experiencing larger than average joining delay and inadequate chunk rate in the sub-overlay of server i
Z_A	Suitability value of a server A, where $A \in Z$
U_A^r	Residual upload capacities of server A
$Child$	Set of tree peers which are directly connected to the CDN server
U_{subA}^a	Aggregate upload capacity of the sub-overlay of CDN server A
D_{subA}^a	Aggregate download capacity of the sub-overlay of CDN server A

LIST OF SYMBOLS

Symbol	Description
U_n^A	Fraction of upload capacity of new peer allocated to server A
D_n^A	Fraction of download capacity of new peer allocated to server A
λ	Peer arrival rate in peers/unit time
R	Average streaming rate desired by arriving peer (chunks/unit time)
X	Set of peers with reserved upload capacity to create virtual sources
X_i	Amount of reserved upload capacity of a peer i, $i \in X$
M	Total number of sub-streams created by the source
$q_{i,j}$	Quality of the i^{th} sub-stream received by a peer j
$N_{i,j}$	Number of chunks of i^{th} sub-stream received by a peer j before playback deadline
l_j	Elapsed session duration of j^{th} peer
l_e	Elapsed stream duration
N_i	Number of chunks received by i^{th} peer before playback deadline
Q_i	Quality of stream received by the i^{th} peer
S_i	Serviceability value of i^{th} peer
N_i^t	Number of sub-streams delivered by the i^{th} peer at time t
T_i	Utility of the i^{th} peer (sum of its serviceability value and upload capacity utilization)

Chapter 1

Introduction

With the increased use of high bandwidth links in the last mile of the Internet, the demand for live video streaming services has surged tremendously [2,3]. A variety of applications that include streaming of live events, Internet TV, online games, and distance education are responsible for the growth of video traffic. In a traffic forecast report given by Cisco, it is reported that the video traffic is expected to increase from 73% in 2016 to 82% of all Internet traffic by 2021 [4]. Of this, 13% is expected to be live video traffic by 2021. Content Delivery Networks (CDN) are used traditionally to manage such an enormous volume of live video traffic on the Internet. It is estimated that by 2021, CDNs are expected to support 77% of all Internet video traffic [4]. The CDN infrastructure consists of a large number of cache servers deployed at geographically distributed locations and Internet Service Providers (ISPs). For example, Akamai has deployed more than 137,000 servers in 87 countries [5].

Building and maintaining the CDN infrastructure requires a huge amount of investment by the operators [6]. Despite this, the number of users that can be supported is limited by the number of servers, which results in limited scalability [7, 8]. Studies show that during peak hours of the demand, CDNs get overloaded and lack in fulfilling the Quality of Service (QoS) requirements of users [9,10]. QoS can be defined in terms of requirement for high streaming quality, low startup delay and low playback delay. The playback delay is the time between generation of the video frames till they are played by users. The startup delay is defined as the waiting time experienced by a user before the playback since joining the session. Streaming quality experienced by a user is defined as the streaming rate that it receives [11].

In a recent article on live streaming of FIFA World Cup and Wimbledon 2018, it is found that CDNs were able to provide ultra high definition quality video to upto 60,000 users, with more than one million users waiting for similar quality [12]. It is also noticed that there is a trade-off between minimizing the latency and improving the streaming quality. Another study on live video streaming traffic reported that more than 20% of users in over 200 million users experienced interruptions in the playback for more than 10% of the time [10]. The startup delay experienced was greater than ten seconds and over 10% failed to watch the video. With the users being sensitive towards the QoS perceived, it was observed that 1% increase in the duration of playback interruptions reduces the viewing time of the users by more than 3 minutes [13]. The demand for better QoS requires the operators to deploy huge infrastructure [14]. The attrition in number of users/subscribers due to poor QoS adversely impacts the revenue and profit of the CDN operators and content providers likewise [15].

Considering the limited scalability of CDNs, research community is exploring to augment them with peer-to-peer (P2P) overlay networks, termed peer-assisted live streaming systems. These systems exploit the benefits of P2P overlays such as self-scalability, easy deployment and cost-effectiveness, to meet the ever increasing demand of streaming services without investing in larger infrastructure [11, 16]. However, these systems also inherit the innate limitations of P2P overlays such as handling peer churn and flash crowd as well as dealing with heterogeneous resource contribution of peers [17, 18]. Consequently, ensuring guaranteed QoS to users in peer-assisted live streaming systems is still an open research problem.

An Overview of Peer-assisted Live Streaming Systems: In peer-assisted live streaming systems, the streaming server(s) stream data to some of the users. These end users or peers then form an overlay with other peers to deliver the stream. In a P2P overlay, the end users act both as client and server, such that they forward received stream data to each other. Peers contribute a part of their resources, such as computing power, storage capacity and network bandwidth to the system [19]. This makes sure that an increase in the number of users also increases the streaming capacity of the system. Trends show that the growth in demand of resources is always less than the availability of resources in such systems [20]. Some of the popular peer-assisted live streaming systems are LiveSky [14], PPLive [21], CoolStreaming [22], and SopCast [23].

While analyzing the performance of these systems, authors of [24] reported that PPLive was able to provide streaming service to millions of users with the help of only 5 Mbps to 10 Mbps of server bandwidth. Authors of [25] also reported that only 30% of peers contributed upto 80% of upload capacity required by the system in Coolstreaming. Considering the benefits of peer-assisted live streaming systems, various commercial CDNs also modified their infrastructure to support P2P overlays [26, 27]. For example, Akamai acquired RedSwoosh P2P system in 2007 and started its hybrid CDN-P2P service named NetSession in 2010 [26]. Akamai offloaded streaming traffic of CDN servers by up to 66% using P2P overlays [5]. Few other studies also report that with the help of P2P overlays 50% to 88% stream delivery traffic can be offloaded from the servers [20, 26, 28]. Hence, the inclusion of P2P overlays with CDNs also reduces the cost of content delivery and increases the profit margin of both CDN operators as well as content providers [29].

Despite the benefits of scalability and cost-effectiveness, peer-assisted live streaming systems lack in the ability to maintain the desired QoS due to unreliable and heterogeneous resource contribution of peers as discussed further.

Challenges of Peer-assisted Live Streaming Systems: The major challenge of peer-assisted live streaming systems is to provide guaranteed QoS to users in terms of startup delay, playback delay, streaming quality and stream continuity, while dealing with heterogeneity in session duration and upload capacity of peers [17]. The session duration of a peer, which is the time from the joining till its departure, also represents the stability of peer. The session duration of peers varies because the peers independently join and leave the system based on their interest in the content and the quality of experience [30]. This leads to varying number of peers and varying upload capacity contributed to the system during the session. The upload capacity is also heterogeneous due to different types of Internet subscriptions used [17]. This unreliability in peer behavior and heterogeneity makes it difficult to maintain QoS because the resources of peers contribute largely to the capacity of the system.

It is essential to deal with this issue because the QoS perceived by peers impacts their resource contribution and session duration [13]. The authors of [21,22] concluded that larger percentage of peers leave the system if they experience higher startup delay and lower streaming quality. The authors of [31] concluded that peers contribute higher amount of upload capacity if the streaming quality is good initially. The trace-based studies in [32,33] also reported that the peers accept startup delay in the range of 15 seconds to 45 seconds and about 10 to 20 percent degradation in streaming quality. However, the acceptable playback delay depends on the type of application [19]. For example, in video conferencing applications it needs to be within hundreds of milliseconds, whereas it could be upto few minutes for Internet TV. Large playback delay experienced by peers also leads to the departure of peers. Authors of [34] presented a Bayesian network model on user behavior which shows that session duration and upload contribution of peers depend on parameters such as streaming quality, elapsed session duration, playback delay, startup delay, peer arrival and departure rate. Therefore, to increase the lifetime of peers and their upload contribution, it is necessary to maintain an acceptable value of all the QoS parameters during the session.

1.1 Motivation for the Research Work

Some of the popular peer-assisted live streaming applications like Skype, Tribler-Swarm Player, PPLive and BBC iPlayer, have been using P2P overlays or have used them earlier [21, 26, 35, 36]. Since the release of Skype application in 2003, it had been using P2P overlay for serving millions of users [37]. It implemented hierarchical overlay architecture by making peers with sufficient resources as super nodes [38]. The super nodes act as relay nodes to maintain connection between peers. It is found through a study that an efficient selection of super nodes helps in mitigating the impact of churn on the QoS [39].

Despite its success, Skype suffered from severe outages in year 2010, affecting millions of users [40]. Fig. 1.1 shows how the number of Skype users swelled up starting from October 2010 [1]. One of the reasons for the failure of Skype to cope up with this surge is found to be its complex overlay management algorithm and heterogeneous resource contribution of peers [40, 41]. A work in [42] also found that the super node selection algorithm used in Skype is far from optimal.

Tribler-Swarm Player uses BitTorrent protocol for stream delivery [43]. The major issue with the Swarm Player is found to be large startup delay [44]. It is reported that peer selection time and stream downloading from selected peers results in an unacceptable startup delay.

The BBC iPlayer used P2P overlay to provide its stream delivery services for a few years until 2008. Later, it started using HTTP downloads because downloading

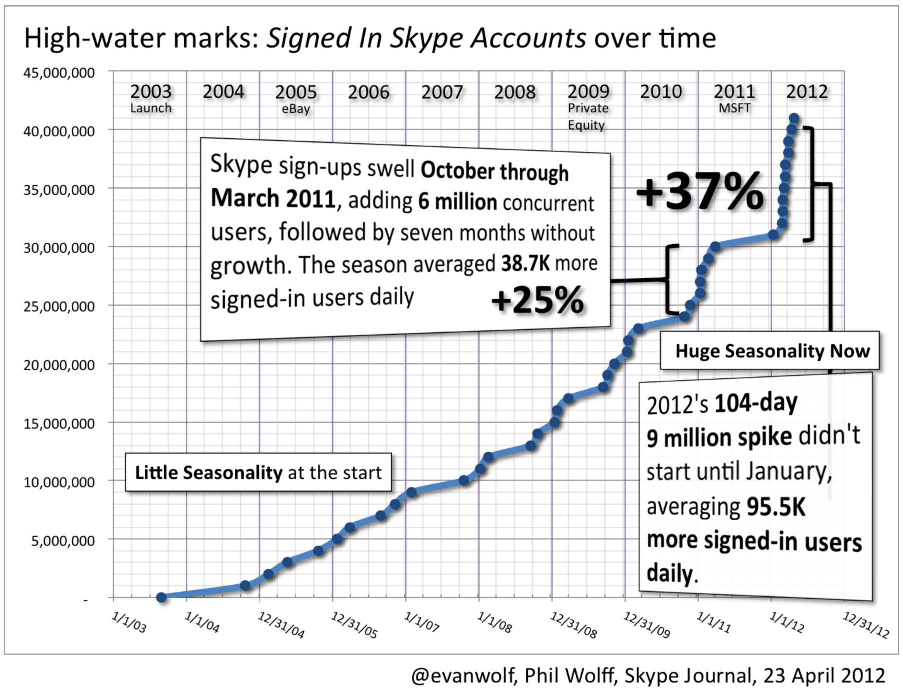


Figure 1.1: Surge in number of users in Skype in 2010 [1]

content directly from the servers have become more cost-effective solution than maintaining P2P overlay architecture [45–47]. Despite this, it also supports an additional P2P plug-in to provide high definition video at low cost when bandwidth consumption is more. It is also predicted that with the use of TV set-top boxes by millions of users, the bandwidth consumption will increase 10 times [45]. To deal with this surge in demand of bandwidth, P2P overlays might become the most cost-effective solution.

The above mentioned challenges experienced by some real peer-assisted live streaming applications motivate us to develop some overlay management strategies to deal with peer heterogeneity and provide desired QoS to a large number of users.

To provide guaranteed QoS, existing peer-assisted live streaming systems organized peers into different P2P overlay topologies like, tree, multi-tree, mesh, and hybrid tree-mesh [19]. Initially, tree topologies were used to create P2P overlays, in which the peers follow a parent-child relationship and the parent peer pushes content to its children. Poor resilience under peer churn and improper utilization of upload capacity of peers in the tree overlays, led to the development of multi-tree and mesh overlays. Multi-tree overlays divide the stream data into multiple sub-streams and deliver each sub-stream using a different tree [48]. Peers join multiple trees to download the desired number of sub-streams. The complex structure of multi-tree overlay makes it difficult to maintain during peer churn [49]. In mesh overlays, peers connect mutually following a partner or neighbor relationship to exchange stream data [21, 22]. The stream data is exchanged as equal and small-sized blocks called chunks, that are distributed using arbitrarily long paths leading to high playback delay and jitter [49]. To overcome the shortcomings of tree and mesh overlays, hybrid tree-mesh overlays were developed. In these overlays, the tree part is created using peers that are predicted to be stable to increase its resilience [50]. Considering that no peer can continue forever, it is required to predict the stability of peers precisely. It is also required to minimize the playback delay of the mesh part to leverage the benefits from the tree part.

In the research literature, there are several ways in which the overlay construction and management are handled to improve QoS. These are typically done through better peer selection strategies [51–53], chunk scheduling strategies [54, 55] and by organizing peers considering heterogeneity in upload capacity and session duration [56, 57]. The work in this thesis only focuses on designing peer selection strategies and organizing the peers in the overlay considering heterogeneity.

Peer Selection Strategies: In most of the existing approaches, peer selection is done in two stages [51–53]. In the first stage, the bootstrap node or tracker suggests a few prospective peers/partners to a newly joined peer. In the second stage, the new peer selects its partners from these to get the stream data. The selection of partners is mostly random in the first stage, whereas in the second stage, the peers are selected by considering parameters like upload capacity and propagation delay. The existing strategies neglect other parameters like, chunk buffering duration at each hop and buffering level of peers, that have significant impact on playback delay and startup delay experienced. The chunk buffering duration is the time for which a chunk is stored at a peer before being forwarded. The buffering level is the number of contiguous chunks available in the buffer and it indicates the chunk upload rate of the peer. To minimize the delay in the overlay topology we need to consider these additional parameters during peer selection as well as re-selecting the peers during churn.

Organizing Peers Considering Heterogeneity: Some studies showed that the upload capacity of peers can be utilized better in live streaming compared to video on-demand streaming, due to the synchronized streaming and higher lifetime of peers in the system [17, 28]. Despite this peer-assisted live streaming systems suffer from poor QoS and inefficient utilization of the upload capacity of the peers. The main reasons for this are reported to be the heterogeneity in session duration (lifetime) and upload capacity of peers [58].

In the existing literature, the stability of peers and their upload capacity contribution are predicted using the session duration [56]. Peers with higher stability and upload capacity are organized to form a backbone tree in the overlay to ensure better QoS to other peers [57, 59]. This only utilizes peers with longer session duration and/or higher bandwidth. Moreover, they do not consider received streaming quality while predicting stability, and session duration of all the peers remains short and almost same initially during the session. The literature also does not consider parameters like, availability of chunks, upload and download capacities, and the stability altogether, in arranging the peers. We define serviceability of peer as a parameter that captures the effect of all these parameters on its suitability. We argue that it is required to predict the serviceability of peers and then strategically exploit the capacity of peers with heterogeneous session duration and/or upload capacity to improve QoS.

1.2 Contributions of the Thesis

The main objective of this thesis is to design overlay management strategies that improve the QoS of peer-assisted live streaming systems. Towards this, we first demonstrate the limitations of different P2P overlay topologies in delivering QoS, using a trace-based simulation. We show the impact of flash crowd, peer churn, and heterogeneous upload capacity of peers on the QoS considering different overlay topologies. We use parameters such as playback delay, startup delay and streaming quality. We also evaluate the stability of peers during churn using peer joining failure rate, parent re-selection rate, and stream recovery duration.

Based on this study, we conclude that multi-tree overlay gives highest startup delay because, peers take longer time to select parents and change them multiple times before playback startup. The unavailability of chunks with the selected parents/partners also adds to the startup delay. Contrarily, the playback delay is least because of lower per-hop chunk buffering delay from the source whereas, it is the highest in mesh overlay. We also observe that streaming quality of peers in hybrid overlay is lower than the mesh overlay because of the bottleneck bandwidth and departure of intermediate tree nodes. Multi-tree shows poorer streaming quality. We notice that mesh overlay exhibits lower parent re-selection rate and stream recovery duration compared to other overlays. Hence, it is more robust to churn.

Considering these observations, we propose a peer selection strategy for mesh overlays to minimize the playback delay experienced by the peers. Then, we propose an overlay management strategy for a hybrid tree-mesh overlay to enhance the utilization of upload capacity of peers and also to improve the QoS. We elaborate these two major contributions in detail.

1.2.1 A Peer Selection Strategy for Mesh Overlays

We propose a peer selection strategy to minimize the delay in mesh overlay using three different stages. During peer selection we consider parameters such as upload capacity, propagation delay, chunk buffering duration and buffering level. In the first stage (tracker-tier), the tracker selects prospective partners considering upload capacity and buffering level of the peers. At the second stage (peer-tier), a peer selects its partners out of those suggested by the tracker considering their upload capacity, propagation delay, chunk buffering duration and buffering level. The last stage is the topology adaptation stage, where the peers replace their ancestors considering same parameters to reposition themselves closer to the source and to maintain minimum delay during churn.

The performance of proposed strategy is compared with two existing strategies, Fast-Mesh [51] and HLPSP [60] implemented in OMNeT++ simulator. The key observations from the results are as follows:

- The playback delay is reduced by 30-32% under no churn scenario and by 15-20% with peer churn. This is because peers with higher upload capacity are positioned closer to the source which reduces the path length as well as per-hop latency.

- The parameter-based peer selection at all the stages increases the startup delay. However, selection of peers considering availability of chunks compensates this increase in startup delay by reducing buffering delay significantly. Overall, the startup delay is reduced by 20-25% under no churn scenario and upto 10% with churn.

- It takes 8% lesser time to stabilize after flash crowd because, peers avoid selecting newly joined peers with fewer chunks. The recovery time is also upto 20% lower during peer departure because buffering level is considered in re-selecting partners.

1.2.2 An Overlay Management Strategy for Hybrid Tree-Mesh Overlays

We propose an overlay management strategy that organizes peers based on their serviceability, defined in terms of stability, streaming quality, upload and download capacities. A peer joins multiple sub-overlays, each of them built and maintained by a CDN server. To join the sub-overlay, a peer uses a CDN server selection strategy that considers geographical proximity of servers, stream quality and joining delay experienced by peers in the sub-overlay. Next, the peer uses a bandwidth allocation mechanism to share its bandwidth among joined sub-overlays.

Within a sub-overlay, the peers are organized in a hybrid tree-mesh topology. The peers with higher upload capacity are part of an extended CDN tree with the server at the root. Other peers in the mesh topology receive startup chunks from virtual sources, created by reserving some upload capacity of existing peers. Each mesh peer selects partners considering their serviceability. The mesh peers also use a topology adaptation strategy to replace partners based on serviceability and upload capacity utilization to maintain QoS during churn.

We implemented the proposed strategy in OMNeT++ simulator and compared its performance with two existing systems, PROSE [57] and LiveSky [14]. The key observations from the results are as follows:

- The proposed overlay management strategy exhibits 20% lower startup delay because peers join sub-overlays with relatively lesser load. The delivery of startup chunks with the help of dedicated virtual sources also ensures quick startup with desired streaming quality, while avoiding the CDN server overloading.

- The streaming quality experienced by peers is also improved by 25% because higher upload capacity seeders are created as a part of the extended CDN tree, as well as partners with high stability, availability of chunks, and free upload capacity are selected. The topology adaptation also helps in improving streaming quality because peers replace partners with those who provide better streaming quality consistently.

- The upload capacity utilization of peers is enhanced by 30% using proposed strategy because it strategically utilizes upload capacity of all the peers with heterogeneous session duration and upload bandwidth. This also reduces the stream delivery load of CDN servers by 20%.

- The stability of peers is improved by 20% because they get better streaming quality and lower startup delay. This also leads to an improvement in their upload capacity contribution by approximately 15%.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows:

- **Chapter 2** provides a background on the architecture of peer-assisted live streaming systems and the literature on overlay construction and management.

- **Chapter 3** presents a simulation study of multi-tree, mesh and hybrid tree-mesh overlays with respect to their QoS and the stability of peers in the overlay.

- **Chapter 4** discusses the work done to minimize the delay in mesh overlays where a three-stage peer selection strategy to reduce the playback delay is discussed.

- **Chapter 5** presents an overlay management strategy for the hybrid tree-mesh overlay to enhance the upload capacity utilization of peers and to improve the QoS despite the heterogeneity of peers.
- **Chapter 6** summarizes the thesis and suggests a few possible extensions to this work.

Chapter 2

Background and Literature Survey

In this chapter, we first describe the architecture of peer-assisted live streaming systems briefly, along with the different modules. Then we present a survey of the literature pertaining different overlay construction and management strategies to ensure QoS.

2.1 Architecture of a Peer-assisted Live Streaming System

A peer-assisted live streaming system typically consists of a source server, a tracker or CDN manager, several CDN servers and peers [61]. The source server encodes the stream and distributes it through the CDN servers, as shown in Fig. 2.1. The CDN servers act as reliable and high upload capacity seeders for the P2P overlay to disseminate content to the peers. The tracker acts as a directory server to maintain the information about CDN servers and peers. When a new peer joins the system, the tracker redirects it to either the CDN server or the peers.

Current peer-assisted systems use different methods to support newly joining peers. In one method, the tracker provides a list of some existing peers to a new

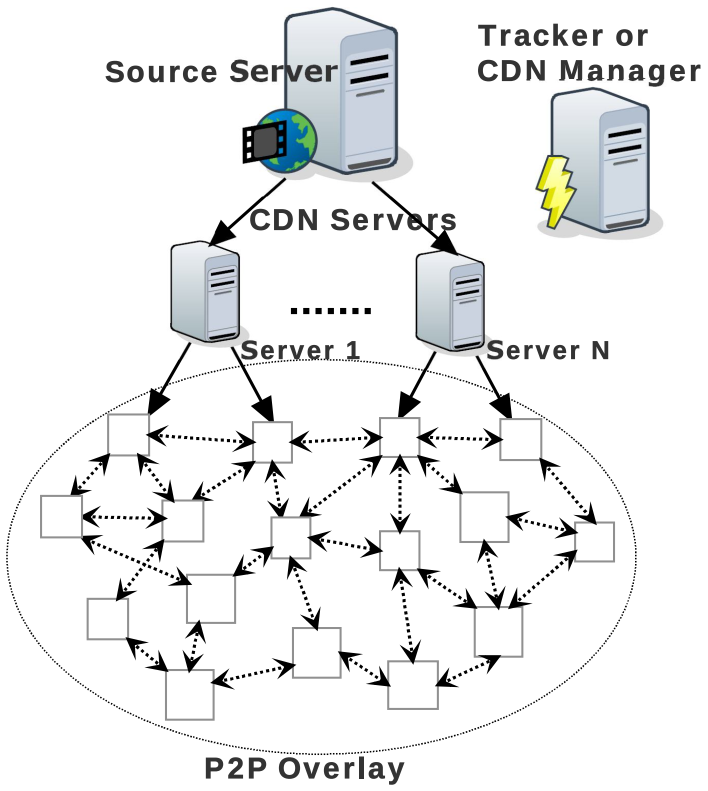


Figure 2.1: Architecture of a peer-assisted live streaming system

peer [21]. If the new peer does not receive satisfactory streaming service from the suggested peers then, it also gets the information about other peers from those connected or from the tracker. In another method, the tracker redirects the new peer to a nearest CDN server using DNS based redirection [14,62]. The CDN server either serves the new peer directly or gives a list of others peers from the same domain. There are three major functions in building and maintaining the overlay; peer selection, scheduling the data delivery to other peers, and to adapt the overlay after peer churn or when the streaming quality is not satisfactory.

On the basis of arrangement of peers, the overlays are broadly classified as tree, mesh and hybrid tree-mesh overlays. The data dissemination in these overlays could be push-based, pull-based or push-pull based [19].

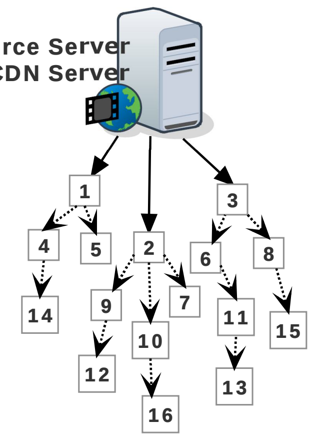


Figure 2.2: Tree overlay

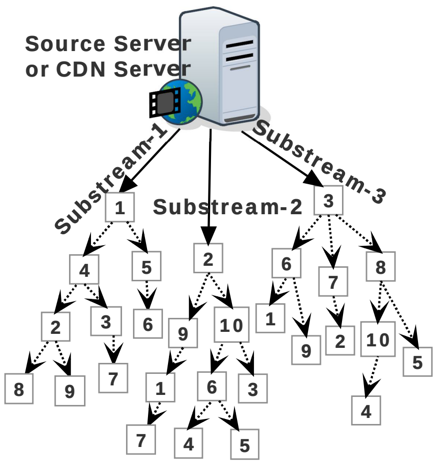


Figure 2.3: Multi-tree overlay

Tree Overlays: In the tree overlays, peers follow a parent-child relationship. The peers push the stream data to its children without waiting for any explicit request. A few well known live streaming systems that use tree overlays are CoopNet [63], Splitstream [48] and Chunkyspread [64]. The tree overlays are further categorized into single tree and multi-tree overlays. Fig. 2.2 shows a single tree overlay, where each peer has only one parent who sends full stream to the peer. In multi-tree overlays, the video stream is divided into multiple sub-streams corresponding to different layers in encoding [48,63], as shown in Fig. 2.3. Each sub-stream is delivered to peers using a different tree. The peers can join multiple trees (and have multiple parents) to receive one sub-stream from each tree. The number of sub-streams subscribed depends on the desired quality of streaming.

Mesh Overlays: In the mesh overlays, peers connect to each other using a partner or neighbor relationship, where peers act as both the parent and a child, as shown in Fig. 2.4. The streaming server first divides the stream data into small and equal sized blocks called chunks, which are distributed through the overlay. The peers exchange a buffer map with the partners to inform about the chunk availability in the buffer. The peers follow either pull-based approach for data delivery or push-pull based approach. In the pull-based approach, a peer requests its partners to provide specific chunks that are not available in its buffer. In the push-pull data delivery, a peer requests different set of chunks from each partner. On receiving the request for first chunk, partners push all the chunks in the requested set. Some of the popular peer-assisted systems that use mesh overlays are PPlive [21], Coolstreaming [22] and Fast-Mesh [51].

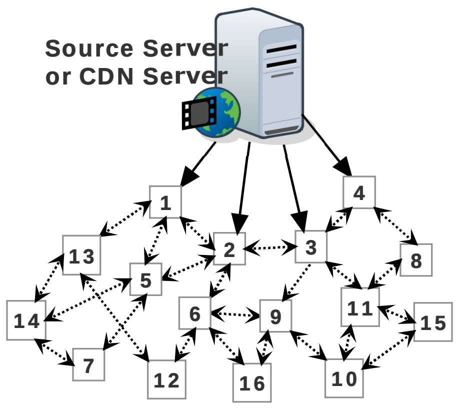


Figure 2.4: Mesh overlay

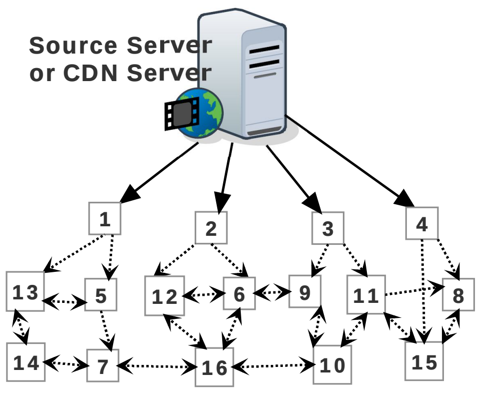


Figure 2.5: Hybrid tree-mesh overlay

Hybrid Tree-Mesh Overlays: In the hybrid tree-mesh overlays, a fraction of the peers are a part of the tree topology while the rest connect to each other using a mesh topology, as shown in Fig. 2.5. Most of the existing hybrid systems use peers with higher predicted stability to form a resilient backbone tree while the remaining peers connect at the edges of the tree creating mesh. The tree peers maintain a parent-child relationship among themselves and use push-based data delivery approach to disseminate chunks, whereas the mesh peers use push-pull based approach. A few peer-assisted systems that use hybrid tree-mesh overlay are mTreebone [50], Anysee2 [65], and ToMo [66].

2.2 Literature Survey

In the literature, various aspects of overlay construction and management are explored to improve the QoS while dealing with heterogeneous peers. In this section, we present a brief summary of the papers addressing specific aspects namely, peer selection strategies, chunk scheduling mechanisms, and approaches to overlay formation considering the heterogeneity.

2.2.1 Impact of Overlay Topologies on QoS

There are few studies in the literature that compare the performance of different overlay topologies based on QoS and stability. The authors of [67] compared the resilience of various tree and multi-tree overlays and found that multi-tree is more resilient to churn because departure of one parent causes loss of only one sub-stream. Others studied parameters such as streaming quality, upload capacity utilization, scalability, playback delay, churn adaptability and bandwidth heterogeneity to understand the impact of overlay topology on the QoS. The work in [68], compared multi-tree and mesh topologies to find that multi-tree has lower bandwidth utilization because the intermediate nodes with low bandwidth create a bottleneck for high quality stream dissemination. Multi-tree also exhibits higher rate of changing the ancestors and frequent deadlock events because the departure of an internal node in multi-tree impacts all its descendants. The authors of [69] compared multi-tree and mesh topologies to find that mesh topology gives better streaming quality and can cope with flash crowd or peer departure easily due to its simpler structure. On the contrary, multi-tree results in lower latency due to its push-based data delivery mechanism.

Table 2.1: Summary of existing works comparing performance of the Multi-tree, Mesh and Hybrid tree-mesh Overlays

Existing Comparison Works	**Compare**				**Parameters used for comparison**					
	Tree	**Multi-tree**	**Mesh**	**Hybrid**	**Resilience to churn**	**Bandwidth Utilization**	**Overlay Stability**	**Streaming Quality**	**Playback delay**	**Startup Delay**
[67]	✓	✓	–	–	High: Multi-tree	–	–	–	–	–
[68]	–	✓	✓	–	–	High: Mesh	High: Mesh	–	–	–
[69]	–	✓	✓	–	High: Mesh	–	–	High: Mesh	Low: Multi-tree	–
[70]	✓	–	✓	–	–	–	–	High: Mesh	Low: Tree	–
[71]	✓	–	✓	–	–	–	–	–	Low: Tree	–
Our simulation study	–	✓	✓	✓	✓	–	✓	✓	✓	✓

The work in [70] studied the impact of two peer selection strategies, random selection and neighbor-with-nearby-peers, on playback delay and chunk loss rate for both tree and mesh overlays. They found that playback delay is higher in mesh for both peer selection strategies. The chunk loss rate is lower in tree overlay with random selection. In [71] it was concluded that the delay incurred to distribute chunks is higher in mesh due to sub-optimal height of the diffusion tree in mesh. They also reported that the height of the tree can be reduced, but it also increases the number of children at each hop and correspondingly the time taken to send chunks to the children.

Overall, Table 2.1 shows that the existing studies compared only tree and mesh overlay topologies. We noticed that the existing literature does not compare the performance of multi-tree, mesh and hybrid tree-mesh overlays in terms of the streaming quality, playback delay, startup delay and stability during churn. The conclusions from such a study can be used to organize the peers in the tree and mesh parts of the hybrid overlay to exploit their respective advantages. We conduct a detailed simulation study to understand the impact of the overlay topology on QoS and use the conclusions from the study in designing other strategies to improve the QoS. The details of this simulation study are presented in Chapter 3.

2.2.2 Peer Selection Strategies

In mesh overlays, storing and forwarding chunks at each hop from the source increases the playback delay. It also takes time to search for existing peers and fetch chunks from them, which also affects the startup delay. In general, a random selection of peers results in increased playback delay [49]. The studies in [21, 22] showed that peers experiencing larger playback delay or startup delay are more likely to leave the system. Therefore, to improve the lifetime and upload contribution of peers it is necessary to reduce the playback delay and startup delay experienced by peers.

There are several peer selection strategies in the literature that mostly focus on minimizing playback delay and startup delay experienced by peers in mesh overlays, as shown in Table 2.2. In [53], the selection of peers was prioritized based on their upload capacity and latency. The authors of [72] additionally considered availability of stream data at buffers while selecting peers. In [73], peers with high fanout capacity were prioritized, where fan-out capacity of a peer is measured in terms of its upload capacity and the playback delay. The peers with high fan-out value were placed closer to the source to meet the delay bounds of peers. In [52], it was proposed to keep peers with higher upload capacity closer to the source to improve the stream quality and playback delay experienced. In [74], peers pull more content from high bandwidth partners to make upload rate proportional to their upload capacity. The authors of [75] proposed to arrange peers in different layers in the overlay based on their delay requirements. In all these approaches, peers were first selected randomly by the tracker and then based on a few parameters by the peers. However, in [76] peers were selected considering their ability to deliver the stream both by the tracker as well as by the peers. This ability was measured using outdegree and online time.

The authors of Fast-Mesh [51] also proposed a peer selection strategy to minimize playback delay of mesh overlays. The tracker randomly suggests a few recently joined peers as prospective partners to a new peer. The new peer computes *power* of each prospective partner and selects those with higher power greedily as partners. *Power* of a partner is computed as the ratio of its residual upload bandwidth to its source-to-end delay. During churn, peers are also replaced considering their upload bandwidth and source-to-end delay.

The selection of newly joined peers reduces the probability of selecting peers not having free upload capacity, but increases the probability of selecting peers without sufficient data in their buffers. This increases the startup delay of peers.

The authors of [22] also reported that the selection of newly joined peers increases the startup delay. Furthermore, the source-to-end delay in Fast-Mesh only includes the transmission and propagation delays at each hop. Chunk buffering duration at each hop and delay due to freezes, when there is insufficient data in the buffer are not considered. We argue that the playback delay can be significantly reduced by considering these two additional parameters during peer selection.

2.2.3 Chunk Scheduling Mechanisms

The chunk scheduling mechanisms developed so far focus on speeding up the diffusion of chunks across the overlay and to improve the streaming continuity/quality. In these mechanisms, chunks are selected either randomly or prioritized based on latest chunk, rarest chunk or nearest deadline chunk strategies [77, 78]. Receiving peers are selected either randomly or prioritized based on upload bandwidth or the most deprived peer is prioritized. With the combination of these chunk and peer selection strategies, the variety of chunk scheduling mechanisms have been developed.

The authors of [77] proved that *Random peer, latest useful chunk* mechanism was able to achieve optimal chunk dissemination rate and delay. In this mechanism, the receiver peer is selected randomly out of the ones which need the latest chunk available at the sender peer. In [79], it was proposed that latest chunk and high bandwidth peers are selected first for faster chunk dissemination. The authors of [80] evaluated the performance of different mechanisms and found that selection of latest chunk helps in faster diffusion of chunks but to a limited number of peers, whereas random chunk selection exhibits better stream continuity and lower playback delay. In [78] it was showed that *Nearest deadline chunk and bandwidth-aware peer* selection results in least chunk delivery delay. The authors of [81, 82] explored the trade-offs of rarest-first and nearest-deadline-first chunk selection strategies and proposed a mixed strategy, where the chunks are first retrieved using rarest-first-strategy and if

Table 2.2: Summary of existing peer selection strategies

Existing Peer Selection Strategies	**Parameters used for Peer Selection at**	
	Level-1: Tracker Level	**Level-2: Peer Level**
[51]	Random and recently joined peers	Residual upload capacity and playback delay (upload capacity + propagation delay) + Adaptation considering upload capacity and distance
[52]	Random	Upload capacity
[53]	Random	Upload capacity and playback delay
[72]	Random	Upload capacity, playback delay and chunk availability
[73]	Random	Upload capacity and playback delay
[74]	Random	Upload capacity
[75]	Random	Acceptable playback delay
[76]	Upload capacity and online time	Upload capacity and online time
Contribution-1: Proposed Peer Selection Strategy	**Upload capacity and buffering level**	**Upload capacity, playback delay, buffering level, availability of most recently created chunks** + **Level-3: Topology Adaptation considering same parameters**

sufficient number of chunks are not received, then the nearest-deadline-first strategy is used.

To exploit the bandwidth heterogeneity of peers, the authors of [54, 83, 84] proposed to prioritize selection of peers with higher upload bandwidth. In [83], snowball streaming algorithm was proposed, where a peer keeps disseminating the latest chunk received until all its partners received that chunk. A partner with higher upload capacity was selected first. The authors of [54, 84] also reported that *Latest Useful chunk, Strict-Priority Peer* strategy exhibited lower delay and faster chunk diffusion. In this mechanism, the peer selects one of the latest chunks required and a peer with the highest upload bandwidth is prioritized.

2.2.4 Organizing Peers Considering Heterogeneity

In the literature, most of the existing systems organize peers based on their stability and upload capacity. This is because the peers with longer elapsed session duration are predicted to be more stable [50, 56]. The stable peers are maintained as a part of the backbone tree in the overlay. In [57], authors proposed that CDN servers serve only those peers labelled as choke point expansion nodes and super nodes. The choke point expansion nodes are those that supply data to a large number of peers, but the demand is more than the supply. The super nodes are the ones with higher upload capacity, are highly stable, and have sufficient data in buffers. In [59], it was suggested that the CDN servers serve the peers with highest stability. The peers with longest elapsed session duration are considered most stable. The authors of [85] proposed that peers keep only superior peers (or stable peers) as their partners. A superior peer is identified using a *superior index*, calculated as the product of predicted session duration and average upload bandwidth. In [56], the stability of peers was predicted using elapsed session duration and a backbone tree of stable peers was created to ensure better QoS.

Overall, Table 2.3 shows that the strategies focused on utilizing upload capacity of only stable and high bandwidth peers neglect the others. They also neglect parameters like streaming quality and bandwidth that impact the prediction of stability of peers. These parameters also cause the departure of peers in future. It is also difficult to predict the stability of peers at the beginning of a streaming session using these techniques because, all the peers have short as well as almost equal elapsed session duration.

Some earlier studies also recommended prioritizing the delivery of stream based on the playback deadline, a summary of these shown in Table 2.4 [61,86,87]. In [61] it was proposed to consider three regions of the buffer *i.e.*: startup region, common region and emergency region. The chunks of the startup region are prioritized over common region. The chunks of the emergency region are delivered by the CDN server only when other partners fail to deliver them and the playback deadline is close. In [86], the CDN servers maintain three queues to handle chunk requests. The first queue stores the requests of all new joining peers in the first-come-first-serve (FCFS) order. The second queue stores chunks which are not served by peers and chunks closer to the playback deadline, in the order of playback deadline. The third queue is for those chunks which are available only at the server. In [87], a peer first fetches chunks from the other peers that belong to same cluster considering the availability of chunks in their buffers. If chunks are not delivered within a threshold, the peer requests the CDN server for the chunks.

Table 2.3: Summary of existing systems organizing peers in the overlay considering their heterogeneity

Existing Systems	**Overlay Topology**	**Parameters considered for**			**Remark**
		CDN server selection	**Organizing peers closer to CDN servers**	**Parent/partner selection**	
[8, 14]	Hybrid tree-mesh	Geographical proximity, Load of CDN server	[14]Peers with upload capacity >0 [8] Only higher upload capacity peers	Random	*** Do not consider stability or serviceability of peers for partner selection * New peers joining CDN servers increases load of servers * Predict peer stability using only session duration * Do not consider instantaneous streaming quality and download capacity to predict peers serving capability * Do not utilize resources of less stable and low upload capacity peers**
[56]	Hybrid tree-mesh	DNS redirection	High stability	Random	
[57]	Mesh	DNS redirection	High stability, upload capacity and chunks availability; New peers join CDN only to receive startup chunks	Random	
[59]	Hybrid tree-mesh	DNS redirection	High stability and upload capacity	Random	
[85]	Mesh	DNS redirection	High stability and upload capacity	High stability and upload capacity	

Continued on next page...

Table 2.3: Continued....

Existing Systems	Overlay Topology	Parameters considered for			Remark
		CDN server selection	Organizing peers closer to CDN servers	Parent/partner selection	
Contrib-ution-2: Proposed Strategy	**Hybrid tree-mesh**	**Geographical proximity, load of CDN servers, joining delay and stream quality of peers in sub-overlay**	**High upload capacity**	**Serviceability (Session duration, streaming quality, upload and download capacity); Re-select regularly considering serviceability and upload capacity utilization**	**Utilize upload capacity of peers by creating virtual sources to provide startup chunks**

In LiveSky [14] all the peers are initially connected to edge nodes to receive startup chunks and reduce startup delay. The peers that contribute upload capacity by becoming a LiveSky client are ensured better QoS. However, the peers that contribute more upload capacity are not prioritized. In [8], it was proposed to select LiveSky clients based on their upload contribution and regular stream delivery performance.

All the proposed techniques utilize the upload capacity of CDN servers to provide startup chunks and missing chunks. As a result, the number of peers that receive assured QoS is limited by the capacity of servers. Moreover, the CDN servers may also get overloaded due to the geographically skewed distribution of peer population [14]. To overcome these issues, it is required to utilize the upload capacity of all the peers for quick startup and to deliver the missing chunks to peers.

Table 2.4: Summary of existing systems prioritizing delivery of stream chunks

Existing Systems	Priority classification	Prioritizing delivery of stream chunks by		Remarks
		CDN server	Peer parents/partners	
[14]	Types of peers: Legacy and LiveSky	Standard quality stream delivery to legacy and LiveSky peers	Additional chunks to LiveSky peers for better streaming quality	Mostly startup chunks and missing chunks with closer playback deadline are delivered by servers
[57]	Divide server bandwith for: 1. Startup chunks 2. Stream delivery to Supernodes	1. First come first serve (FCFS) 2. Proactively push chunks to supernodes	All stream chunks to ordinary peers	
[61]	Divide peers buffer into: 1. Startup region 2. Common region 3. Emergency region	Emergency chunks when playback deadline is close	Startup chunks and common chunks	
[86]	CDN servers maintain queues: 1. New peers 2. Chunks requested closer to playback deadline 3. Chunks available only with server	1. FCFS: Least priority 2. Earliest deadline first: Highest priority	Rest of the chunks	

Continued on next page...

Table 2.4: Continued....

Existing Systems	Priority classification	Prioritizing delivery of stream chunks by		Remarks
		CDN server	Peer parents/partners	
[87]	Types of chunks: 1. Startup chunks 2. Undelivered chunks closer to playback deadline 3. Other common chunks	1. Startup chunks 2. Undelivered chunks closer to playback deadline	3. Other common chunks	
Contribution-2: Proposed Strategy	**Types of peers:** **1. Tree peers- High upload capacity** **2. Mesh peers- Low upload capacity**	**Full stream to few high upload capacity peers**	**Most of the tree peers and all mesh peers**	**Startup and missing chunks are delivered by peers**

2.3 Summary

In this chapter, we presented some important strategies in overlay construction and management. We covered various peer selection strategies used in mesh overlays to minimize playback delay. In most of these, the tracker suggests prospective peers at random, whereas peers select their partners considering propagation delay and upload capacity. These strategies do not consider chunk buffering duration at each hop and buffering level of peers that significantly impact the playback delay. This motivates us to propose a peer selection strategy that considers these parameters at all the stages of overlay construction. Next, we explored various mechanisms to organize peers in the overlay considering their heterogeneity. These mechanisms lack in precisely identifying the stable and high upload capacity peers to improve QoS. Moreover, they do not consider utilizing the upload capacity of peers with short session duration and/or low bandwidth. These shortcomings motivate us to propose an overlay management strategy which organizes peers in the overlay considering parameters that impact QoS.

While, there are some studies which compare tree and mesh overlay topologies based on parameters such as streaming quality, playback delay and peer heterogeneity, there is no study which explores the constraints of hybrid tree-mesh overlays. Hence, in the next chapter we explore the limitations of multi-tree, mesh and hybrid tree-mesh overlays using simulations based on streaming quality, playback delay, startup delay and stability, particularly under peer churn. The results help us in identifying the abilities of different overlay topologies to meet the QoS requirements.

Chapter 3

A Simulation Study to Explore the Limitations of Overlay Topologies

3.1 Introduction

With the existence of different overlay topologies; tree, mesh, and multi-tree in peer-assisted live streaming, a few studies in the literature also studied the performance of multi-tree and mesh overlays based on scalability, streaming quality, bandwidth utilization, churn handling and bandwidth heterogeneity. The work in [68] studied the impact of churn and bandwidth heterogeneity on the bandwidth utilization and streaming quality. It was found that multi-tree performs poorly because intermediate nodes create bottleneck and their departure results in higher ancestor changing rate and frequent deadlock events. Authors of [69] compared the streaming quality of the overlays using parameters such as throughput, goodput and continuity index. They observed that mesh showed better streaming quality while coping with churn and that multi-tree overlay leads to lower latency while dealing with heterogeneity.

Overall, the existing studies compared only tree and mesh overlay topologies. None of the existing works compared multi-tree, mesh and hybrid tree-mesh overlays in terms of the streaming quality, playback delay, startup delay and stability during churn. Such a study would help to precisely exploit the benefits of tree and mesh topologies while creating hybrid tree-mesh overlays to improve QoS. It also provides insights for organizing peers in the tree and mesh part of the hybrid tree-mesh overlay considering their heterogeneity in lifetime and bandwidth.

In this chapter, we explore the limitations of multi-tree, mesh and hybrid tree-mesh overlay topologies in delivering QoS, using a trace-driven simulation study. We first study the various modules of overlay construction and management like peer joining procedure, parent or partner selection procedure, data delivery mechanism, and procedure to handle peer departures. Then we evaluate how these modules cope up with flash crowd, peer churn and heterogeneous upload capacity of peers to maintain QoS. The parameters used for the evaluation of QoS are playback delay, startup delay and streaming quality. We also assess the stability of peers during churn using parameters such as peer joining failure rate, parent re-selection rate and stream recovery duration. The simulation results show that the multi-tree overlay gives highest startup delay, whereas playback delay is least. The mesh overlay is found to provide better streaming quality, exhibits lower parent re-selection rate and stream recovery duration during churn, whereas it gives highest playback delay. The hybrid-tree mesh overlay results in slightly lower startup delay, playback delay and streaming quality compared to the mesh overlay.

Organization of the Chapter: The rest of the chapter is organized as follows. Section 3.2 provides an overview of the construction and management of the different overlay topologies. The evaluation criteria and results are discussed in Section 3.3. Section 3.4 concludes the chapter with an insight for the future work.

3.2 Construction and Management of Overlay Topologies: An Overview

In this section we provide an overview of the multi-tree, mesh and hybrid tree-mesh overlay topologies based on three aspects of overlay construction and management; peer joining procedure, data delivery mechanism, and procedure to handle peer departure.

3.2.1 Multi-tree Overlays

Multi-tree overlays are developed to improve the resilience and upload capacity utilization of peers in single tree overlays [19]. Tree overlays have poor resilience under peer churn because each peer is connected to only one parent to receive the full video stream, the departure of which disrupts the playback continuity for all the descendant peers [88, 89]. In multi-tree overlay, each tree delivers a sub-stream [48]. Hence, the departure of a single parent disrupts the delivery of only one sub-stream. The upload capacity utilization of peers is lower in single trees because upload capacities of leaf nodes are not utilized. Considering this, peers in multi-tree overlays join at least one tree as an intermediate node to utilize their upload capacity [48, 63]. Moreover, the intermediate nodes with lower upload capacity create a bottleneck in tree overlays. This issue does not exist in multi-tree overlays due to the use of multiple parents. Some of the popular multi-tree overlays are CoopNet [63], Splitstream [48] and Chunkyspread [64].

Peer Joining Procedure: The number of trees a new peer joins in the overlay depends on its download capacity and video encoding scheme used. If download capacity of a new peer is greater than or equal to the full streaming rate then it joins all the trees. If video is encoded using schemes like scalable video coding

(SVC) and multiple description coding (MDC), then peers join the number of trees equivalent to number of subs-streams it is capable of downloading [48, 63]. These schemes ensure that the download of larger number of sub-streams enhances the streaming quality perceived by peers.

To join multiple trees, new peers find a parent in each tree with the help of server or root of the tree. In Splitstream, the new peer sends join request to the root [48]. If the root does not have enough free upload capacity to accommodate more children then it sends the list of its children. The new peer selects a closest peer as the parent and sends join request. This procedure continues till a new peer finds the parent with free upload capacity. This also helps in adding the new peers at the shortest depth in the trees. In CoopNet [63] the server finds a parent for the new peer in each tree, whereas in [64] new peers select a parent randomly to download a portion of the stream. While selecting parents, the new peer considers their data delivery load and latency. Each peer joins at least one tree as an interior node and all other trees as a leaf node in [48,63]. This creates interior-node-disjoint trees. The number of internal nodes in all the trees are maintained to be nearly the same to ensure balanced upload capacity contribution and stream delivery load.

Data Delivery Mechanism: In multi-tree overlays, peers use push-based data delivery mechanism. The peers keep pushing the stream chunks to their children without waiting for any explicit request. The chunks pushed by a peer depend on the sub-stream requested by its children during parent selection.

Handling Peer Churn: The departure of peers in P2P overlays can be either graceful or abrupt. During graceful departure the peer notifies its parent and children (or the server) about its departure, whereas peers find out about abrupt departure of the parent by monitoring loss of data.

In [19,49,63], root of the tree or server provides the information of new parents to the orphaned peers, whereas in [48] the orphaned peers follow the peer joining mechanism to rejoin the disconnected trees. Due to peer departure interior-node-disjoint trees suffer from a deadlock condition, where trees run out of free upload capacity to accommodate more leaf nodes [48, 68]. This particularly occurs when significant number of interior nodes of one tree leave the overlay in a short span of time. In such conditions the orphaned leaf nodes keep trying to join the tree. To ease this rejoining process, a spare capacity group of peers is created using peers with free upload capacity. The orphaned leaf nodes find a parent from this group that provides the desired sub-stream and also that does not create a cycle in any other tree. To avoid creating a cycle, each peer of the spare capacity group maintains the information of path from the root.

The complex structure of the multi-tree overlays and the difficulty to maintain it during peer churn led to the development of mesh overlays [90].

3.2.2 Mesh Overlays

The mesh overlays are built with the goal of creating a robust and simpler topology compared to multi-tree overlays. Some of the popular mesh overlays are Prime [91], Chainsaw [92], Coolstreaming [22, 93], PPLive [21] and Fast-Mesh [51]. In mesh overlays, peers maintain a partial list of others peers in the overlay and connect with them randomly to exchange stream chunks. Peers maintain a partner or neighbor relationship, where they act as both parent and child for each other. Each peer maintains a playout buffer to collect stream chunks and arrange them in the playout order before delivering to the player.

Peer Joining Procedure: Most of the mesh overlays consist of a tracker or a rendezvous point in addition to the server and peers in the system. The tracker helps new peers in joining the overlay by providing them a partial list of active peers of the overlay. The new peer then creates its own list of active peers using this suggested list. Each peer selects its partners to obtain stream chunks from the list of active peers. In some mesh overlays peers select their partners randomly [21,22,91], whereas in Fast-Mesh [51] partners are selected considering source-to-end delay in stream delivery. A new peer select its partners greedily considering its *power*, that is defined as the ratio of the residual upload bandwidth and source-to-end delay.

Data Delivery Mechanism: Peers use buffer maps to represent the chunks available in the buffer of peers. Each peer regularly exchanges its buffer map with all its active peers. This helps peers in fetching required stream chunks from partners. A peer fetches chunks from its partners using either pull-based mechanism or push-pull based mechanism. In pull-based data delivery, a peer requests its partners for specific chunks that are not available in its buffer. Each peer first creates a schedule for requesting chunks with the help of buffer maps of its partners. The schedule decides which chunks need to be requested from which partner. Different mesh overlays use different scheduling algorithms; Chainsaw [92] uses random chunk, random peer scheduling and Prime [91] uses latest useful chunk, random peer scheduling. In [93] peers select partners based on chunk delivery delay and available bandwidth.

The pull-based data delivery mechanism avoids the delivery of duplicate chunks and the wastage of bandwidth, but it introduces the overhead of sending one extra message for each chunk and delays the delivery of chunks. To avoid this push-pull based mechanisms are used [22, 51]. The server first decomposes the video stream into multiple sub-streams by grouping chunks. The peers retrieve one sub-stream from each partner with the help of buffer maps. On receiving the request for first chunk, partners push the subsequent chunks continuously.

Handling Peer Churn: The peers in mesh overlays follow a self-organization mechanism to maintain playback continuity during partners departure. During graceful departure of a partner, the peer quickly selects another partner from its list of active peers and request desired chunks [22, 91, 92]. However, peers in most of the mesh overlays identify abrupt departure of partners by monitoring chunk upload rate or periodic messages [91, 92]. In [22], peers keep track of the chunk delivery efficiency of its partners based on deviation between the number of chunks in any two sub-streams and the availability of chunks at the partners. Each peer considers replacing its partner with another one when it finds that the partner is not providing sufficient upload capacity or recent chunks. In [51] a peer determines abrupt departure of its partners by monitoring periodically exchanged *Keep-Alive* messages with its parents and children. Peers also adapt the overlay by regularly replacing their ancestors to keep higher upload capacity peers closer to the server.

3.2.3 Hybrid Tree-Mesh Overlays

Hybrid tree-mesh overlays combine the benefits of tree and mesh topologies. In most of the hybrid overlays, peers are arranged in two layers, where one layer follows tree topology and another layer uses mesh topology [50, 56, 60, 66, 94], while peers are arranged in three layers in [95]. In [50, 56, 94] peers with higher stability are part of the tree topology, closer to the server, whereas others form a mesh at the edge of the tree. In [60] peers with similar upload capacities create a cluster using mesh topology. These mesh clusters are then arranged to form a multicast tree with server being the root. The clusters with peers having higher upload capacities are positioned closer to the server.

In contrast to these hybrid overlays, peers in [66] are organized to create an auxiliary mesh topology at top layer and bottom layer consists of tree topology. The hybrid overlay created in [95] uses three layers. On the top layer mesh topology is formed, middle layer arranges peers in tree topology and the bottom layer creates mesh clusters of geographically closer peers.

In all cases mentioned above, a fixed topology is used for the hybrid overlay, whereas authors in [96] proposed a transition between multi-tree and mesh depending on peer dynamics and QoS perceived. Under highly dynamic environment, the topology converges into mesh topology because peers frequently update their partners, whereas under stable scenario peers maintain a multi-tree topology. At any instant different peers in the overlay connect with a different topology based on the received streaming quality.

Peer Joining Procedure: In [50, 56, 94] the new peers initially join the mesh topology. Later, they are added to tree topology based on the lifetime or elapsed session duration. In [50], a peer is considered eligible to become a tree node when its elapsed session duration becomes thirty percent of the residual session length. The work in [56] also proposed that a peer can be considered stable if it has already stayed for the duration equal to half of the residual session length. Authors of [95] proposed that new peers are added to a suitable part of the topology based on their reputation, measured based on its lifetime, upload capacity and delay from the source server. In [66], initially all the new peers are added to the mesh topology at the top layer until a specific threshold of hopcount. Subsequently, the new peers are also added to the tree topology maintaining a higher fraction of mesh peers.

Data Delivery Mechanism: The peers joining tree topology (tree peers) use push-based mechanism, whereas the mesh peers use either pull or push-pull based data delivery mechanism. The mesh peers receive chunks from either tree peers or mesh peers. In [50] the stable peers of the tree topology receive most of the chunks from its parent tree node, but also maintain auxiliary links with other peers to pull chunks in case of packet loss. Peers in [66] use push-based data delivery at both the layers. In the auxiliary mesh topology at the top layer, the peers are arranged to form multiple sub-trees. Within a sub-tree, peers use push-based data delivery. However, sub-trees are also clustered to facilitate mesh links between peers of different sub-trees. To avoid the overhead of extra messages for creating mesh connections, the peers maintain the mesh links for a short period of time to deliver a specified number of chunks. Hence, chunks are delivered using push-based mechanism using mesh links. In [95], peers use push-based data delivery at the top and middle layer. The peers at the top layer also maintain auxiliary mesh links to retrieve chunks if they experience data inadequacy. At the bottom layer, the peers within a mesh cluster use pull-based data delivery, except the cluster head that receives and delivers chunks using push-based data delivery. In [96] peers use push or pull-based data delivery depending on the duration of a connection. In an extremely dynamic scenario when peers show unreliability, each and every chunk is pulled using an extra request message.

Handling Peer Churn: Most of the hybrid overlays handle the departure of peers with the help of mesh topology. The work in [94, 95] proposed that the orphaned/dependent tree peers use the mesh peers for retrieving lost chunks temporarily to maintain playback continuity. In [50], each peer maintains a partners list of some active peers. During departure of a mesh peer its dependent peers select another partner from this list. If a tree peer leaves, then the orphaned peers use auxiliary mesh links to pull lost stream chunks temporarily and also search for new parent with free capacity out of tree peers. If there is no such parent available, then it replaces a mesh peer that is connected to a tree peer to receive stream.

Authors of [56] proposed to utilize residual upload capacity of tree peers to recover lost chunks during peer churn. Each tree peer connects with other tree peers to pull chunks when parent peers do not provide adequate number of chunks either due to packet loss or departure. In [60] peers detect the departure of partners by examining inadequate delivery of chunks or *Keep-Alive* messages. Peers inform the tracker about the departure of their partners to get a list of other active partners in the overlay. In [66], orphaned peers rejoin the overlay by sending a join request to the server. If server does not have free upload capacity, then it redirects the orphaned peer randomly to other peers in the overlay, which continues till the orphaned peer finds a new parent.

3.3 Performance Evaluation

In this section we study the performance of multi-tree, mesh and hybrid tree-mesh overlays using trace-based simulations. We simulate Splitstream [48] as a multi-tree overlay, Coolstreaming [22] as a mesh overlay and mTreebone [50] as a hybrid tree-mesh overlay. The performance of the overlays is evaluated based on startup delay, playback delay and streaming quality experienced by peers. We also evaluated stability under peer churn scenario using parameters such as peer joining failure rate, parent re-selection rate and stream recovery duration. Next, we present the simulation scenarios and the metrics used for the evaluation.

3.3.1 Evaluation Scenarios

The overlays are implemented in a discrete event simulator, OMNeT++ [97]. Its OverSim [98] framework is used to simulate P2P overlay and underlay TCP/IP network is simulated using INET [99] framework. The simulation parameters are selected based on the trace-based studies conducted on real peer-assisted live streaming systems.

Table 3.1 shows the values of various parameters used in the simulation setup. The source server generates video stream at the rate of 360 kbps considering that a study in [21] reported PPLive supported streaming rates ranging from 250 kbps

Table 3.1: Simulation Setup Parameters

Parameter	**Value**
Streaming rate	360 Kbps
Number of sub-streams	6
Number of peers	10,000
Length of streaming session	100
Peers inter-arrival time	0.5 - 5 secs
Flash crowd duration	5 min (in the beginning of session)

to 400 kbps. The source server generates stream in the form of small-sized data blocks called chunks. The chunks are arranged into six sub-streams of 60 kbps each. The download capacity of approximately 90% peers is set to be equal to or greater than streaming rate to capture the bandwidth heterogeneity among peers [48], as shown in Table 3.2. The upload capacity distribution of peers is given in Table 3.3, where thirty percent peers are free riders. This distribution follows a bandwidth measurement study of a real-world streaming event, where the data was collected from servers of Akamai over a 3-month period from October 2003 to January 2004 [100]. We simulated a streaming session of 100-minutes with up to ten thousand peers joining the overlay during the streaming duration.

Flash Crowd and Churn Settings: The peer churn is simulated with the help of LifeTimeChurn configuration of OMNeT++. Flash crowd is simulated during initial five percent duration of the streaming session [32, 100]. Peer arrival rate follows Poisson distribution with mean inter-arrival time between 0.5 and 5 secs [96, 101]. To simulate peer departure, the lifetime or session duration of peers is set using Pareto distribution [56, 102]. It represents that the age of peers is an indicator of their expected remaining time in the system. Peers leave the overlay

Table 3.2: Download Capacity Distribution

Percentage of peers	Download Capacity
50%	360 Kbps
25%	512 Kbps
15%	720 Kbps
10%	280 Kbps

Table 3.3: Upload Capacity Distribution

Percentage of peers	Upload Capacity
30%	0 Kbps
58%	16 Kbps - 160 Kbps
5%	320 Kbps - 1.4 Mbps
7%	1.6 Mbps

either after completing its specified lifetime or due to unacceptable startup delay and/or streaming quality. Each peer tolerates unacceptable streaming quality for one minute and leaves the overlay if it does not improve. The peers are set to randomly tolerate 10 to 20 percent degradation in desired streaming rate and 15 to 45 seconds of startup delay [30, 32, 103].

3.3.2 Evaluation Metrics

We assess the performance of the different overlay topologies based on their stability during peer churn, startup delay, playback delay and streaming quality experienced by peers. The various metrics used in the evaluation are defined as follows.

- **Startup delay:** It is the waiting time experienced by a peer before it starts the playback since it joins the streaming session. The startup delay consists of two parts: 1) Parent/partner selection delay- It is the time spent by a new peer in selecting parents or partners to retrieve stream chunks, and 2) Buffering delay- It is the delay in receiving stream chunks from selected parents or partners.

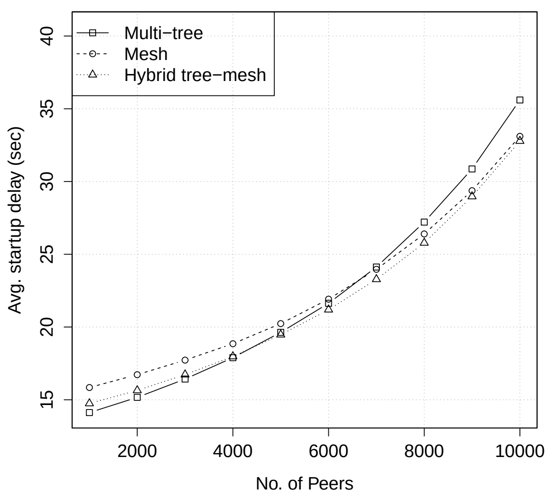
- **Joining failure rate:** It is the fraction of peers who fail to join the streaming session due to unacceptable startup delay.

- **Streaming quality:** It is the percentage of chunks delivered to a peer before the playback deadline compared to the total number that should have been received for a desired streaming rate.
- **Stream recovery duration:** It is the duration for which peers experience degradation in streaming quality before re-selecting parent/partners during peer departure and recovering the desired quality.
- **Parent/partner re-selection rate:** It is the fraction of peers who re-select parents or partners either due to peer departure or during overlay adaptation.
- **Playback delay:** It is the time taken to deliver chunks to a peer from the source.
- **Hopcount:** It is the average number of hops traversed by chunks.
- **Push/pull ratio:** It is the ratio of number of chunks pushed to those pulled by peers while using push-pull based data delivery mechanism.
- **Chunk buffering duration:** It is the duration for which a chunk is buffered at a peer before being forwarded to another peer.

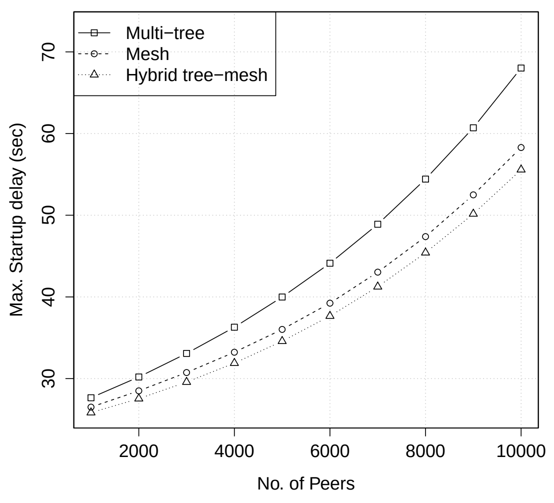
3.3.3 Results and Discussion

First, we study the reasons behind the startup delay perceived by peers in the multi-tree, mesh and hybrid overlays. Then we investigate the stability of different overlays during peer churn and its impact on streaming quality. Finally, we also examine the playback delay experienced by peers and its causes.

Evaluating Startup Delay: Fig. 3.1 shows the average and maximum startup delay perceived by peers for different number of peers in the multi-tree, mesh and hybrid tree-mesh overlays. It is observed that the multi-tree overlay exhibits highest



(a) Average startup delay

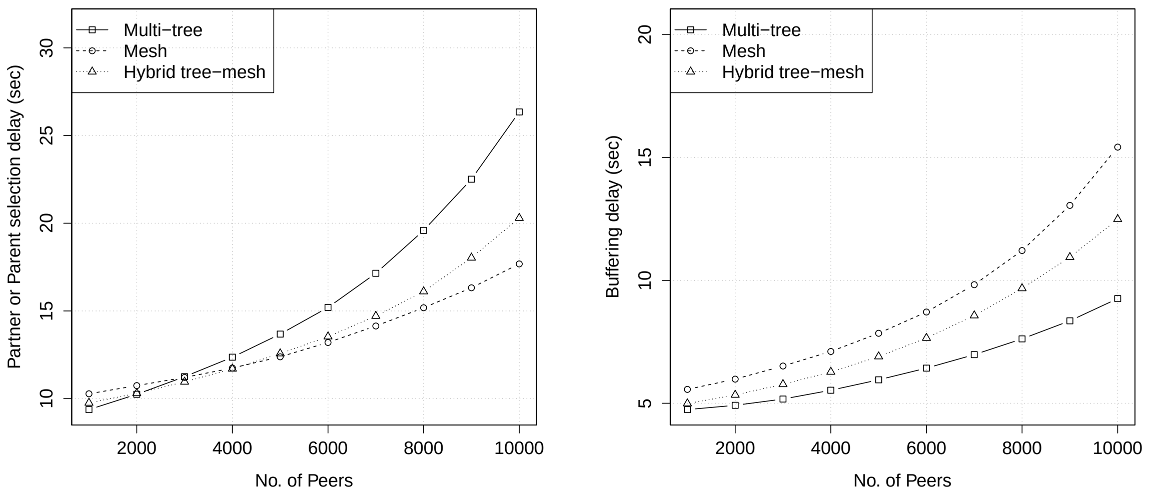


(b) Maximum startup delay

Figure 3.1: Comparing startup delay perceived by the peers

and hybrid overlay exhibits least startup delay. It is also seen from Fig. 3.1a and Fig. 3.1b that when the number of peers is small, average startup delay of the multi-tree overlay is lesser compared to the mesh and hybrid overlays, whereas maximum startup delay of the multi-tree overlay is always highest. To record the maximum startup delay experienced by peers we disabled peer departure due to unacceptable startup delay in this simulation.

To understand the reasons behind these results, we split the startup delay into two parts: parent/partner selection delay and buffering delay. Fig. 3.2 compares these two parts of the startup delay for the three overlays. It can be observed that the parent selection delay of the multi-tree overlay is highest, whereas buffering delay is lowest compared to other overlays. It can also be observed from Fig. 3.2a and Fig. 3.1a that parent selection delay and startup delay of the multi-tree overlay follows a similar pattern. The delay is low when there are fewer peers in the overlay, but it increases with the number of peers. We conclude that the startup delay



(a) Parent or partner selection delay (b) Buffering delay

Figure 3.2: Comparing components of startup delay

perceived by peers in the multi-tree overlay is more influenced by the parent selection delay compared to the buffering delay.

We observe that the multi-tree overlay results in higher parent selection delay because each new peer traverses multiple trees hop by hop till it finds a parent with spare upload capacity in each tree. When there are fewer peers in the overlay, a larger percentage of new peers find parents with spare capacity easily. Hence, the startup delay and parent selection delay of the multi-tree overlay are lower. The parent selection delay of the multi-tree overlay is also higher due to multiple re-selection of parents before playback startup during peer churn. Fig. 3.5b shows that a larger percentage of peers re-select parents before playback in the multi-tree overlay. This observation is discussed further while discussing the stability of overlays under peer churn.

The mesh overlay has least partner selection delay because the bootstrap node suggests partners randomly and the new peer retrieves chunks considering availability. The hybrid overlay results in larger delay compared to the mesh because the source server takes longer time in providing a list of existing peers. In the list, the source server adds some stable tree peers and some mesh peers.

Fig. 3.1 and Fig. 3.2 also show that the mesh overlay results in higher startup delay compared to hybrid, despite lower partner selection delay. Higher buffering delay negates the lower partner selection delay and raises the total startup delay in the mesh overlay. To verify this, Fig. 3.2b presents the buffering delay experienced by peers for the three overlays. It can be observed that buffering delay of the multi-tree overlay is least, whereas it is highest for mesh overlay. The multi-tree overlay exhibits least buffering delay because chunks are delivered using push-based data delivery mechanism, whereas the mesh and hybrid overlays use push-pull based data delivery mechanism. Push-based data delivery lowers the buffering delay.

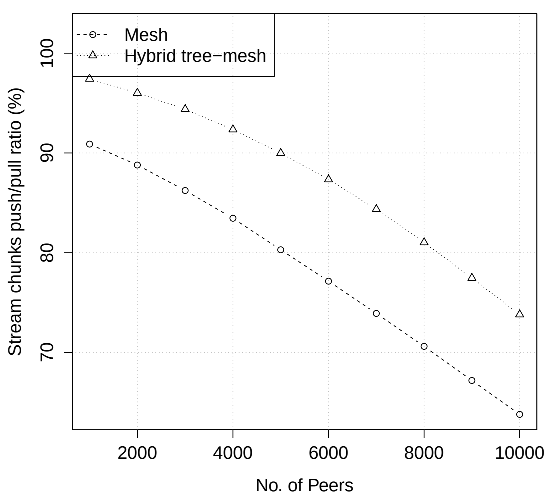


Figure 3.3: Comparing push/pull ratio of chunks delivered

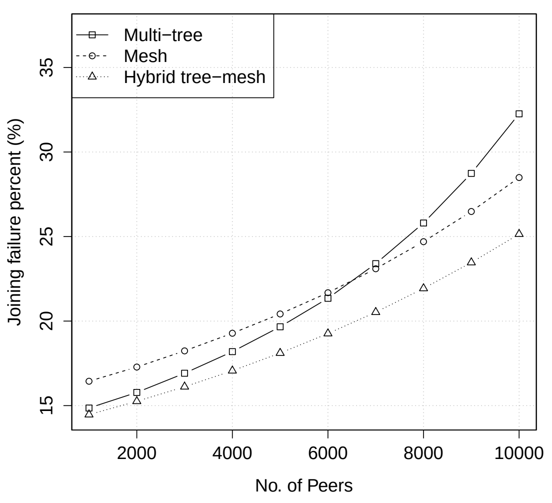


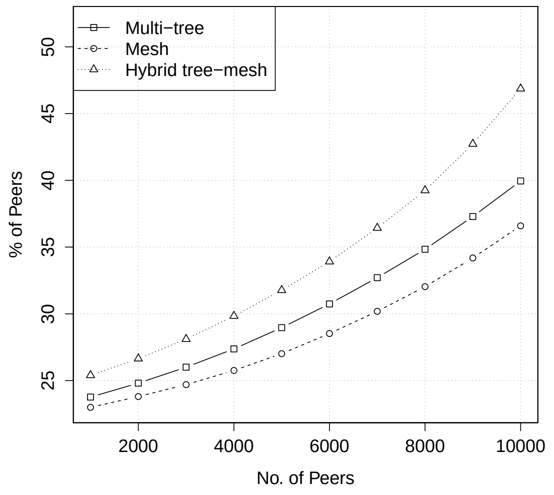
Figure 3.4: Comparing percentage of peers experiencing joining failure

Though hybrid and mesh overlay use same push-pull based data delivery mechanism, the buffering delay is lower for hybrid overlay. This is because peers in the backbone tree use push mechanism to deliver chunks. It can be observed from Fig. 3.3 that a larger percentage of chunks are delivered using push-based data delivery mechanism in the hybrid overlay. Another reason for higher buffering delay in mesh overlay is the random selection of partners that do not either have chunks in the buffer or the required residual upload capacity.

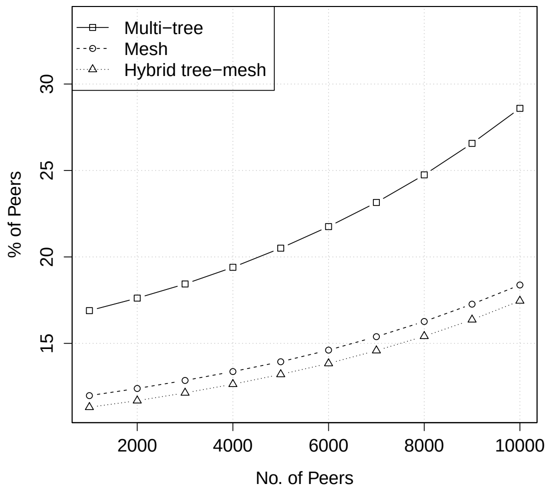
Overall, larger parent selection delay negates the reduction in the buffering delay in the multi-tree overlay and causes higher startup delay compared to the mesh and hybrid overlays. On the other hand, larger buffering delay in the mesh overlay increases the startup delay compared to the hybrid overlay.

Evaluating Stability: We evaluate the stability of the three overlays with the help of three parameters; peer joining failure rate, parent/partner re-selection rate, and stream recovery duration. Fig. 3.4 presents the joining failure rate of peers in the three overlays. It can be seen that a larger fraction of peers experience joining failure in the multi-tree overlay. The joining failure rate of peers depends on the startup delay experienced by peers. The hybrid overlay has lowest joining failure rate because, startup is quick due to the selection of some stable tree peers as partners.

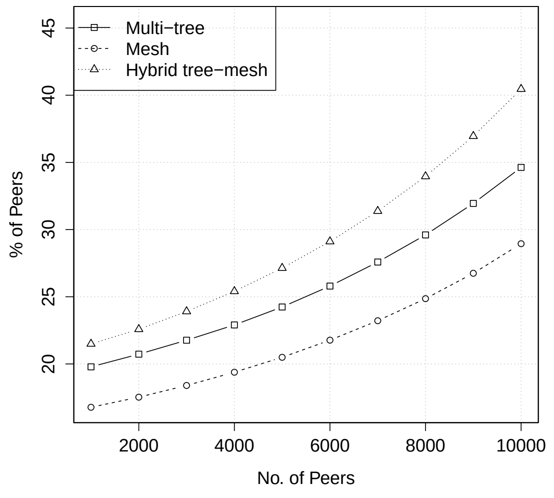
Fig. 3.5 compares the parent/partner re-selection rate for the three overlays. It can be observed that a larger fraction of peers experience parent re-selection in the hybrid overlay throughout the streaming session. However, it can be seen from Fig. 3.5b and Fig. 3.5c that most of the peers in the hybrid overlay change their parent after startup. This is because of two reasons. First, when mesh peers are flagged stable with the progress of streaming session, then they join backbone tree by replacing the mesh partners with tree parents. Second, peers regularly re-arrange themselves to maintain a minimum depth tree topology. In the hybrid overlay, parent re-selection takes place before startup either due to departure of parents or inadequate chunk rate delivered by parents. The possibility of either of these two events is low because each peer selects at least one stable peer as its parent.



(a) Total no. of peers experiencing parent/partner re-selection



(b) Parent/partner re-selection before startup



(c) Parent/partner re-selection after startup

Figure 3.5: Comparing percentage of peers experiencing parent/partner re-selection

On the contrary, a larger fraction of partners are re-selected before startup in mesh compared to the hybrid overlay due to random selection of partners and peer churn. The random selection of partners causes re-selection due to insufficient upload capacity and/or chunk delivery rate. The random selection also ignores the stability of peers. The mesh overlay has lower partner re-selection rate after startup because peers re-select their partners either when they find partners with better chunk availability or during peer churn. In the hybrid overlay, peers regularly update their parents irrespective of churn and this leads to frequent partner re-selection.

The multi-tree overlay has higher parent re-selection rate compared to mesh overlay, as shown in Fig. 3.5. This is because peers re-select their parents not only during churn, but also while adding new peers to the overlay. It is done to maintain the overlay design restrictions to ensure that each peer connects to the closest parent available to receive a sub-stream. In this process, the new peers replace existing peers, if required. Moreover, each peer becomes interior node in one of the trees and leaf node in others. The joining of new peers throughout the streaming session also results in parent re-selection of existing peers and increases the startup delay, if the peers are orphaned multiple times before playback, as shown in Fig. 3.1a.

We also evaluate the stream recovery duration of peers during peer churn, as shown in Fig. 3.6. It is observed that peers in the mesh and hybrid overlays are able to recover their streaming quality faster during churn compared to the multi-tree overlay. This is because orphaned peers in the multi-tree overlay traverse the tree of the lost sub-stream to find another parent with spare capacity, which takes time. In contrast to this, peers in the mesh and hybrid overlays recover their stream by finding another partner from its list of active peers. However, the departure of stable tree peers is handled differently in the hybrid overlay. When a tree peer departs, its children temporarily use their mesh partners to recover lost chunks till they find

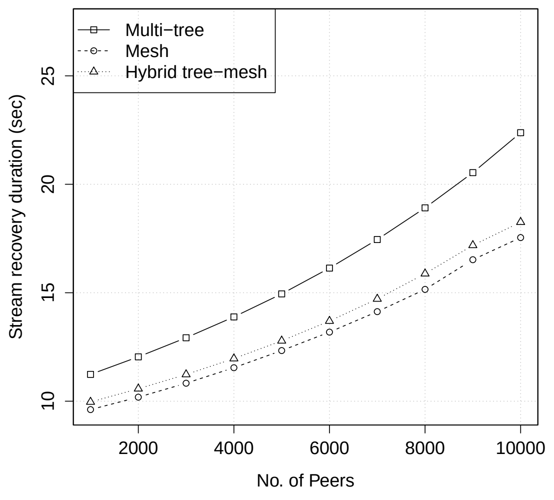


Figure 3.6: Comparing stream recovery duration of peers

another tree parent. An orphaned child finds a new parent with free upload capacity. Otherwise, it replaces a mesh peer attached to a tree peer. This time consuming procedure of finding a new tree parent causes peers to wait for longer time before they recover the desired streaming quality. Hence, peers in the hybrid overlay have longer stream recovery duration compared to mesh.

Evaluating Streaming Quality: The stability of the overlay during peer churn also affects the streaming quality perceived by peers. Fig. 3.7 presents the streaming quality experienced by peers in the three overlays. It is observed that mesh overlay has the highest streaming quality, while it is the lowest in multi-tree overlay. This is because, departure of peers in the multi-tree overlay degrades streaming quality of all the descendants till the orphaned peers select another parent. The peers in the multi-tree are also orphaned by their parents with the arrival of new peers to maintain the complex structure of the overlay. The frequent parent re-selections and longer parent re-selection time also lowers the streaming quality.

The peers in the mesh overlay experience better streaming quality because they

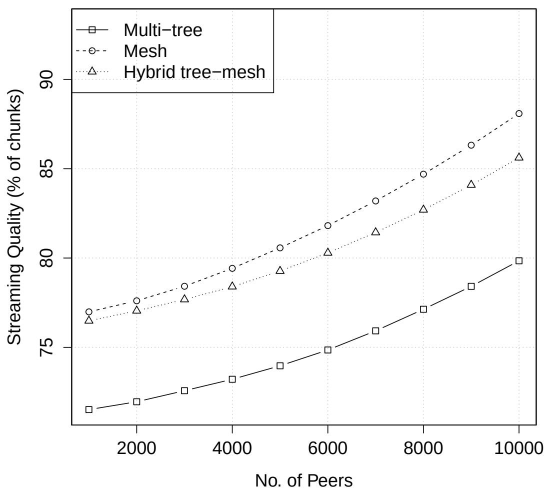


Figure 3.7: Comparing streaming quality experienced by peers

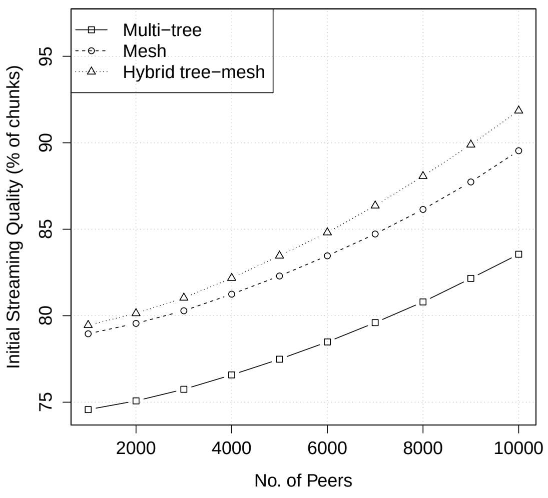


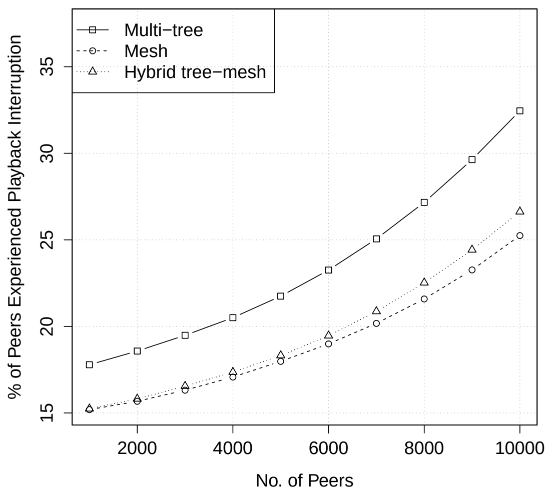
Figure 3.8: Comparing initial streaming quality experienced by peers

keep replacing their partners based on stream delivery efficiency or chunk availability. The lower partner re-selection rate and stream recover duration during churn also helps in maintaining the streaming quality. We observe that the hybrid overlay has slightly lower average streaming quality than the mesh due to the bottleneck bandwidth of tree links and departure of intermediate tree nodes. The peers in the hybrid overlay take longer time in re-selecting parents (as discussed earlier), which also degrades the streaming quality.

However, we found that the hybrid overlay has better initial streaming quality, as shown in Fig. 3.8. The initial streaming quality is calculated as the quality perceived by the peers in the first one minute of their session or during the startup. The hybrid overlay exhibits better initial streaming quality because, at least one stable tree peer is selected as a partner for new peers and peers experience lower parent re-selection rate before startup in the hybrid overlay.

We also assess the playback interruption experienced by peers, as reported in Fig.3.9. We identified the playback interruptions due to insufficient number of contiguous chunks in the buffer to continue the playback. Fig.3.9 compares the percentage of peers experiencing playback interruptions at least once in the session. It can be observed that this is larger for multi-tree overlay, while it is lowest for the mesh overlay. The reason for this is same as that mentioned while comparing streaming quality.

Fig. 3.10 compares the average duration of interruptions experienced by peers. It is observed that peers in the multi-tree overlay experience longer interruptions because peers take more time to re-select a parent. The mesh overlay has shorter interruptions because peers quickly select another partner from its active peers list. We also observed that a significant number of chunks were received after the deadline due to use of random paths in the mesh and hybrid overlay.



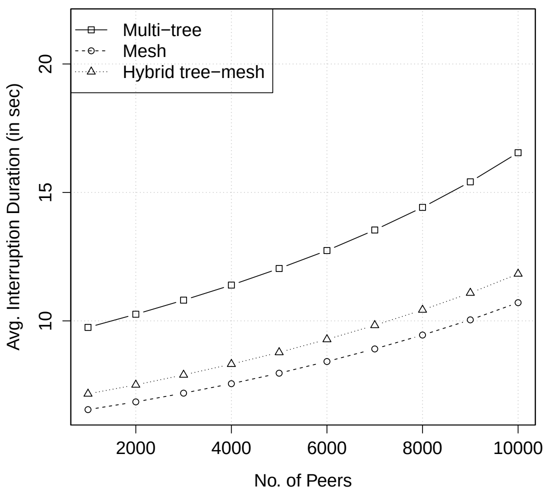
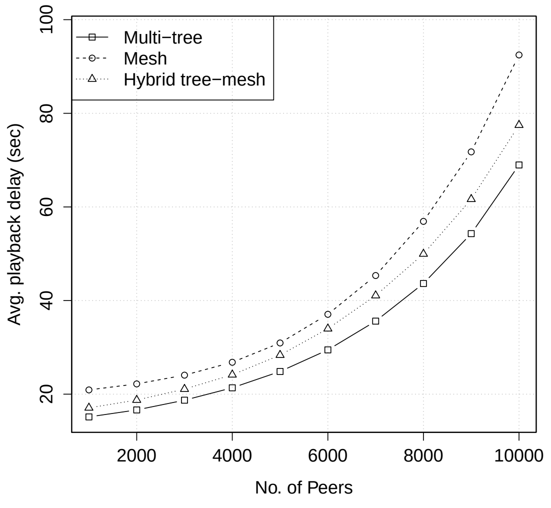
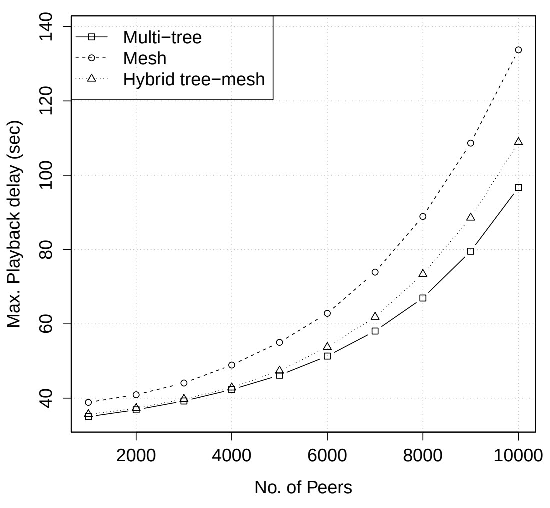


Figure 3.9: Comparing percentage of peers experiencing playback interruptions

Figure 3.10: Comparing duration of playback interruptions



(a) Average playback delay



(b) Maximum playback delay

Figure 3.11: Comparing playback delay perceived by the peers

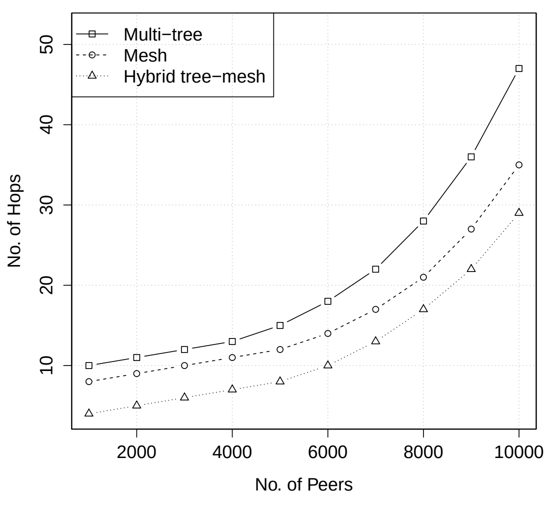


Figure 3.12: Comparing hopcount for different number of peers

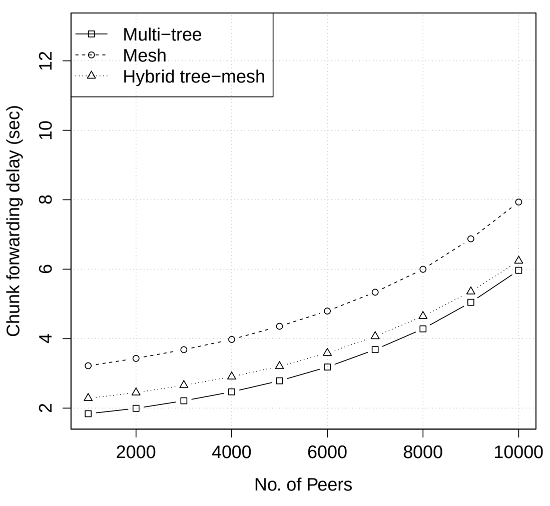


Figure 3.13: Comparing chunk buffering duration of peers at each hop

Evaluating Playback Delay: Fig. 3.11 shows the average and maximum playback delay experienced by peers in the three overlays. It is observed that the multi-tree overlay has the least playback delay, whereas mesh overlay exhibits the highest. The playback delay experienced by a peer depends on per-hop transmission delay, per-hop propagation delay, per-hop chunk buffering duration and the number of hops traversed by chunks from the source to the peer. In this evaluation, we assume propagation delay and transmission delay remain same in all the overlays. We report the chunk buffering duration and the hopcount in Fig. 3.12 and Fig. 3.13, respectively. It can be observed from Fig. 3.12 that the multi-tree overlay has larger hopcount compared to other overlays. This is due to skewed tree topology construction during peer churn. The mesh and hybrid overlays have lower average hopcount due to random selection of partners. The hopcount of the hybrid overlay is lowest because depth is optimized regularly.

Despite exhibiting higher hopcount, the multi-tree overlay results in lower playback delay compared to other overlays because per-hop chunk buffering duration is smaller, as shown in Fig. 3.13. The chunk buffering duration depends on the data delivery mechanism and parent/partner selection procedure used. In the multi-tree overlay, peers use push-based data delivery mechanism and forward chunks to their children as soon as they receive them. On the other hand, peers in the mesh and hybrid overlays use push-pull based data delivery mechanism. In the pull-based data delivery, chunks are stored for longer time before being forwarded. This results in larger chunk buffering duration for peers while using pull-based data delivery. In the mesh and the hybrid overlays, the chunk buffering duration experienced by peers depends on the push-pull ratio of chunks. We can see from Fig. 3.3 that push-pull ratio of chunks in the hybrid overlay is higher compared to the mesh overlay. Overall larger hopcount and per-hop chunk buffering duration in the mesh overlay leads to larger playback delay.

Overall, Table 3.4 summarizes the performance of the Multi-tree, Mesh and Hybrid tree-mesh overlays by providing a mean value along with the standard deviation for the various parameters that impact startup delay, stability, streaming quality and playback delay experienced by peers.

Table 3.4: Comparing performance of the Multi-tree, Mesh and Hybrid tree-mesh Overlays

Parameter	**Multi-tree** Mean (SD)	**Mesh** Mean (SD)	**Hybrid tree-mesh** Mean (SD)	**Reason**
Startup delay	**23.26 (6.7)**	22.41 (5.4)	21.65 (5.6)	**a)** Highest in multi-tree due to complex peer joining process, higher parent selection delay and higher parent re-selection rate before startup.
1) Parent/partner selection delay	**16.77 (5.3)**	13.28 (2.3)	13.79 (3.3)	
2) Buffering Delay	6.49 (1.4)	**9.12 (3.1)**	7.86 (2.4)	**b)** Higher buffering delay in mesh is due to chunks pulling, unavailability of chunks and free upload capacity at partners
3) Parent/partner re-selection rate (before startup)	**21.76 (3.7)**	14.64 (2)	13.86 (1.9)	
Stability				
1) Stream recovery duration	16.04 (3.5)	**13.10 (2.5)**	13.62 (2.6)	**a)** Mesh is most stable due to least parent/partner re-selection rate and stream recovery duration.
2) Total Parent/partner re-selection rate	30.64 (5.1)	**28.57 (4.3)**	34.09 (6.7)	**b)** In hybrid, stable peers regularly join backbone tree and change parents to maintain minimum depth tree
3) Parent/partner re-selection rate (After startup)	25.89 (4.7)	**21.81 (3.8)**	29.23 (6)	**c)** Joining of new peer might need to replace existing peer in multi-tree.
4) Joining failure rate	**21.69 (5.5)**	21.61 (3.8)	**19.13 (3.4)**	**d)** Joining failure highest in multi-tree due to higher startup delay

Continued on next page...

Table 3.4: Continued....

Parameter	**Multi-tree** Mean (SD)	**Mesh** Mean (SD)	**Hybrid tree-mesh** Mean (SD)	**Reason**
Streaming Quality	74.94 (2.6)	**81.71 (3.6)**	80.29 (2.9)	**a)** Highest in mesh due to faster stream recovery during peer departure. **b)** Lowest in multi-tree because disconnection/departure of a peer impacts quality of all its descendants. **c)** Frequent and longer parent re-selection duration also degrades quality in multi-tree.
Playback delay	32.86 (16.9)	**42.84 (22.7)**	37.35 (19.1)	**a)**Playback delay highest in mesh due to random partner selection, higher chunk buffering duration and lower push/pull ratio of chunks delivery
1) Chunk buffering duration	3.34 (1.3)	**4.96** (1.4)	3.73 (1.2)	
2) Push/pull ratio	100	**78.23 (8.8)**	87.41 (7.6)	
3) Hopcount	**21.2 (11.6)**	16.4 (8.3)	12.1 (7.8)	**b)**Highest hopcount in multi-tree due to skewed topology, but lowest per hop delay.

3.4 Summary

In this chapter, we explored the limitations associated with the construction and management of different P2P overlay topologies, using a trace-based simulation study. We studied the impact of peer churn and heterogeneous bandwidth of peers on QoS and stability of multi-tree, mesh and hybrid tree-mesh overlays.

We found that the multi-tree overlay shows better playback delay, whereas the mesh overlay exhibits better stability and streaming quality. The hybrid overlay results in least startup delay but, it is not significantly lower than that in mesh overlay. The peers in the mesh and hybrid tree-mesh overlay experience startup delay due to unavailability of chunks and upload capacity at selected peers. The multi-tree overlay shows highest startup delay due to higher peer selection time.

The hybrid overlay also lacks in taking advantage of lower playback delay exhibited by tree topology. The playback delay of the hybrid overlay is significantly higher than that in the multi-tree overlay because, a larger fraction of peers are part of mesh topology. To reduce the playback delay in the mesh part of the hybrid overlay, the number of hops traversed by chunks as well as per-hop latency need to be reduced. Our study also revealed that the hybrid overlay exhibits higher stream recovery duration and lower streaming quality compared to mesh overlay, despite predicting peers stability and using stable peers to form backbone tree in the overlay. However, stability based parent selection in the hybrid topology helps in providing a better streaming quality to peers during startup. Overall, it is observed that a precise stability prediction mechanism and a parent/partner selection strategy that considers peer heterogeneity can improve startup delay, playback delay and streaming quality of peers in mesh and hybrid tree-mesh overlays.

The conclusions from this study motivates us to design a peer selection strategy to minimize playback and startup delay of the mesh topology. In the next chapter we propose a three-stage peer selection strategy to reduce delay in the mesh overlay.

Chapter 4

A Three-stage Peer Selection Strategy for Mesh Overlays

4.1 Introduction

Peer selection is one of the fundamental elements to build an overlay that affects the QoS experienced by peers. In this regard, various peer selection strategies are proposed in the literature for mesh overlays. In the existing strategies peers are selected at two different stages of overlay construction. First, the tracker suggests a few existing peers to the new peer (referred tracker-tier selection). Then the new peer selects its partners out of those suggested by the tracker (referred peer-tier selection). In most of the existing systems peers are selected randomly at the tracker-tier [51–53] whereas, in [60] the tracker selects the peers based on the upload capacity. However at the peer-tier, peers are selected considering parameters like upload capacity and propagation delay. On the other hand, the existing peer selection strategies neglect chunk buffering duration at each hop and buffering level of the peer, which have significant impact on playback delay and startup delay. Here, the chunk buffering duration is the duration for which a chunk is stored at a peer before being forwarded and the buffering level indicates the number of contiguous chunks available in the buffer.

In this chapter, we propose a three-stage peer selection strategy to reduce the playback delay and startup delay experienced by the peers in mesh overlays. The proposed strategy uses peer selection in three stages of overlay construction. In the first stage (tracker-tier), the tracker selects prospective partners considering upload capacity and buffering level of the peers. At the second stage (peer-tier), a peer selects its partners out of those suggested by the tracker based on upload capacity, propagation delay, chunk buffering duration and buffering level of the partners. The last stage is the topology adaptation stage, where the peers replace their ancestors considering the same parameters to maintain a minimum delay overlay during churn. We compare the performance of the proposed strategy with Fast-Mesh [51] and HLPSP [60], both with and without peer churn. Results show that the proposed strategy reduces playback delay significantly and startup delay marginally.

Organization of the Chapter: The rest of this chapter is organized as follows. Section 4.2 describes the system model assumed and the proposed peer selection strategy. Results from the evaluation of the proposed peer selection strategy are discussed in Section 4.3. Section 4.4 concludes the work.

4.2 Proposed Peer Selection Strategy

In this section we describe the basic system model, the assumptions used, and the proposed peer selection strategy.

4.2.1 System Model

The mesh overlay in this work is similar to that used in Fast-Mesh, which we briefly describe here for a quick reference [51]. Fig. 4.1 illustrates the system, consisting

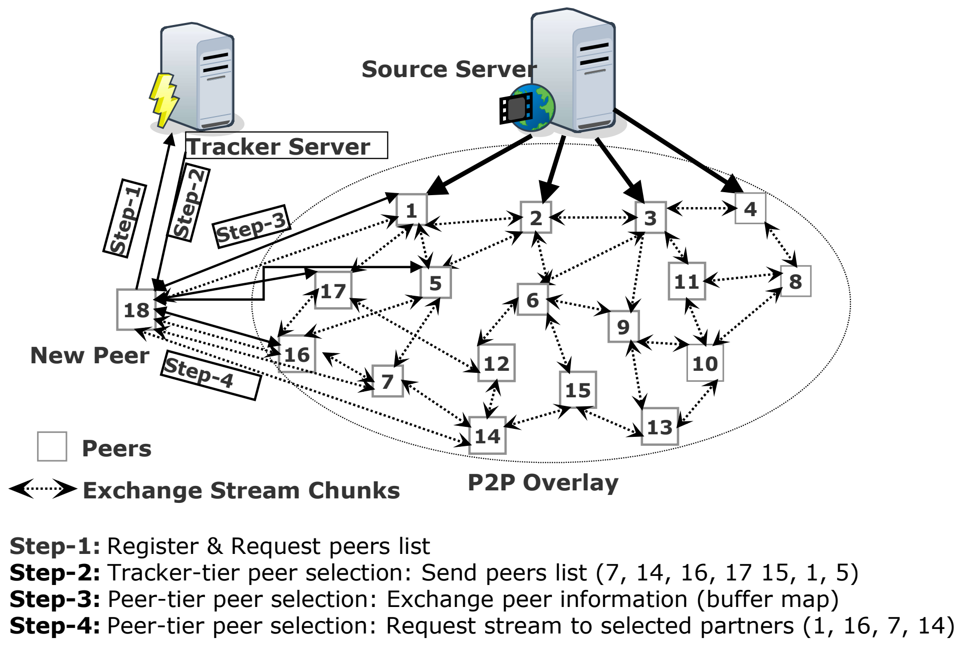

Source Server
Tracker Server
Step-1
Step-2
Step-3
Step-4
New Peer
18
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
Peers
Exchange Stream Chunks
P2P Overlay
Step-1: Register & Request peers list
Step-2: Tracker-tier peer selection: Send peers list (7, 14, 16, 17 15, 1, 5)
Step-3: Peer-tier peer selection: Exchange peer information (buffer map)
Step-4: Peer-tier peer selection: Request stream to selected partners (1, 16, 7, 14)

Figure 4.1: A mesh based peer-assisted live streaming system with tracker

of a source server, a tracker and N peers which join the streaming session of length l. The server divides the video content into small size blocks called chunks. The chunks are generated by the server at the rate of C chunks per unit time. The server distributes the chunks randomly to the peers connected to it. The tracker maintains information like peer identifier and upload capacity for all the peers. When a new peer joins the system the tracker suggests few prospective partners to the new peer with the help of a peer selection strategy (tracker-tier). After receiving a list of prospective partners from the tracker, a new peer selects its initial partners using another peer selection strategy (peer-tier). Once a peer starts receiving the stream, it performs a topology adaptation process to reposition itself in the overlay such that it perceives least playback delay.

Assumptions and Definitions: We assume that peers have heterogeneous upload and download capacity. Peers can also announce zero upload capacity. The partners which provide chunks to a peer are termed active partners, and those which download chunks are termed passive partners. Let U_i^a be the announced (or aggregate) upload capacity and U_i^r be the residual upload capacityt of the i^{th} peer. Let $D_{n,i}$ be the propagation delay from peer i to the new peer n and B_i be the buffering level of peer i, respectively. The propagation delay of each hop is not negligible and buffering level of a selected peer can be zero. All these assumptions induce a non zero latency between any two peers or between the source and a peer.

4.2.2 Proposed Three-stage Peer Selection Strategy

In the proposed peer selection strategy, a peer selects its partners at three different stages of the overlay construction. Stage 1: Tracker-tier selection strategy, Stage 2: Peer-tier selection strategy, and Stage 3: Topology adaptation strategy. At Stage 1 the tracker performs prioritized selection of high upload capacity peers and the peers with data availability. At Stage 2 the new peer selects the peers based on their upload capacity, propagation delay, lag perceived by the peer and the buffering level. The first two stages help a peer to obtain an initial position in the overlay and start its video playback. The third stage helps in repositioning a peer in the overlay to reduce delay from the source. The detailed process of joining new peer in the overlay and partner selection at every stage is illustrated in Fig. 4.2 and discussed further.

Stage 1: Tracker-tier Selection Strategy: The Stage 1 peer selection strategy aims to keep new peers closer to the source and ensure data availability in the buffers of the selected partners. To achieve these objectives the tracker prefers to suggest peers with free residual upload capacity, high aggregate upload capacity and non-zero buffering level. The selection of high upload capacity peers helps in reducing length of the data dissemination path and also helps in reducing transmission delay at each hop. The initial buffering level helps in reducing delay due to freezes and

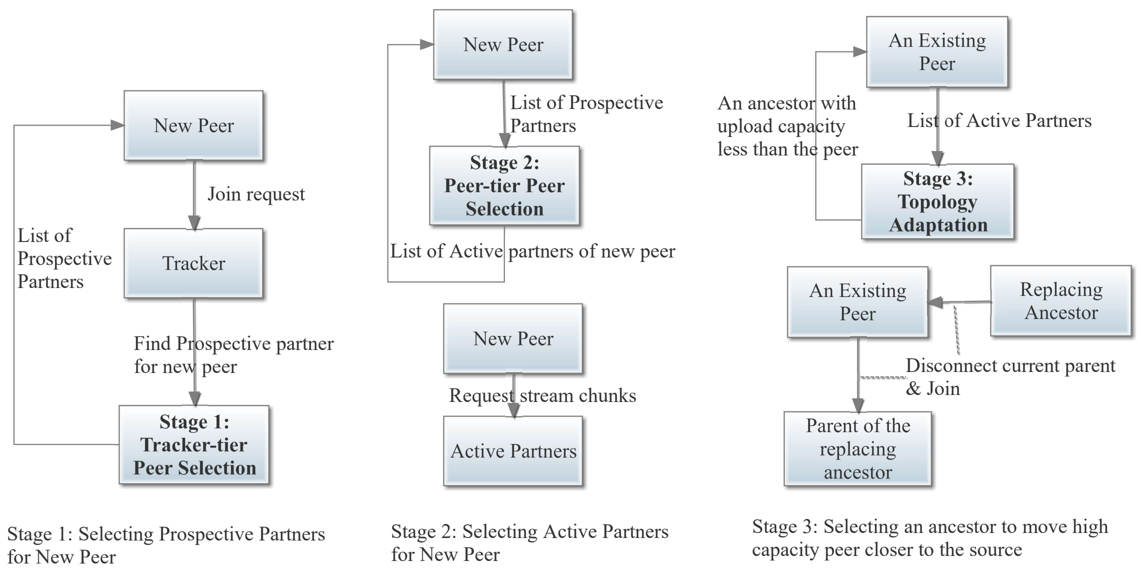

New Peer
Join request
Tracker
List of Prospective Partners
Find Prospective partner for new peer
Stage 1: Tracker-tier Peer Selection
Stage 1: Selecting Prospective Partners for New Peer
New Peer
List of Prospective Partners
Stage 2: Peer-tier Peer Selection
List of Active partners of new peer
New Peer
Request stream chunks
Active Partners
Stage 2: Selecting Active Partners for New Peer
An Existing Peer
An ancestor with upload capacity less than the peer
List of Active Partners
Stage 3: Topology Adaptation
An Existing Peer
Replacing Ancestor
Disconnect current parent & Join
Parent of the replacing ancestor
Stage 3: Selecting an ancestor to move high capacity peer closer to the source

Figure 4.2: Proposed Three-stage Peer Selection Strategy

insufficient chunk upload rate. This peer selection strategy of the tracker also helps in reducing startup delay. By suggesting peers with free upload capacity, the time spent by a new peer in the search of a partner is reduced.

When a new peer n joins the system, the tracker suggests K peers as prospective partners which have already joined the system. To select these K peers, the tracker maintains two lists of peers: High-priority Partners List (HPL) and Low-priority Partners List (LPL). The peers in both the lists are arranged in the decreasing order of their aggregate upload capacity. Therefore, during prospective partner selection the tracker only fetches the first K peers from the lists. HPL and LPL help the tracker in quick selection of highest upload capacity peers each time a new peer joins. The tracker keeps only the peers with free residual upload capacity in both the lists. However, the HPL contains those peers which also have data in their buffers while, the LPL might contain peers with empty buffers. Therefore, the tracker gives preference to peers in the HPL. The peers are also selected from the LPL because the peers with empty buffers will receive data subsequently. The detailed peer selection strategy at the tracker-tier is given in **Algorithm 1**.

Algorithm 1: Stage 1 Peer Selection Algorithm

```
Input: $HPL \leftarrow$ High-priority Partners List, $|HPL| \leftarrow$ The number of
       peers available in the HPL, $LPL \leftarrow$ Low-priority Partners List
Output: $P \leftarrow$ Set of '$K$' prospective partners selected by the tracker for a
        new peer '$n$'
if $|HPL| \geq$ '$K$' then
    Select first '$K$' peers from the HPL and add them into $P$.
else
    Select all the peers of HPL and also select first $K - |HPL|$ peers from
     LPL.
    Add selected peers into $P$.
```

Maintenance of HPL and LPL: The HPL and the LPL lists are maintained by the trackers with the help of four messages sent by the peers to the tracker. The details of the messages sent by the peers and the actions taken by the tracker are discussed below.

`Message 1 – Publish aggregate upload capacity` : When a new peer sends join request to the tracker it also informs the tracker about its aggregate upload capacity. If aggregate upload capacity of the new peer is greater than zero, then the tracker adds the new peer to LPL. This ensures only those peers who are able to upload data are suggested as prospective partners.

`Message 2 – Buffering level greater than zero` : When a peer receives its first chunk it sends a message to the tracker to inform that its buffering level has become greater than zero. The tracker then deletes the peer from LPL and adds it to HPL. This helps in suggesting partners which have data in their buffers.

`Message 3 – Upload capacity exhausted` : When peer has acquired enough passive partners to completely utilize its upload capacity it informs the tracker. The tracker then deletes the peer from LPL and HPL. With the help of this message tracker avoid suggesting partners which do not have free upload capacity.

`Message 4 – Aggregate upload capacity regained` : A peer sends this message to the tracker when all the passive partners of the peer have left and peer regains its aggregate upload capacity. After receiving this message the tracker checks peer's buffering level. If buffering level of the peer is greater than zero then it adds the peer to the HPL, otherwise it adds the peer to the LPL. This ensures selection of peers which have data in their buffers along with the free upload capacity.

Message complexity: Each peer sends `Message 1` and `Message 2` to the tracker only when it joins the system and when it receives its first chunk, respectively. However, `Message 3` and `Message 4` can be sent by the peers more than once, in the respective order. This happens when a peer connects to the maximum possible passive partners and it does not have free upload capacity. Later the same peer may regain its full upload capacity when all the passive partners of the peer have left. Therefore, the number of messages to maintain LPL and HPL is in $O(N)$.

Stage 2: Peer-tier Selection Strategy: In Stage 2 a new peer selects a set of active partners to find an initial position in the overlay. The selection is designed such that the new peer receives video with minimum playback delay and startup delay.

In this stage, the prospective partners suggested by the tracker are at first ranked based on few parameters and then top ranked peers are selected by the new peer as active partners. The parameters considered for ranking partners are: upload capacity of the partner, propagation delay from the partner, lag experienced by the partner and the buffering level of the partner. The playback delay of the new peer is estimated as the sum of the lag perceived by the partner and the lag caused on the last hop delay. The startup delay is estimated with the help of last hop delay. The partners which are estimated to provide lower delay are ranked higher. The ranking helps to estimate the playback delay and startup delay for a new peer through each prospective partner. Next, we explain the algorithm for the Stage 2 peer selection strategy.

Let P be the set of peers suggested by the tracker as prospective partners for the new peer n. Further, let U_i^r, B_i and L_i be the residual upload capacity, buffering level and lag experienced by a peer i, respectively where $i \in P$. Let $D_{n,i}$ be the propagation delay from peer i to the new peer n, where $i \in P$. The tracker receives the join request from the new peer at time t_1 and a peer receives buffer map request from the new peer at time t_2. We assume that the source informs the tracker about the chunk number every time its generates a new chunk. While suggesting prospective partners to the new peer, the tracker also informs the new peer about the chunk number of the latest chunk generated by the source. Let $C_s(t_1)$ be the chunk number of the latest chunk generated by the source by t_1 and $C_i(t_2)$ be the latest chunk received by the peer i by t_2.

The procedure to rank peers among those suggested by the tracker is as follows: First, the new peer requests all the peers in the set P to provide residual upload capacity and their buffer maps. The new peer also calculates the propagation delay from each peer i in the set P using measured round trip time. With the help of buffer map, the new peer also calculates the buffering level and lag for each peer i in the set P. The buffering level of a peer is calculated as the percentage of buffer filled with the maximum number of contiguous chunks. We assume that each peer has buffer of equal size. As shown in Eq. 4.1, the lag experienced by a peer with respect to source is calculated as

$$L_i = \frac{C_s(t_1) - C_i(t_2)}{C}, \forall i \in P \tag{4.1}$$

The rank (R_i) of each peer in P is calculated as normalized weighted sum of U_i^r, $D_{n,i}$, L_i and B_i as shown in Eq. 4.2.

$$R_i = \alpha \times \left(\frac{U_i^r}{\max\limits_{\forall j \in P}(U_j)} + \frac{\min\limits_{\forall j \in P}(D_{n,j})}{D_{n,i}} + \frac{\min\limits_{\forall j \in P}(L_j)}{L_i} \right) + (1-\alpha) \times \frac{B_i}{\max\limits_{\forall j \in P}(B_j)} \tag{4.2}$$

Algorithm 2: Stage 2 Peer Selection Algorithm

```
Input: $P \leftarrow$ Set of '$K$' preferred partners suggested by the tracker to the
       new peer '$n$'
Output: Active partners of the new peer '$n$'
Calculate '$R_i$' for each peer '$i$' in set $P$.
Select peers with highest value of '$R_i$' as active partners.
```

As shown in *Algorithm 2*, the new peer selects peers with highest R_i as its active partners greedily until it receives chunks with the desired streaming rate. In Eq. 4.2, to add the parameters of different units they are first normalized to transform the values into dimensionless numbers. This also helps in understanding the relative contribution of the parameters in the peer selection process. The value of α is set after tuning with the help of experiments, the results of which are described in Sec. 4.3.1 of this chapter.

The rationale behind the selection of partners based on these four parameters is as follows. The latency of a path between two peers in a peer-assisted system is composed of four major components: per-hop transmission delay, per-hop propagation delay, per-hop chunk buffering duration and per-hop chunk upload rate. Therefore, to estimate the playback delay of the new peer through prospective partners the latency calculation is divided into two parts. In the first part the latency of the path from the source to the prospective partners is estimated with the help of lag perceived by the partners. In the second part the latency of the last hop *i.e.*, the path from the prospective partner to the new peer is estimated.

The chunk buffering duration is the time for which a chunk is buffered at a peer before it is forwarded to another peer. The chunk buffering duration up to the last hop is calculated using lag. Therefore lag ensures to select partners which have most recent chunks or the chunks which have been buffered for smaller time. To estimate propagation delay of last hop, approximate round trip time from the prospective partner to the new peer is calculated. The transmission delay of last hop is estimated with the help of residual upload capacity of the prospective partner. It is not only upload capacity of peer which ensures a desired chunk download rate to a peer but also the availability of chunks at the sender's buffer. Therefore, to calculate chunk upload rate of a prospective partner its buffering level is assessed. The buffering level of a peer not only indicates its chunk upload rate, but it also represents by what rate it is receiving chunks. Considering this also ensures selection of peers with faster filling of buffer with contiguous chunks and hence avoids lag due to playback freezes.

The selection of partners based on last hop latency not only ensures least playback delay, but also helps in reducing startup delay. This is because after the selection of partners the startup delay experienced by a peer depends on the delay in delivering chunks by its partners. Next, we describe the third stage of constructing overlay where peers use topology adaptation strategy to move closer to the source and to handle churn.

Stage 3: Topology Adaptation Strategy: In the proposed topology adaptation strategy, the peers reposition themselves once they start video playback. To reposition itself, a peer first finds replaceable ancestor(s). The replaceable ancestor(s) of the peers are the ancestor(s) which have lesser aggregate upload capacity than the peer. Then the peer ranks all the replaceable ancestors based on various parameters like the aggregate upload capacity, propagation delay, relative lag and the buffering level of the ancestors. The peer then replaces the top ranked ancestor. A peer iteratively moves closer to the source by replacing its ancestors until it does not find any ancestor which can be replaced. Next, we describe the algorithm for the topology adaptation.

Let T be the set of ancestors of i^{th} peer which have lesser aggregate upload capacity U_j^a, where $j \in T$. H_j is the highest chunk number available at an ancestor j. Rank R_j for all the ancestors $j \in T$ is calculated with the help of Eq. 4.3.

$$R_j = \alpha \times \left(\frac{H_j}{\max\limits_{\forall l \in T}(H_l)} + \frac{U_j^a}{\max\limits_{\forall l \in T}(U_l^a)} + \frac{\min\limits_{\forall l \in T}(D_{i,l})}{D_{i,j}} \right) + (1-\alpha) \times \frac{B_j}{\max\limits_{\forall l \in T}(B_l)} \tag{4.3}$$

The calculation of R_j considers four parameters: aggregate upload capacity, propagation delay, highest chunk number and buffering level. The selection of an ancestor which has highest aggregate upload capacity minimizes the difference between the aggregate upload capacity of the peer and that of the selected ancestor. This helps in keeping the higher upload capacity peers closer to source. The propagation delay helps replaced ancestor in maintaining least delay after repositioning. With the help of highest chunk number, the relative lag of the ancestors is calculated. The ancestor with the highest chunk number available in its buffer is assumed to suffer from least lag. Further, the buffering level of the ancestor indicates the chunk receiving rate of the ancestor and hence chunk delivery efficiency of the ancestor's parent. Hence, along with repositioning peers closer to the source this topology adaptation strategy also maintains minimum playback delay for the peers by selecting better partner.

Algorithm 3: Stage 3 Topology Adaptation Algorithm

```
Input: Set of active partners of a peer ‘$i$’
Output: Replaceable peer ‘$j$’
if $U_i^r \geq C$ then
    Find ancestor(s) peer ‘$j$’ such that $U_j^a < U_i^a$.
    Add $j$ in the set ‘$T$’
    if $T \neq \phi$ then
        for each peer $j \in T$ do
            Calculate $R_j$
        Return ‘$j$’ with the highest value of $R_j$ .
    else
        Return ‘$i$’
else
    Return ‘$i$’
```

As shown in *Algorithm 3*, the topology adaptation process consists of three major steps:

1. A peer first checks whether its residual upload capacity is greater than or equal to streaming rate. If yes, then it searches for those ancestor(s) whose aggregate upload capacity is less than that of the peer, otherwise it quits adaptation. To find such ancestors the peer sends a request message to its ancestors which consists of two fields, a) the peer's aggregate upload capacity, and b) the time-to-live (TTL) field.

2. Each time an ancestor receives a request message it decrements the TTL by 1 before forwarding it. The request message is forwarded by the ancestors in the

upstream until the TTL becomes zero. When an ancestor receives a request message it compares its aggregate upload capacity with the peer's aggregate upload capacity. If aggregate upload capacity of the ancestor is less than that of the peer then the ancestor responds with a reply message. The reply message consists of ancestor's buffer map and its aggregate upload capacity.

3. After receiving the reply message from the ancestor(s), the peer calculates the rank of each ancestor with the help of Eq. 4.3. Then, it selects the ancestor with the top rank for replacement. To reposition the peer at the position of the selected ancestor, the parents of the selected ancestor becomes the parents of the peer and in turn the peer becomes the parent of the selected ancestor.

4.2.3 Computational Complexity

In the proposed three-stage peer selection strategy, first the tracker selects K prospective partners for each new peer using Algorithm 1. It takes constant time ($O(1)$) to select prospective partners from sorted lists of peers, i.e. HPL and LPL. However, it takes exchange of $O(N)$ messages between peers and the tracker to maintain these lists, as discussed before in Section 4.2.2. Next, the new peer selects its partners out of K prospective partners using Stage-2 peer selection algorithm which takes $O(KlogK)$ time. Further, the peers also perform topology adaptation using Algorithm 3 to reposition higher upload capacity peers closer to the source in the overlay. If a peer is connected to M number of parents to receive video in the form of M sub-streams, then each peer takes $O(M)$ unit of time to find an ancestor to replace and reposition itself in the overlay. Overall, when N peers join the overlay it takes $N * (M + KlogK)$ units of time.

Here, $M < K << N$, i.e. the number of prospective partners (K) and number of parents of a peer (M) are very less as compared to number of peers (N) joining the overlay. Therefore, it takes $O(N)$ time for N number of peers to join the overlay.

4.3 Performance Evaluation

The performance of the proposed peer selection strategy is evaluated by comparing it with the best known alternate solutions, Fast-Mesh [51] and HLPSP [60]. The OMNeT++ simulator [97] with its Oversim framework [98] and INET framework [99] are used to simulate the peer selection strategies. The primary parameters used for the performance evaluation are playback delay and startup delay perceived by the peers. The performance is evaluated both with and without peer churn.

4.3.1 Evaluation Scenarios

The simulation parameters are selected considering various experimental and trace-based studies of live streaming systems. The justification for parameters considered is as follows. We used a streaming rate of 320 Kbps in accordance with the study reported in [21] showed that PPLive supported rates ranging from 250 Kbps to 400 Kbps. The upload capacity distribution of peers is set similar to that in [51], given in Table 4.1. This distribution follows a bandwidth measurement study of a real-world streaming event [100], where the data was collected from servers of Akamai over a 3-month period from October 2003 to January 2004.

Table 4.1: Upload Capacity Distribution of Peers

Percentage of peers	**Upload Capacity**
30%	0 Kbps
58%	16 Kbps - 160 Kbps
5%	320 Kbps - 1.4 Mbps
7%	1.6 Mbps

The value of TTL is set to 3 in the topology adaptation algorithm of proposed strategy and the adaptation process of the Fast-Mesh [51]. This is because both

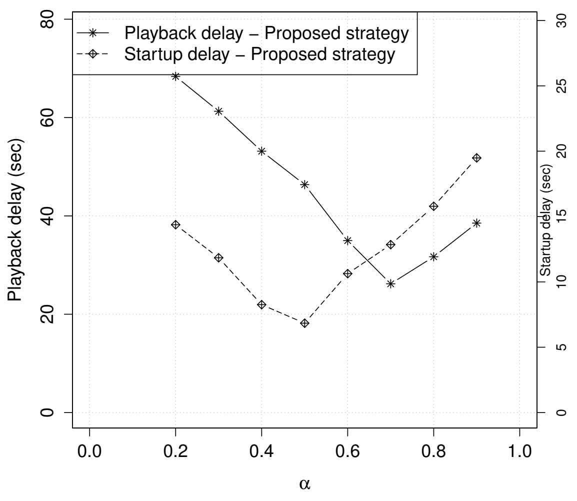


Figure 4.3: Impact of values of α on playback delay and startup delay

strategies follow a similar adaptation process and the results of Fast-Mesh show that lower TTL values result in significant reduction in playback delay. However, higher TTL values do not show significant improvement in the delay. We varied α from 0.2 to 0.8 and observed its impact on the average playback delay and start up delay perceived by the peers, as shown in Fig. 4.3. It is found that playback delay is minimum when $\alpha = 0.7$, whereas startup delay is minimum when $\alpha = 0.5$. This is because when peers were selected with $\alpha = 0.7$ the parameters like transmission delay, propagation delay and chunk buffering duration were given higher priority over delay due to non-contiguous chunks. We set $\alpha = 0.7$ for all the simulations.

Peer Churn Setting: The scenario of no peer churn is simulated with the help of a NoChurn configuration available in OMNeT++. In this configuration peers are added in the overlay until target number of peers is reached and peers do not leave the streaming session until simulation ends. To simulate the scenario of peer churn we use LifeTimeChurn configuration of the OMNeT++. In the LifeTimeChurn configuration peers lifetime or session duration is randomly set using

Pareto distribution. This is because, authors of [56, 102] showed that peers' lifetime follows a heavy-tailed distribution such as Pareto and Weibull distributions. It represents that the peers' age is an indicator of their expected remaining time in the system. The peers arrive following a Poisson process where peers inter-arrival time is modeled using exponential distribution which is also used in [104–106]. To simulate flash crowd, peer arrival rate is set high during the initial 5% time of the streaming session because, similar trend was observed in measurement studies [32, 100].

4.3.2 Evaluation Metrics

Our work primarily focuses on improving playback delay and startup delay. However, while building a low-delay system, it is also important to ensure that other parameters like streaming quality and stability of the system are not compromised during churn. Therefore, we also evaluate these parameters. The streaming quality is studied with the help of buffering level of peers. The stability of the system is measured with the help of stabilization duration after flash crowd and partner recovery time during peer churn.

The primary metrics used in the evaluation are defined as follows:

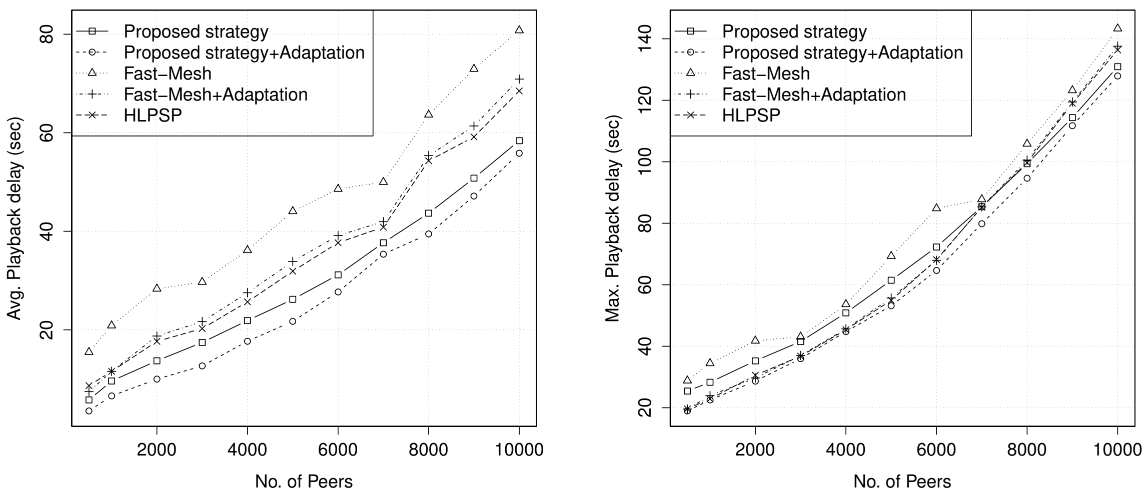
- **Playback delay:** It is measured as the difference between the time when a video frame is generated by the source to the time when it is delivered to the peer.

- **Startup delay:** The startup delay of a peer is measured as the waiting time experienced by the peer to start watching the stream after it joins the streaming session.

  The startup delay is composed of three components: a) Stage 1 delay: Delay incurred due to peer selection process at the tracker, b) Stage 2 delay: Delay incurred due to peer selection process at the peers, c) Buffering delay: Delay in receiving data after peer selection process is completed.

4.3.3 Results and Discussions

First we present the performance of our proposed peer selection strategy under no churn scenario and then under peer churn scenario.



(a) Average Playback Delay (b) Maximum Playback Delay

Figure 4.4: Variation in the playback delay with the number of peers in the system

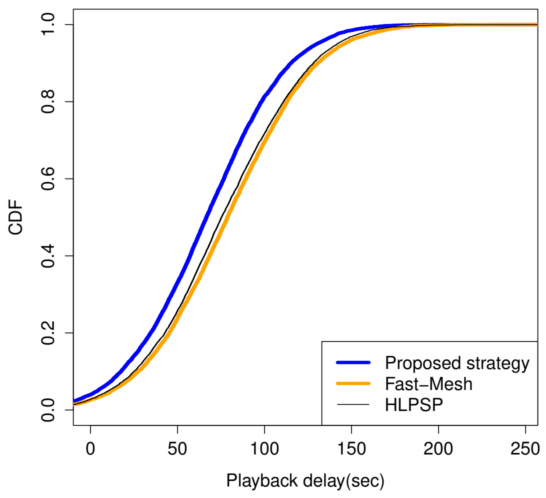


Figure 4.5: CDF of playback delay with an average hop count of 50

Performance Evaluation without Peer Churn: Fig. 4.4 shows the playback delay experienced by the peers for different number of peers in the system. Fig. 4.4a and Fig. 4.4b show the impact of number of peers on the average and maximum playback delay perceived by peers in Fast-Mesh, HLPSP and the proposed strategy, respectively. It can be observed that the playback delay increases with the increase in number of peers. This is because with the increase in number of peers the length of the path from the source to the peers increases. Each intermediate hop accounts for the delay due to transmission, propagation and chunk buffering.

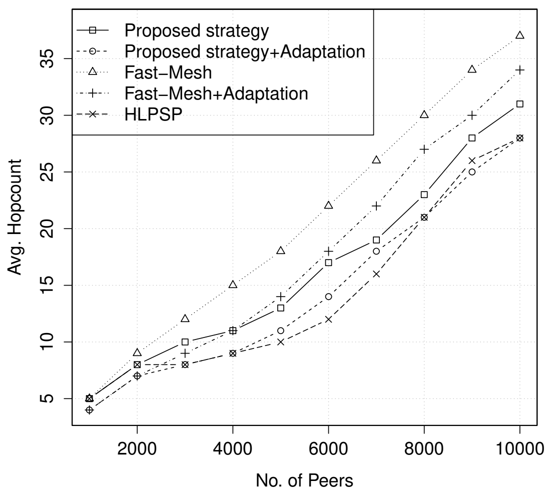
Fig. 4.4a and Fig. 4.4b also compare the playback delay perceived by peers in Fast-Mesh, HLPSP and the proposed strategy with and without topology adaptation. Topology adaptation is not performed with HLPSP because the tracker already arranges all the joining peers closer to the source in the decreasing order of their upload capacity. It is observed that the proposed strategy exhibits lower playback delay than Fast-Mesh and HLPSP in all conditions.

Our strategy outperforms Fast-Mesh because, the tracker in Fast-Mesh randomly selects peers. In HLPSP the tracker and the peers considers only upload capacity while arranging peers and selecting partners. However, in the proposed strategy the tracker selects peers based on upload capacity and data availability in Stage 1 and uses chunk buffering duration and buffering level additionally for the partner selection in Stage 2. To substantiate this claim we compare the playback delay for all the three strategies in Fig. 4.5 for peers with same number of intermediate nodes to the source. We see that the playback delay in our case is lower. This shows that the per-hop delay is lower in the proposed strategy. However, Fast-Mesh and our strategy both perform peer selection considering upload capacity and propagation delay at the peer-tier. Therefore, the reduction in playback delay is mainly due to the additional factors and selection of partners with more upload capacity at each hop. In comparison with HLPSP, we conclude that playback delay cannot be reduced by only considering the upload capacity.

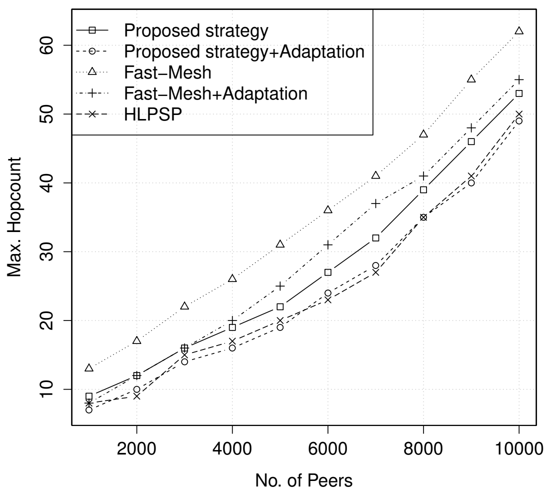
We have also calculated the contribution of various parameters (lag, upload capacity, propagation delay and buffering level) in the selection of peers at Stage 2 when system exhibits least average playback delay in the proposed strategy. It is found that 85% peers are selected based on smallest lag value, whereas, only 10% peers are selected with highest buffering level. However, it is also observed that more than one parameter can also dominate other parameters during a selection.

Similar to the playback delay, hop count also increases with the increase in number of peers as shown in Fig. 4.6a and Fig. 4.6b. However, the proposed strategy results in lower hop count compared to Fast-Mesh due to the selection of higher upload capacity peers at both the stages. This reduces the length of the data dissemination path traversed by the chunks and keeps peers closer to the source. Contrarily, HLPSP and our strategy show similar trend in hop count because higher upload capacity peers are arranged closer to the source in HLPSP. However, larger delay in HLPSP also indicates a higher per-hop delay in HLPSP.

With the help of Figs. 4.4a, 4.4b, 4.6a and 4.6b the contribution of adaptation process in the reduction of playback delay and hop count can also be observed.



(a) Average hop count

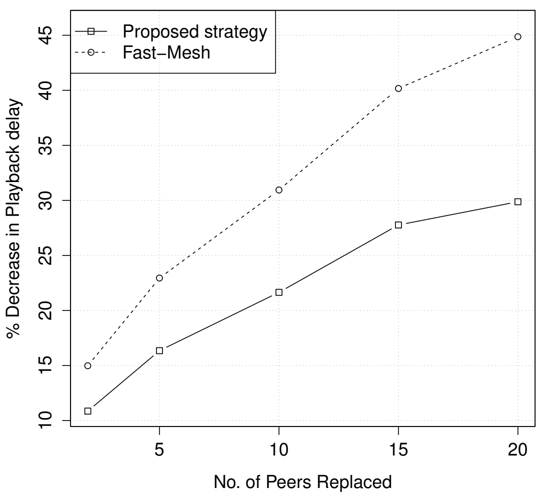


(b) Maximum hop count

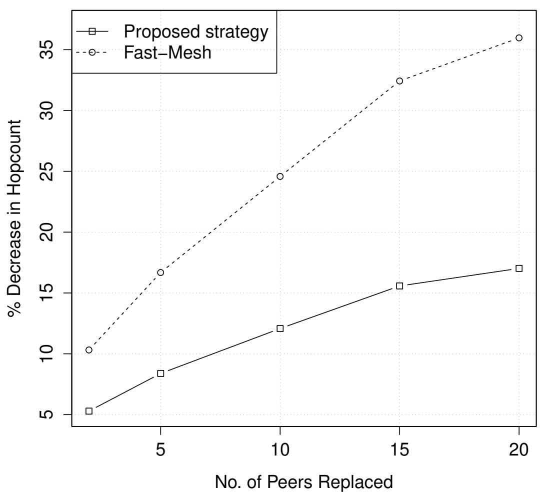
Figure 4.6: Comparing hop count for different number of peers

During adaptation process peers replace their parent or ancestor to reposition peers closer to the source in the overlay which further impacts playback delay and hop count of the peers. It is observed that the average playback delay of proposed strategy is reduced by 15-18% with the help of our adaptation, whereas Fast-Mesh shows significant reduction of approximately 25% using our adaptation. Similarly, the average hop count of proposed strategy is also reduced by 10-12% with the help of our adaptation, whereas Fast-Mesh shows significant reduction of 15-18% using our adaptation. This is because the random peer selection at tracker-tier in Fast-Mesh resulted in a larger number of parents with lower upload capacity. Hence it provides a scope for higher percentage of reduction.

To substantiate this inference Fig. 4.7a and Fig. 4.7b show the average reduction in playback delay and hop count when peers perform replacement. It can be observed that in Fast-Mesh, the reduction in playback delay and hop count is higher for same number of ancestor replacements as compared to proposed strategy



(a) Playback Delay

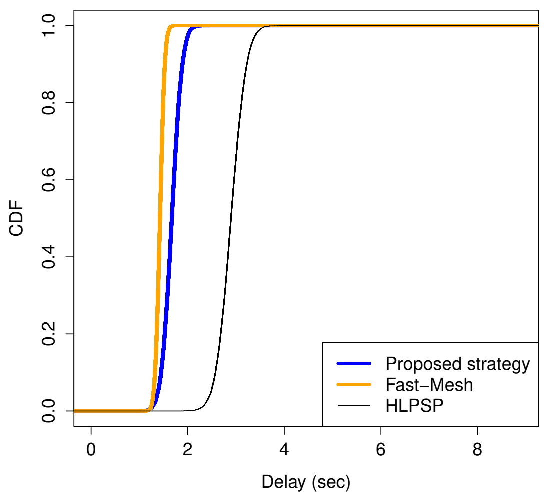


(b) Hop Count

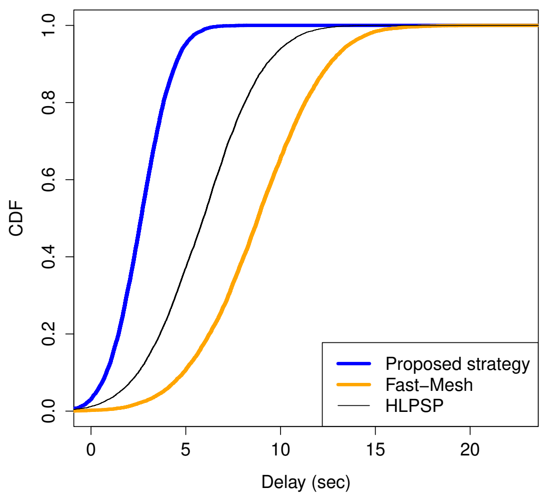
Figure 4.7: Percentage reduction in playback delay and hop count after ancestor replacement

due to above mentioned reason.

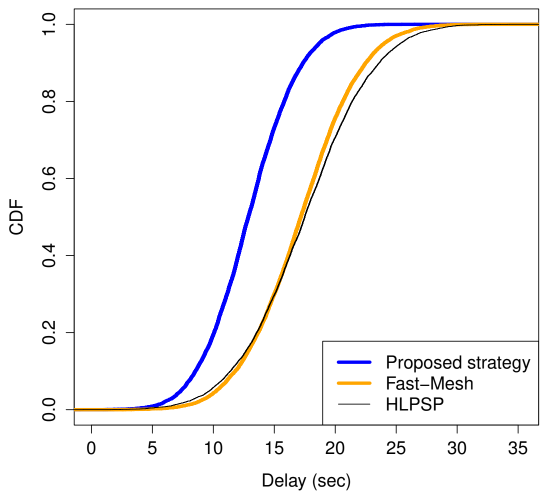
An intelligent peer selection strategy is likely to suffer from large computational overhead, which increases the startup delay. Fig. 4.8a compares the tracker-tier peer selection time of the proposed strategy with that in Fast-Mesh and HLPSP. It can be seen that proposed strategy has a higher tracker-tier delay when compared to Fast-Mesh because, Fast-Mesh uses random peer selection at the tracker-tier. However, both HLPSP and our strategy use parameter-based peer selection at the tracker-tier. It is found that load on tracker is higher in HLPSP because, the tracker checks upload capacity of the existing peers level-wise every time a new peer joins the system. On the other hand, as shown in Fig. 4.8b the buffering delay in Fast-Mesh and HLPSP is significantly higher than that in the proposed strategy. One of the reason is unavailability of sufficient data at the selected partners. This occurs mostly because newly joined peers are also suggested by the tracker as partners. Another reason for higher buffering delay in HLPSP is the frequent change of partners during



(a) CDF of Stage 1 or Tracker-tier Delay



(b) CDF of Buffering Delay



(c) CDF of Startup Delay

Figure 4.8: Comparing startup delay and its components without peer churn for 10,000 peers

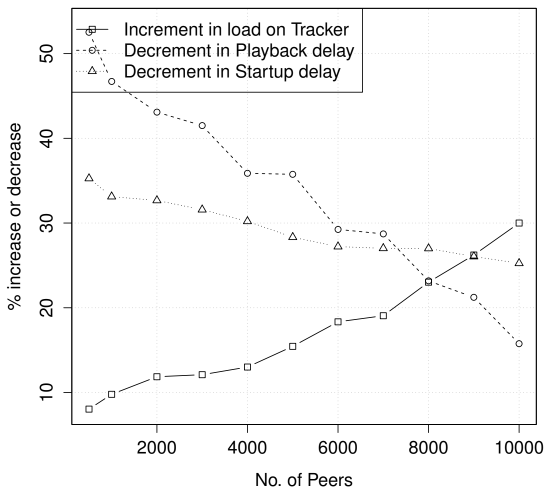


Figure 4.9: Comparing improvement in delay with the increase in overhead of the tracker in the proposed strategy

joining phase. In our proposal peers are selected considering data availability in their buffers at both stages to reduce the buffering delay significantly. The Stage 2 peer selection time is approximately same for all the three peer selection strategies.

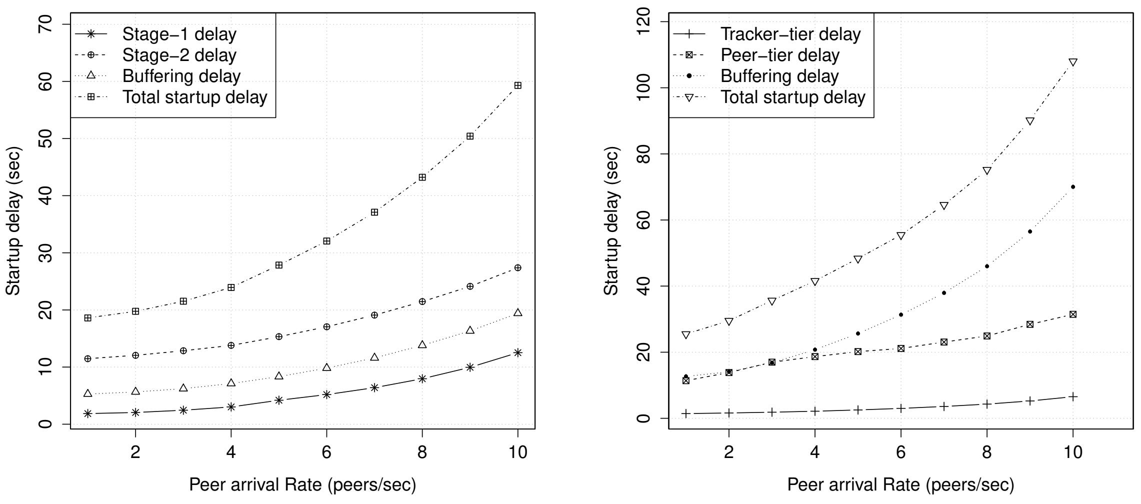
Fig. 4.8c shows the CDF of startup delay experienced by 10,000 peers in proposed strategy, HLPSP and Fast-Mesh. It can be observed that the proposed strategy outperforms Fast-Mesh and HLPSP. The decrease in buffering delay after selection of peers compensates the increase in tracker-tier peer selection time when compared to Fast-Mesh. Higher tracker-tier delay and buffering delay in HLPSP results in better performance of our strategy.

The above mentioned improvements in the playback delay and the startup delay is associated with a trade-off of increase in load on the tracker. Therefore, Fig. 4.9 shows the percentage of improvement in playback delay and startup delay as compared to Fast-Mesh for the corresponding increase in overhead at the tracker. This is because the selection of peers based on few parameters increases computational overhead of the tracker as compared to random selection. However, at the cost of this computational overhead the delay is improved significantly.

Table 4.2: Comparing performance of the proposed strategy and existing strategies under no churn scenario

Parameter	**Proposed Strategy** Mean (SD)		**Fast-Mesh** Mean (SD)		**HLPSP** Mean (SD)	**Improvement** (%)
	without adaptation	with adaptation	without adaptation	with adaptation		
Playback Delay	28.75 (16.5)	25.24 (16.4)	44.58 (20.20)	35.41 (19.6)	34.20 (18.9)	32 %
Hop count	16 (8.2)	14 (7.7)	21 (10.2)	18 (9.7)	16 (8.6)	15%
Startup Delay	12.90 (3.41)		17.12 (4.19		17.45 (4.77)	25%
a) Stage-1 or Tracker-tier delay	1.66 (0.17)		1.42 (0.08)		2.89 (0.25)	-15%
c) Buffering delay	2.63 (1.42)		7.79 (3.01)		5.89 (2.65)	42%

Overall, Table 4.2 shows that proposed strategy reduces the average playback delay by more than 15% and 30% compared to HLPSP and Fast-Mesh, respectively even without topology adaptation. When Fast-Mesh and our strategy use topology adaptation, the playback delay with the proposed strategy is reduced by 30-32%. Similarly, the maximum playback delay is 10% lower without adaptation relative to Fast-Mesh and 5% lower with adaptation. One of the major causes for the improvement in playback delay is found to be 15% reduction in the hop count of the path used to deliver stream chunks. The startup delay is also reduced by upto 25% due to the reduction in buffering delay by about 42%. However, the improvement



(a) Proposed Strategy

(b) Fast-Mesh

Figure 4.10: Comparing startup delay under flash crowd for different arrival rates

in playback delay and startup delay are achieved at the cost of about 15% higher overhead at the tracker.

Performance Evaluation with Peer Churn: To evaluate the performance of the proposed peer selection strategy with peer churn, flash crowd is simulated during initial 5% duration of the session. To observe the system performance under different intensities of flash crowd the peer arrival rate ranging from 1 peer/sec to 10 peers/sec is simulated [104, 107].

Fig. 4.10a and Fig. 4.10b present the impact of flash crowd on startup delay. It can be seen that with the increase in peer arrival rate, startup delay also increases in the proposed strategy as well as in the Fast-Mesh. On inspecting the components of startup delay it is found that the Stage 1 delay of proposed strategy increases by 22% and the tracker-tier delay of Fast-Mesh increases by 18% with the increase in arrival rate. In contrast to this, the buffering delay of proposed strategy increases by 14% and that of Fast-Mesh increases by 20%. The Stage 2 delay of proposed strategy and the peer-tier delay of Fast-Mesh increases by 10-12%. Therefore, the total startup

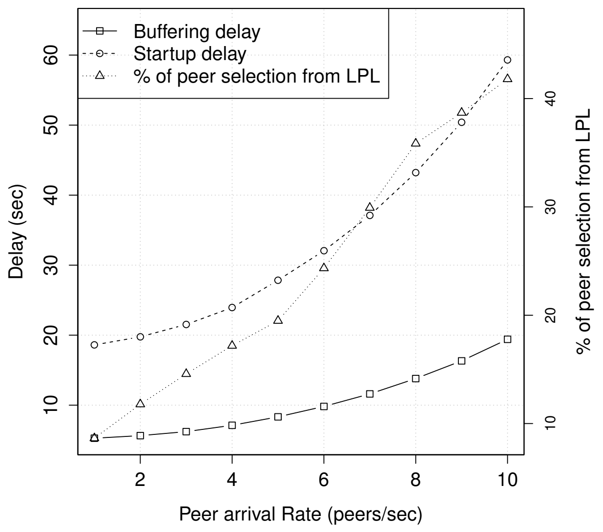


Figure 4.11: Percentage of peers selected from LPL during flash crowd and its impact on startup delay

delay of proposed strategy increases by 12% and in Fast-Mesh it increases by 16% with the increase in peer arrival rate. The increase in tracker-tier delay and peer-tier delay is caused due to increase in network traffic and computational load on tracker as well as at peers. On the other hand the increase in buffering delay is caused due to unavailability of sufficient chunks at the selected partners. Fig. 4.11 and Fig. 4.12 further substantiate this inference. Fig. 4.11 shows that with the increase in peer arrival rate the percentage of peers selected from LPL also increases and hence tracker suggests such partners who are waiting to receive chunks and do not have any data currently in their buffers. Fig. 4.12 compares the average buffering level of the peers in the proposed strategy and in the Fast-Mesh with respect to peer arrival rate. It can be observed that buffering level of the peers decreases with the increase in peer arrival rate by approximately 3%. This is because with an increase in peer arrival rate, the peers select more newly joined peers as partners which results in lack of chunk availability at peers.

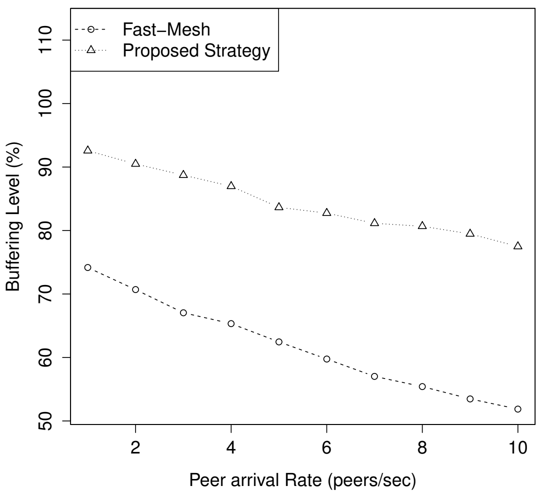


Figure 4.12: Comparing buffering level under flash crowd for different arrival rates

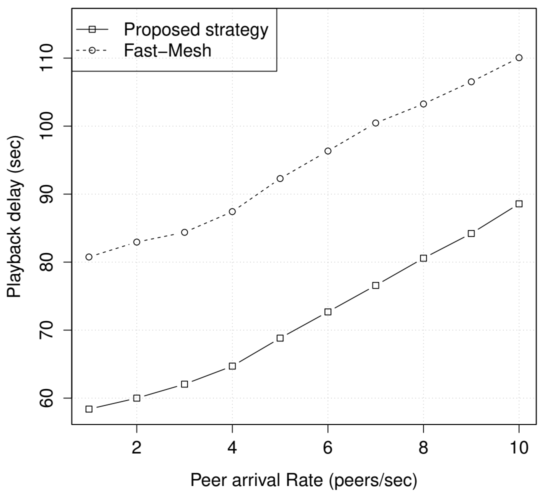


Figure 4.13: Comparing playback delay under flash crowd for different arrival rates

As shown in Fig. 4.10a and Fig. 4.10b, it is also observed that the proposed strategy results in lower startup delay compared to Fast-Mesh during flash crowd. This is due to significant improvement in buffering delay experienced by the peers in the proposed strategy.

Similar to the case of startup delay, the playback delay of the peers also increases with an increase in the peer arrival rate, as shown in Fig. 4.13. This is because the buffering level of the peers not only effects the startup delay, but also the playback delay. The buffering level of the partner represents the chunk upload rate of the partner which contributes to a significant amount of delay at each hop while disseminating chunks.

In the dynamic environment of peer-assisted systems it is difficult to maintain a stable overlay topology. Therefore, next we evaluate the two significant aspects of a dynamic peer-assisted system namely, stabilization duration and partner recovery time. The stabilization duration is the time system takes to cope up with the flash crowd and achieve the same average startup delay as system exhibits without flash crowd. The partner recovery time of the system is the average time peers take to select new partners and achieve same streaming experience after partner(s) departure.

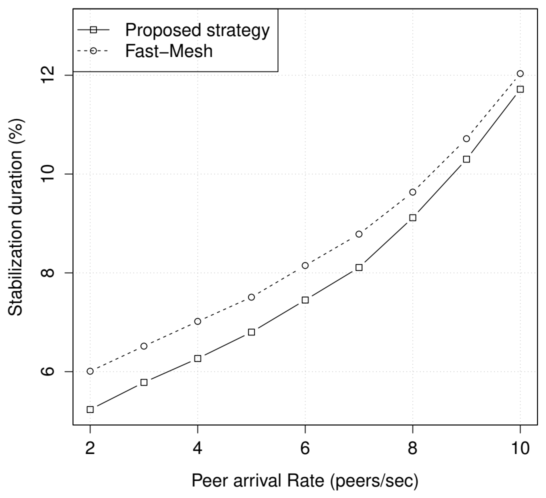


Figure 4.14: Comparing system stabilization duration under flash crowd for different arrival rates

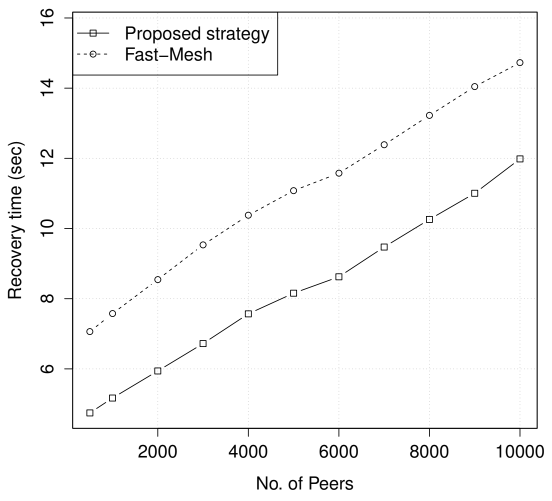


Figure 4.15: Partner recovery time during churn with varying number of peers

Fig. 4.14 shows the system stabilization duration for different peer arrival rate. We calculate the system stabilization duration as the percentage of stream session length. It is observed that the system takes more time to stabilize with the increase in peer arrival rate. This is due to the increase in computational delay and traffic overload at the tracker and at the peers. However, for 10,000 peers proposed strategy takes lower time to stabilize than Fast-Mesh. This is because in Fast-Mesh the tracker suggests newly joined peers as prospective partners. The newly joined peers may not have sufficient chunks in their buffers and hence it increases the startup delay of the peers which connect to them. In contrast to this, the proposed strategy selects only partners with chunks available in their buffers. This helps in stabilizing the system in lesser time compared to Fast-Mesh.

Table 4.3: Comparing performance of the proposed strategy and FastMesh under churn scenario

Parameter	**Proposed Strategy Mean (SD)**	**Fast-Mesh Mean (SD)**	**Improvement (%)**
Startup Delay	33.37 (13.1)	47.39 (17.7)	25%
a) Stage-1 or Tracker-tier delay	5.56 (3.4)	3.22 (1.6)	-7%
b) Stage-2 or Peer-tier delay	17.46 (5.1)	21 (5.9)	15%
c) Buffering delay	10.35 (4.5)	23.17 (10.3)	45%
Buffering level	84.39 (4.7)	61.72 (7.1)	35%
Playback delay	71.66 (10)	94.44 (9.9)	24%
Stabilization duration	7.57 (2.1)	8.13 (2)	8%
Partner recovery time	8.14 (2.2)	10.92 (2.4)	20%

Fig. 4.15 compares the recovery time of the proposed strategy with that for Fast-Mesh. The results show that the partner recovery time of proposed strategy is lower than that in Fast-Mesh. This is because in proposed strategy, after partner departure new partners are selected considering buffering level in addition to propagation delay and upload capacity. This ensures quick uploading of chunks and hence quick recovery from partner departure.

Overall, Table 4.3 shows that 7% increase in the stage-1 delay in the proposed strategy is compensated by the reduction in the buffering delay and stage-2 peer selection delay. This results in upto 25% improvement in total startup delay experienced by peers under churn scenarios. Further, 35% increase in the buffering level at the selected partners in the proposed strategy helped in reducing the playback delay by 24%. We also found that the proposed strategy takes 8% lesser time in stabilizing the overlay after flash crowd and 20% lesser time in partner recovery during peer departures.

Discussion: In the proposed strategy the system might face oscillations and instability due to two reasons one is due to peer traffic dynamics (peer churn) and another is due to its topology adaptation phase. Being an inherent drawback of a peer-assisted system the peer churn impacts the demand and supply ratio in the system and hence makes it unstable. We have evaluated and discussed how flash crowd and peer departure impact the streaming experience of the peers. It has been observed that as compared to existing strategies our proposed strategy performs better during churn. The flash crowd results in increased startup delay but, this is lower than that with the existing strategies. It is also observed that proposed strategy helps in stabilizing system faster by reducing the time taken by the system to achieve same average startup delay as system exhibits without flash crowd. Similarly, it is also observed that during partner departure the peers experience lesser partner recovery time in proposed strategy as compared to existing strategies.

The topology adaptation phase also makes the system unstable because peers keep changing their partners to move higher upload capacity peers closer to the source. However, we found that replacing a partner does not impact streaming experience of the peer due to the presence of multiple active partners. It is necessary to perform topology adaptation to achieve the primary goal of our system *i.e.*, reducing delay experienced by the peers. By keeping higher upload capacity peers closer to the source the number of hops traversed by the chunks is reduced and hence, per-hop delay is reduced which in turn, reduces the delay experienced by the peers.

4.4 Summary

In this chapter, we proposed a peer selection strategy to minimize playback delay and startup delay for the mesh overlays. The proposed peer selection strategy helps in selecting partners for the peers at three different stages of overlay construction. First, at the tracker-tier the tracker selects peers to suggest partners to newly joined peers. Second, at the peer-tier the newly joined peers select their partners out of those suggested by the tracker. At last the peers perform topology adaptation by selecting and replacing its ancestor(s) to reposition itself in the overlay. The topology adaptation process handles peer churn and helps in maintaining overlay which exhibits minimum playback delay. During all the three selections a peer is selected based on parameters like upload capacity, propagation delay, chunk buffering duration and buffering level of the peer.

To evaluate the performance of the proposed strategy it was compared with the Fast-Mesh and HLPSP. The performance was studied under two scenarios *i.e.*, without peer churn and with peer churn. The results showed that the playback delay improves by 30-32% under without churn environment, whereas the startup delay reduces by 20-25%. However, it also increases the computational overhead of the tracker by 10-15%. To evaluate the performance with peer churn, impact of flash crowd and peer departure was observed on various parameters like startup delay, playback delay, peer's buffering level, partner recovery time and stabilization duration. It had been observed that startup delay improves by 25%, whereas playback delay improves by 24% under peer churn. The system takes 8% lesser time to stabilize in the proposed strategy after flash crowd. The average partner recovery time of peers is also upto 20% lower.

The parameter-based peer selection strategy proposed in this chapter increases the partner selection time due to which the startup delay is marginally improved. The topology adaptation mechanism proposed in this chapter is required to maintain lower playback delay during churn, but also introduces message passing overhead. The proposed strategy improves playback delay significantly, while streaming quality and startup delay also need to be improved because they impact the lifetime and upload contribution of peers in the system.

In the next chapter, we propose an overlay management strategy that organizes peers in the hybrid tree-mesh overlay to improve streaming quality, startup delay and upload capacity utilization of peers, while considering heterogeneity in bandwidth and lifetime of peers.

Chapter 5

An Overlay Management Strategy for Hybrid Tree-mesh Overlays

5.1 Introduction

Peer-assisted live streaming systems fail to deliver the desired QoS and utilize the upload capacity of the peers, mainly due to the heterogeneity in session duration (lifetime) of peers and their upload capacity [38, 58]. Therefore, it is desirable to organize peers in the tree and mesh part of the hybrid tree-mesh overlay considering the heterogeneity. In [14, 57], authors considered upload capacity of peers while organizing them in the overlay. Authors of [57] proposed that CDN servers provide stream data to only those peers who are labeled as choke point expansion nodes and super nodes. The choke point expansion nodes are those that supply data to a large number of peers and the super nodes have higher upload capacity, sufficient data in buffer and are highly stable. In LiveSky [14], the peers that become LiveSky clients by contributing their bandwidth receive better streaming quality.

Overall, the existing systems do not predict stability of peers precisely because it also depends on the streaming quality received by the peer instantaneously in addition to its session duration. They also neglect utilizing the upload capacity of peers with short session duration and/or low bandwidth. CDN servers are used to provide stream data limiting the QoS to the scalability of servers. They also do not consider serviceability of peers while organizing them in the overlay, which not only depends on their stability, but also on chunk availability, upload and download capacities altogether.

In this chapter, we propose an overlay management strategy that focuses on enhancing the upload capacity utilization of peers and improving QoS in terms of streaming quality and startup delay. It organizes peers to create a hybrid tree-mesh overlay considering their serviceability, defined in terms of stability, streaming quality, upload and download capacities. The features of the proposed strategy that helps in organizing peers based on their serviceability are as follows:

- A CDN server selection strategy for new peers to balance the stream delivery load of CDN servers and peers.
- Creating a hybrid tree-mesh overlay topology with a resilient tree to generate stable and high upload capacity seeders.
- A bandwidth allocation mechanism for peers to distribute their upload and download capacities among sub-overlays.
- Creating virtual sources using peers to ensure quick startup with better streaming quality to new peers.
- A serviceability-based peer selection strategy to select partners for mesh peers that ensures better streaming quality.
- A topology adaptation strategy to replace partners during churn to maintain streaming quality.

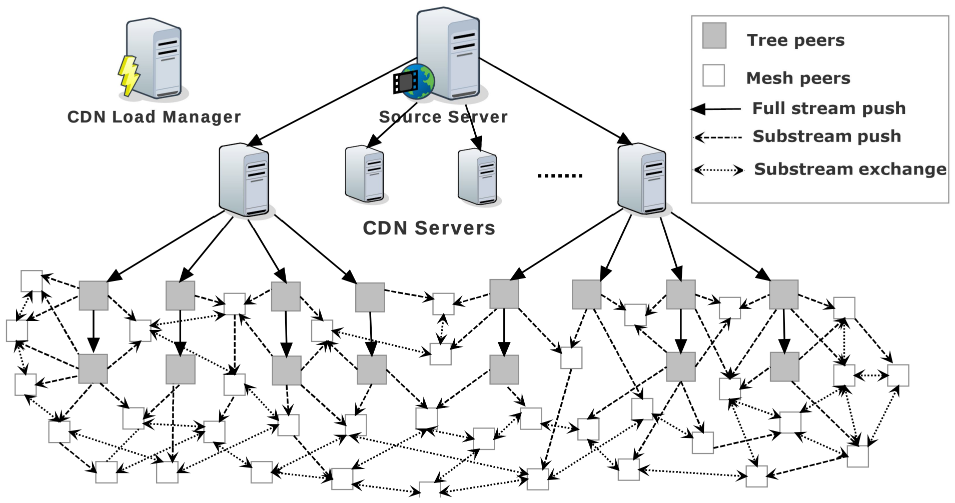


Figure 5.1: A peer-assisted live streaming system with hybrid tree-mesh overlay

The performance of the proposed strategy is compared with LiveSky [14] and PROSE [57] to show that the upload capacity utilization of peers is enhanced by 30%, while the startup delay and streaming quality of peers are improved by 20% and 25%, respectively.

Organization of the Chapter: The rest of this chapter is organized as follows. Section 5.2 presents the details of the proposed overlay management strategy. The results from simulation and the discussion are presented in Section 5.3. Section 5.4 concludes the chapter.

5.2 Proposed Overlay Management Strategy

In this section, we first state the system model and then present the proposed overlay management strategy in detail along with its various modules.

5.2.1 System Model

The peer-assisted live streaming system considered in this work is broadly similar to LiveSky [14]. Fig. 5.1 illustrates the system, consisting of a source server, a number of CDN servers, a CDN load manager, and peers. The source generates video stream in the form of equal and small-sized blocks/chunks distributed as sub-streams to CDN servers. The sub-streams are created so that a peer does not need all the sub-streams to play the video. But, downloading each additional sub-stream improves the quality of video playback. The peers advertise the aggregate upload capacity while joining the system (could also be zero). The download/upload capacity are specified in terms of the number of sub-streams a peer can download/upload.

The overlay consists of multiple connected sub-overlays such that each sub-overlay is built and maintained by a CDN server. A peer may be part of many sub-overlays by sharing its upload and download capacity. The CDN load manager selects the CDN servers for a new peer joining the system. The selected CDN server then helps the new peer with the stream data as well as information about the other peers in the sub-overlay. The topology of a sub-overlay is a hybrid tree-mesh topology. Peers which are part of the tree topology are termed tree peers and those that are part of the mesh topology are termed mesh peers. Considering the role of peers in the stream delivery, we use the terms parent peer (or parent partner) and child peer (or child partner) in this chapter.

Next, we describe the various modules of the proposed overlay management strategy, which mainly include CDN server selection and maintaining the hybrid tree-mesh topology at the sub-overlay. The various steps used in joining a new peer and maintaining the hybrid tree mesh overlay are also illustrated in Fig. 5.2.

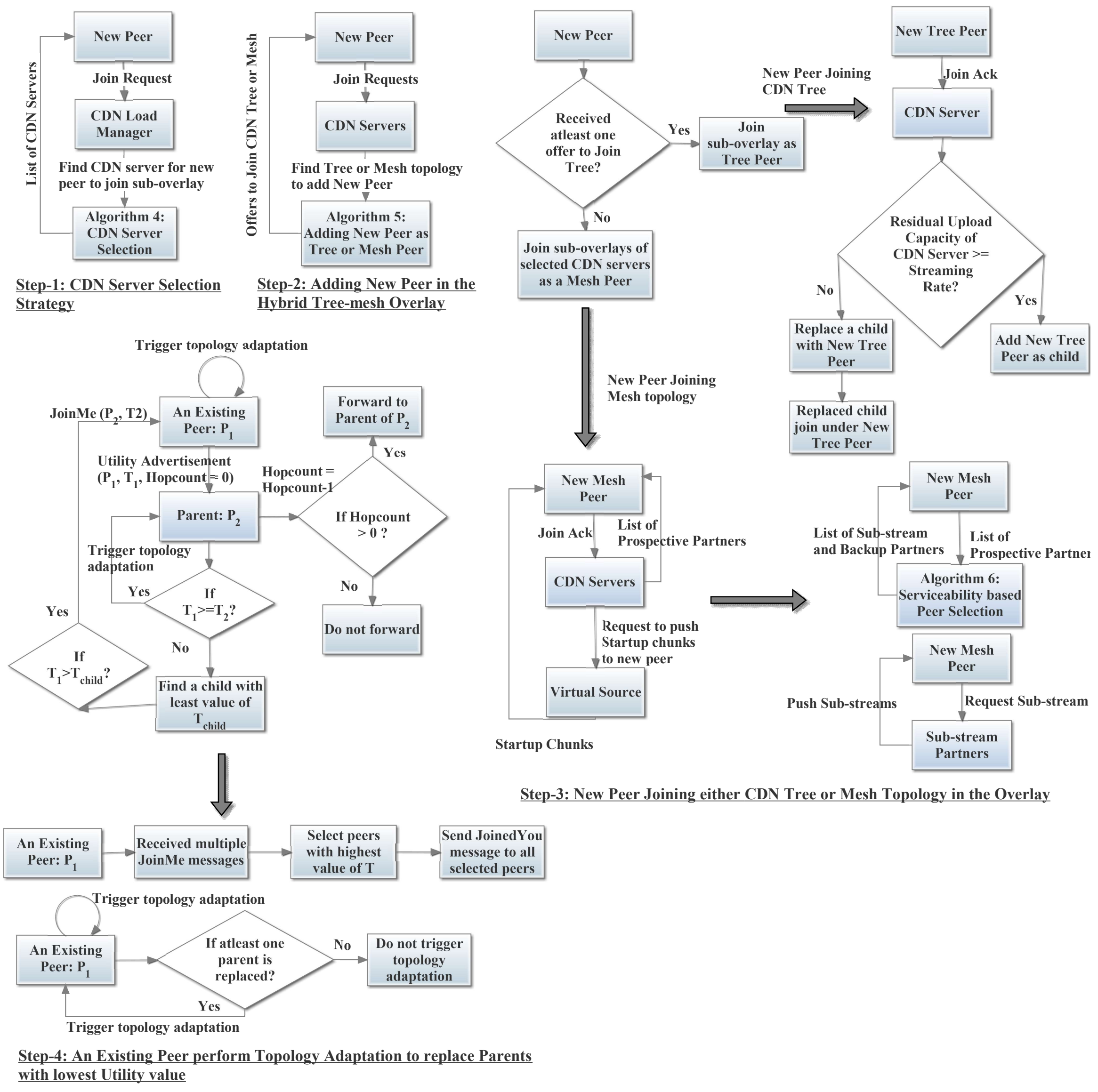

New Peer
Join Request
CDN Load Manager
Find CDN server for new peer to join sub-overlay
Algorithm 4: CDN Server Selection
List of CDN Servers
Step-1: CDN Server Selection Strategy
New Peer
Join Requests
CDN Servers
Find Tree or Mesh topology to add New Peer
Algorithm 5: Adding New Peer as Tree or Mesh Peer
Offers to Join CDN Tree or Mesh
Step-2: Adding New Peer in the Hybrid Tree-mesh Overlay
New Peer
Received atleast one offer to Join Tree?
Yes
Join sub-overlay as Tree Peer
No
Join sub-overlays of selected CDN servers as a Mesh Peer
New Peer Joining CDN Tree
New Tree Peer
Join Ack
CDN Server
Residual Upload Capacity of CDN Server >= Streaming Rate?
No
Yes
Replace a child with New Tree Peer
Replaced child join under New Tree Peer
Add New Tree Peer as child
New Peer Joining Mesh topology
New Mesh Peer
Join Ack
List of Prospective Partners
CDN Servers
Request to push Startup chunks to new peer
Virtual Source
Startup Chunks
New Mesh Peer
List of Sub-stream and Backup Partners
List of Prospective Partners
Algorithm 6: Serviceability based Peer Selection
New Mesh Peer
Push Sub-streams
Request Sub-stream
Sub-stream Partners
Step-3: New Peer Joining either CDN Tree or Mesh Topology in the Overlay
Trigger topology adaptation
JoinMe (P2, T2)
An Existing Peer: P1
Forward to Parent of P2
Yes
Utility Advertisement (P1, T1, Hopcount = 0)
Hopcount = Hopcount-1
Parent: P2
If Hopcount > 0 ?
Trigger topology adaptation
Yes
If T1>=T2?
No
Do not forward
Yes
No
If T1>Tchild?
Find a child with least value of Tchild
An Existing Peer: P1
Received multiple JoinMe messages
Select peers with highest value of T
Send JoinedYou message to all selected peers
Trigger topology adaptation
An Existing Peer: P1
If atleast one parent is replaced?
No
Do not trigger topology adaptation
Yes
Trigger topology adaptation
Step-4: An Existing Peer perform Topology Adaptation to replace Parents with lowest Utility value

Figure 5.2: Proposed overlay management strategy

5.2.2 CDN Server Selection Strategy

The proposed CDN server selection strategy aims to distribute the the stream delivery load of servers and peers among sub-overlays and provide better QoS to peers. The CDN load manager selects most suitable servers for the new peers to help them in joining corresponding sub-overlays. It calculates the suitability values of all servers as the normalized sum of geographical distance of servers from the new peer, fraction of peers experiencing larger than average joining delay and inadequate chunk rate in the respective sub-overlays. The closest servers with peers experiencing least joining delay and receiving desired chunk rate in their corresponding sub-overlays are considered the most suitable ones. The number of servers a peer selects to join depends on the population of peers, number of servers, and supply-demand ratio of upload capacity in the overlay.

Algorithm 4: CDN Server Selection Algorithm

```
Input: $JoinReq(\mathbf{n}) \leftarrow$ Join request sent by new peer '$n$'
$Z \leftarrow$ Set of all CDN servers
$d_i \leftarrow$ Geographical distance of the CDN server '$i$' from the new peer '$n$'
$F_i \leftarrow$ Fraction of peers experiencing larger than average joining delay and
inadequate chunk rate in the sub-overlay of server '$i$'
Output: Most suitable CDN servers for the new peer.
$Z_A \leftarrow$ Suitability value of a server $A$, where $A \in Z$
$Z_B \leftarrow$ Suitability value of a server $B$, where $B \in Z$
$Join(\mathbf{A}) \leftarrow$ Response to inform '$n$' to join sub-overlay of server '$A$'
$Join(\mathbf{B}) \leftarrow$ Response to inform '$n$' to join sub-overlay of server '$B$'

Calculate $Z_i = \frac{F_i}{\max\limits_{\forall j \in Z}(F_j)} + \frac{d_i}{\max\limits_{\forall j \in Z}(d_j)}$ for each server '$i$' in set $Z$.
Select server $A$ and server $B$ such that $Z_A \leq Z_B \leq Z_i, \forall i \in Z$
Send $Join(\mathbf{A})$ and $Join(\mathbf{B})$ to new peer '$n$'.
```

To keep it simple, let us assume that a peer can join two servers. The load manager finds out two most suitable servers A and B for the new peer, in that order. Algorithm 4 shows how CDN load manager selects two most suitable CDN servers for the new peer. For this selection, the load manager maintains a record of the fraction of peers experiencing larger than average joining delay and receiving inadequate chunk rate in the sub-overlays. This information is obtained from the servers of the corresponding sub-overlays regularly. The joining delay of a peer is the time it takes to join the sub-overlay either as a tree peer or as a mesh peer and starts receiving the stream. When a new peer starts its video playback it sends a *start playback message* to the server of the joined sub-overlay. This helps the servers in calculating the joining delay experienced by peers in the sub-overlay. Each peer of the sub-overlay also informs its server whenever it receives chunk rate lower than its desired rate to calculate the fraction of peers receiving inadequate chunk rate.

CDN server selection considering the fraction of peers receiving inadequate chunk rate helps a new peer in joining the sub-overlays where peers are satisfied with the streaming quality. It also indicates that the sub-overlay is relatively less loaded with the streaming traffic and the new peer can receive desired streaming streaming quality. The fraction of peers experiencing larger than average joining delay indicates the load of servers to provide startup chunks and information of existing peers to the new peer. Selecting a server with smaller value of this fraction ensures that it is lightly loaded and can support new peers in joining the overlay faster. These two parameters also help in distributing stream delivery load among the sub-overlays. Considering geographical distance along with these parameters helps in selecting the closest lightly loaded servers and sub-overlays for the new peers.

5.2.3 Creating and Maintaining Hybrid Tree-Mesh Overlay

In our work, the objective of building the hybrid tree-mesh sub-overlay is to enhance the utilization of upload capacity of peers and improve QoS during churn. Peers join either the extended CDN tree of the overlay (called tree peers) or the mesh overlay (called mesh peers). Next, we discuss how new peers are added either as tree peers or as mesh peers in the sub-overlays.

Algorithm 5: Adding a new peer as Tree or Mesh peer

```
Input: $Join(\mathbf{Req}) \leftarrow$ Join request sent by new peer '$n$'
$U_A^r$ (or $U_B^r$) $\leftarrow$ Residual upload capacities of server $A$ (or $B$)
$D_n^a \leftarrow$ Aggregate download capacity of the new peer '$n$'
$U_n^a \leftarrow$ Aggregate upload capacity of the new peer '$n$'
$Child \leftarrow$ Set of all the tree peers which are directly connected to the CDN server
Output: Response for the join request ($Join(\mathbf{Req})$).
$Join(\mathbf{CDNTree}) \leftarrow$ Server's response to new peer to join as tree peer
$Join(\mathbf{Mesh}) \leftarrow$ Server's response to new peer to join as mesh peer
if $D_n^a \geq C$ & $U_n^a \geq C$ then
    if $U_A^r > C$ $OR$ $U_n^a \geq U_i^a, \forall i \in Child$ then
        Send $Join(\mathbf{CDNTree})$
else
    Send $Join(\mathbf{Mesh})$
```

Adding a New Peer into Tree or Mesh: With the help of the load manager a new peer sends a *join request message* to the selected CDN servers (A or B) to join the corresponding sub-overlay(s) (denoted as $subA$ and $subB$). The message includes the aggregate upload and download capacity of the new peer. Algorithm 5 shows the method used by the servers to decide whether the new peer should be added to extended CDN tree or mesh considering their instantaneous residual upload capacity, the aggregate download and upload capacities of the new peer.

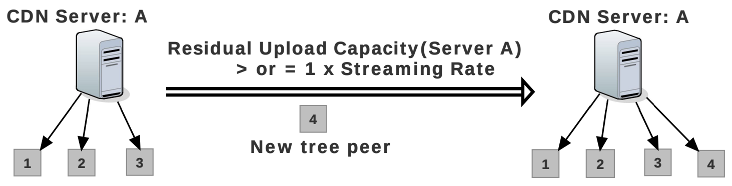
A server finds the new peer eligible to be a tree peer, if aggregate upload and download capacity of the new peer are greater than or equal to streaming rate. But, it sends a response to join the extended CDN tree to the eligible new peer only if one of the following two conditions is satisfied: i) residual upload capacity of the server is greater than the streaming rate, or ii) the aggregate upload capacity of the new peer is greater than the aggregate upload capacity of atleast one of the current children of the server. Otherwise, the server responds the new peer with an offer to join as a mesh peer. If a new peer receives an offer to join the extended CDN tree from at least one of the servers then it joins as a tree peer, otherwise it joins sub-overlays of both the servers as a mesh peer. A new per may also receive offers to join the extended CDN tree from both the servers. In this condition the new peer joins the server with higher suitability value.

The upload and download capacity of a tree peer are completely dedicated to the joining sub-overlay. For a mesh peer, it is distributed among both the joining sub-overlays. This is because the tree peers receive full stream either from the CDN server or other tree peers whereas, mesh peers can receive sub-streams either from the tree or mesh peers. The detailed procedure of upload and download capacity allocation of mesh peers is discussed later in Sec. 5.2.5.

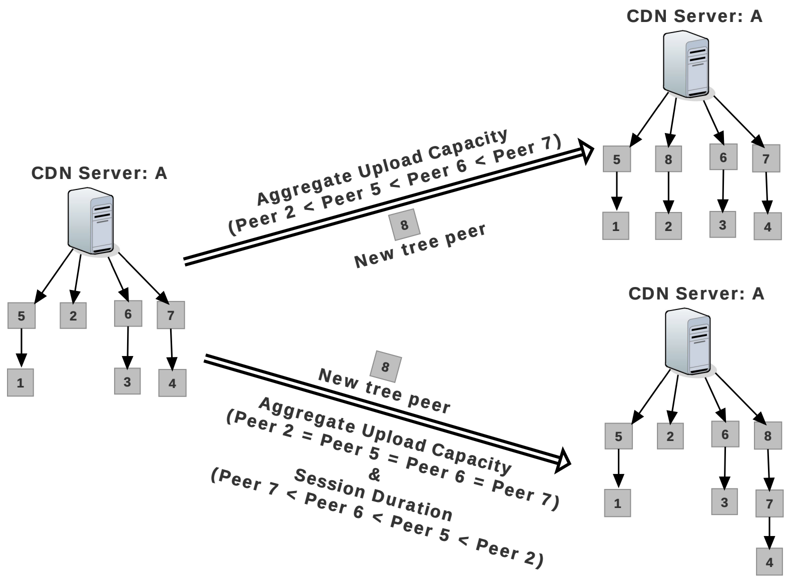
5.2.4 Creating and Maintaining Extended CDN Tree

The extended CDN tree is formed to ensure QoS to high upload capacity peers, which in turn generates stable seeders. In the tree, each peer has only one parent which provides the full stream to the peer and one child to which the peer provides the full stream. This helps in handling peer churn and maintaining stream continuity among tree peers. The residual upload capacity of intermediate nodes and the aggregate upload capacity of leaf nodes are utilized to deliver substreams to mesh peers.

Adding a New Tree Peer: A new peer joins as a direct child to the server as shown in Fig. 5.3. If the server does not have enough residual upload capacity to serve full stream to the new peer, one of the existing children is replaced by the new peer. The replaced peer along with its descendants joins the new peer. The peer for replacement is selected based on upload capacity and elapsed session duration, as shown in Fig. 5.3b. The peer with least upload capacity is selected. If all the children have same upload capacity, then the peer with least elapsed session duration is selected to take care of stability. We do not consider other parameters such as streaming quality received by the peer and its download capacity to predict its stability, because the peer to be replaced is directly connected to the CDN server.



(a) Case 1: Residual upload capacity of CDN server is greater than streaming rate


CDN Server: A
5
2
6
7
1
3
4
Aggregate Upload Capacity
(Peer 2 < Peer 5 < Peer 6 < Peer 7)
8
New tree peer
CDN Server: A
5
8
6
7
1
2
3
4
8
New tree peer
Aggregate Upload Capacity
(Peer 2 = Peer 5 = Peer 6 = Peer 7)
&
Session Duration
(Peer 7 < Peer 6 < Peer 5 < Peer 2)
CDN Server: A
5
2
6
8
1
3
7
4

(b) Case 2: Replace one of the existing children

Figure 5.3: Adding a new tree peer

This peer joining process takes care that the highest upload capacity peers are closer to server in the extended CDN tree. The peers are added to the tree only in the initial half of the streaming session to avoid increasing the height of the tree in an unlikely scenario of peers joining in the increasing order of their upload capacity. This also increases the possibility of adding only stable peers to the tree because, peers who join later cannot have total lifetime greater than those who joined in the initial half of the streaming session and stay till the end.

Handling Tree Peer Departure: Since the tree peers receive prioritized service by either connecting to the server or other peers with high upload capacity, the probability of their departure is small. The tree peer may leave the overlay only due to lack of interest in the content. To preserve stream continuity, each tree peer maintains information of its ancestors as well as its descendants. The CDN server also maintains a record of all the tree peers who join its sub-overlay along with their parent and child information. When a new tree peer joins the sub-overlay, it sends its address to its descendants upto two hops and the CDN server also makes a new entry in its record. In response to the message from a new peer the descendant peers also send their address along with their parent's address. If a tree peer leaves the overlay then the orphaned child along with its descendants join its grandparent. In a very unlikely scenario, when more than one consecutively connected tree peers leave the overlay then orphaned peers request the CDN server to provide information of the ancestor with departed child. Therefore, stream recovery is quick in case of tree peer departure. The tree peers can also temporarily fetch chunks from the server if they run out of buffer.

Utilizing Residual Upload Capacity of Tree Peers: The residual upload capacity of a tree peer after serving one stream to its child is utilized to serve sub-streams to mesh peers. A tree peer is considered stable when the elapsed session duration is greater than half of the average value in the same sub-overlay. Until a tree peer becomes stable, its residual upload capacity is used by the server to create virtual sources to serve startup chunks to newly joined mesh peers (details are discussed in the next section).

Overall, the construction of extended CDN tree helps in utilizing upload capacity of peers to ensure QoS. It also utilizes the upload capacity of newly joined, not yet stable, tree peers for providing startup chunks to the mesh peers.

5.2.5 Creating and Maintaining Mesh Topology

First, each new peer distributes its upload and download capacities among different sub-overlays of the selected CDN servers. Each server creates a virtual source that provides startup chunks to the new peer. Servers also suggest a list of prospective partners to the new peer. Next, the new peer selects its partners considering their serviceability. Each mesh peer also use a topology adaptation strategy to replace partners that don't provide the desired quality of stream. With these operations, quick startup and the desired streaming quality are ensured to the mesh peers. Next, we explain these operations in detail.

Upload and Download Capacity Allocation of Peers: When a new peer receives offers to join as a mesh peer, it distributes its upload and download capacity among sub-overlays of those servers. The upload and download capacities of a peer are allocated in terms of the number of sub-streams it uploads and downloads from sub-overlays.

For this, the selected servers (A and B) first calculate the aggregate upload and download capacities of their sub-overlays ($subA$ and $subB$) an then send this information to the new peer. The aggregate upload capacity of a sub-overlay is calculated as the sum of total upload capacities allocated by the peers in that sub-overlay and the corresponding server. The aggregate download capacity of a sub-overlay is the sum of total download capacities allocated by the peers in that sub-overlay. Let U^a_{subA} and U^a_{subB} be the aggregate upload capacities and let D^a_{subA} and D^a_{subB} be the aggregate download capacities of the sub-overlays. Each new peer first calculates the fraction by which its upload and download capacities are divided among sub-overlays of selected servers (denoted as β). β is calculated considering the supply and demand ratio of upload capacities in the sub-overlays, as

$$\beta = \frac{(U^a_{subA}/D^a_{subA}) - (U^a_{subB}/D^a_{subB})}{(U^a_{subA}/D^a_{subA}) + (U^a_{subB}/D^a_{subB})} \tag{5.1}$$

Table 5.1: Calculating U^A_n, U^B_n, D^A_n and D^B_n

Condition	**Value of U^A_n**	**Value of U^B_n**	**Value of D^A_n**	**Value of D^B_n**
If $\beta > 0$ & $\beta < 1 - \beta$	$\Vert \beta \times U^a_n \Vert$	$\Vert (1-\beta) \times U^a_n \Vert$	$\Vert (1-\beta) \times D^a_n \Vert$	$\Vert \beta \times D^a_n \Vert$
If $\beta > 0$ & $\beta > 1 - \beta$	$\Vert (1-\beta) \times U^a_n \Vert$	$\Vert \beta \times U^a_n \Vert$	$\Vert \beta \times D^a_n \Vert$	$\Vert (1-\beta) \times D^a_n \Vert$
If $\beta < 0$ & $\vert\beta\vert < 1 - \vert\beta\vert$	$\Vert (1 - \vert\beta\vert) \times U^a_n \Vert$	$\Vert \vert\beta\vert \times U^a_n \Vert$	$\Vert \vert\beta\vert \times D^a_n \Vert$	$\Vert (1 - \vert\beta\vert) \times D^a_n \Vert$
If $\beta < 0$ & $\vert\beta\vert > 1 - \vert\beta\vert$	$\Vert \vert\beta\vert \times U^a_n \Vert$	$\Vert (1 - \vert\beta\vert) \times U^a_n \Vert$	$\Vert (1 - \vert\beta\vert) \times D^a_n \Vert$	$\Vert \vert\beta\vert \times D^a_n \Vert$
If $\beta = 0$	$\Vert U^a_n/2 \Vert$	$\Vert U^a_n/2 \Vert$	$\Vert D^a_n/2 \Vert$	$\Vert D^a_n/2 \Vert$

Let the aggregate upload and download capacities of a new peer n be denoted by U^a_n and D^a_n, respectively. If D^a_n is greater than the streaming rate then it is set equal to the streaming rate. Let U^A_n and U^B_n be the fractions of upload capacity of the new mesh peer allocated to servers A and B, respectively with $U^a_n = U^A_n + U^B_n$. Similarly, let D^A_n and D^B_n be the fractions of download capacity of the new mesh peer allocated to servers A and B, respectively with $D^a_n = D^A_n + D^B_n$. The rules for calculation of these parameters are given in Table 5.1.

The new peer allocates a higher fraction of its upload capacity to the sub-overlay where the ratio of aggregate upload and download capacities is smaller. Contrarily, a higher fraction of its download capacity is allocated to the sub-overlay where the ratio of aggregate upload and download capacities of the sub-overlay is larger. This allocation balances the stream delivery load of peers and servers in the sub-overlays and also ensures better QoS, because more sub-streams are downloaded from lightly loaded sub-overlays. Next, we discuss the procedure used by a CDN server to create, maintain and utilize virtual sources to give startup chunks to the mesh peers.

Virtual Sources: A virtual source consists of a group of peers providing a sub-stream to the new mesh peer. Servers designate virtual sources by reserving a portion of upload capacity of a few peers in the sub-overlay. The virtual sources ensure quick startup with desired streaming quality to the mesh peers and prevent the initial departure of peers. A server creates a virtual source when a new peer joins the mesh sub-overlay. The number of peers in a virtual source is equal to the number of sub-streams the new peer can download. To find these peers, the server maintains a reserved peers list that consist of peers who joined sub-overlay before the concerned new peer. The peers are arranged in the list in the decreasing order of their joining time from top to bottom. The server selects a larger fraction of peers from the top of the list to include both newly joined and stable peers. The stable peers ensure consistent delivery of stream to the new peer. The selection of larger fraction of new peers improves their upload capacity utilization.

Maintaining virtual sources: The CDN servers continue adding newly joined peers to the reserved peers list as soon as they receive the stream. A newly joined mesh peer is added to the list after reserving its upload capacity worth one sub-stream whereas, a newly joined tree peer is added after reserving its residual upload capacity. As explained earlier (see Sec. 5.2.4), the residual upload capacity of a tree peer is reserved and utilized to create virtual sources until it becomes stable. This ensures better streaming quality to mesh peers during startup because tree peers are connected closer to the CDN servers.

The peers in the list are maintained considering peer arrival rate, average streaming rate and reserved upload capacity of peers. Let the arrival rate be λ peers/unit time and the desired average streaming rate be R chunks/unit time. Let X be the set of peers available in the reserved peers list and X_i be the amount of reserved upload capacity of a peer i, $i \in X$. The size of the reserved peers list is maintained such that the inequality in Eq. 5.2 is satisfied throughout the streaming session.

$$\lambda \times R < \sum_{i=1}^{|X|} |X_i| \tag{5.2}$$

The above condition reserves upload capacity for serving startup chunks to mesh peers until the peer arrival rate either stabilizes or decreases. Once the peer arrival rate reduces, peers from the bottom of the list are released to prioritize the release of upload capacity of peers with longer elapsed session duration. This ensures that upload capacity of stable peers is utilized more to serve sub-streams to mesh peers by becoming their partners.

Stream delivery using virtual sources: Each peer who is a part of the virtual source (group) delivers its best quality sub-stream to the new peer until the new peer selects partners and receives the stream from them. It finds out the best quality sub-stream using the number of chunks received before the playback deadline. Let the chunks be generated at the rate of C chunks/unit time. Let $q_{i,j}$ be the quality of the i^{th} sub-stream received by a peer j. Let $N_{i,j}$ be the number of chunks of i^{th} sub-stream received by a peer j before playback deadline. Let l_j be the elapsed session duration of j^{th} peer and M be the total number of sub-streams created by the source. Eq. 5.3 defines the quality of a sub-stream as the ratio of number of chunks received by a peer before the playback deadline to the total number of chunks generated at the source for the same sub-stream.

$$q_{i,j} = \frac{N_{i,j}}{(C/M) \times l_j}, 1 \leq i \leq M \tag{5.3}$$

Each peer j delivers the i^{th} sub-stream to maximize $q_{i,j}$, $i \in \{1 \ldots M\}$. If more than one peer of the designated virtual source delivers the same sub-stream to the new peer, then the new peer requests the peer with lower quality sub-stream to deliver an another required sub-stream. When a virtual source is not able to provide desired quality to the new peer, it fetches missing chunks from the server.

Overall, virtual sources ensure quick startup with the desired streaming quality to new mesh peers by utilizing upload capacity of the existing peers. It reduces the stream delivery load of servers and utilizes the upload capacity of peers with shorter session duration and lower upload capacity.

Serviceability based Peer Selection: The serviceability of a peer is measured considering its stability, streaming quality, upload and download capacities. It represents the ability of a peer to deliver a desired quality of stream to other peers consistently.

First, each new mesh peer receives the list of prospective peers from the selected servers and the virtual source. The servers suggest tree peers with free upload capacity and peers of the virtual source. Each peer of the virtual source also suggests its parents or partners as prospective partners to the new peer. The suggested list of prospective partners altogether consists of tree peers, new peers and stable peers. This helps a new peer to select partners that have chunks in the buffers, free upload capacity and are consistent in providing good quality stream.

The new peer use this list to find the partners with residual upload capacity worth one sub-stream. Then, it calculates their serviceability using the elapsed session duration, download capacity, received streaming quality, and total stream duration. The new peer select prospective partners with highest serviceability value as partners, as shown in Algorithm 6. These partners are further categorized as sub-stream partners and backup partners. A sub-stream partner continuously pushes the selected sub-stream(s) to the peer, whereas backup partners are used to recover the missing chunks due to packet loss or partner departure.

Algorithm 6: Serviceability based Peer Selection

```
Input: $P \leftarrow$ Set of '$K$' prospective partners received by new mesh peer '$n$'
$M \leftarrow$ Number of sub-streams.
Output: Set of sub-stream and backup partners of new mesh peer '$n$'
for $k = 1$ to $2M$ do
    Find $\max_{\forall i \in P}(S_i)$
    Add peer '$i$' to $PartnersList$
    Remove peer '$i$' from $P$
for $i = 1$ to $M$ do
    Find $\max_{\forall j \in PartnersList}(q_{ij})$
    Select peer '$j$' as sub-stream partner to retrieve sub-stream '$i$'.
Select rest of the peers from $PartnersList$ as backup partners.
```

Let P be the subset of prospective partners with residual upload capacity worth one sub-stream, l_i be the elapsed session duration of the i^{th} peer, l_e be the elapsed stream duration and l be the total stream duration. Let N_i be the number of chunks received by the i^{th} peer before playback deadline, Q_i be the quality of stream received by the i^{th} peer and D_i^a be its download capacity. A peer first calculates the streaming quality of i^{th} peer, $i \in P$ as the ratio of the total number of chunks received by the peer before the playback deadline to the total number of chunks generated. The following equation defines this computation.

$$Q_i = \frac{N_i}{C \times l_i}, \forall i \in P \tag{5.4}$$

Next, the serviceability value of i^{th} peer, $i \in P$ is calculated as

$$S_i = \frac{l_e}{l}\left(\frac{l_i}{\max_{\forall j \in P}(l_j)}\right) + \left(1 - \frac{l_e}{l}\right)\left(\frac{Q_i}{\max_{\forall j \in P}(Q_j)} \times \frac{D_i^a}{\min\left(\max_{\forall j \in P}(D_j^a), C\right)}\right) \tag{5.5}$$

The serviceability is calculated as the weighted sum of the normalized values of the elapsed session duration, the received stream quality and the download capacity. The weights consider the elapsed and the total stream durations.

The rationale behind the selection of the parameters to calculate serviceability of a peer is as follows. The stability of a peer depends on the interest of user in the content and its quality, which can be predicted using the elapsed session duration, received quality of stream and the download capacity. It is assumed that the peers who stay for a longer duration are satisfied with the streaming quality. The streaming quality and download capacity also affect the probability of peer departure. The peer with higher download capacity has higher degree of connectivity and receives better quality, thereby remains unaffected by peer departure. The received quality also indicates the availability of chunks in the buffer, which predicts the ability to deliver better quality. The ratio of elapsed stream duration to the total duration gives higher weightage to the quality of stream and download capacity of the peer at the beginning of the session. This is due to the fact that all the peers have almost equal (and short) elapsed session duration initially, which cannot be used to predict satisfaction of the peers. With the progress in streaming session, the elapsed session duration of peers is diversified and can be used to predict the interest and satisfaction of peer with the content and quality of stream.

Selecting sub-stream and backup partners: After computing the serviceability, the new mesh peer selects its sub-stream partners and backup partners. A partner that provides a sub-stream with the highest quality becomes a sub-stream partner. The number of sub-stream partners selected is equal to the number of sub-streams the mesh peer can download. This is because each partner provides one sub-stream. The mesh peer also maintains a list of all the partners providing next best quality of the required sub-streams as backup partners.

When a sub-stream partner leaves, the mesh peer temporarily pulls chunks from the backup partners and also searches for another sub-stream partner based on the serviceability. Each mesh peer also maintains backup partners list because they play a significant role in maintaining quality and stability during churn. This strategy prevents peers from experiencing stream quality degradation for longer duration and provides faster recovery.

Topology Adaptation: To maintain streaming quality during churn, peers relocate themselves in the overlay by replacing their partners based on *utility value*. The partners with lowest utility value are replaced first. Let N_i^t be the number of sub-streams delivered by the i^{th} peer at time t. The utility T_i of the i^{th} peer is measured as the sum of its serviceability value and upload capacity utilization, as defined by

$$T_i = \frac{\sum_{t=0}^{l_i} N_i^t}{l_i \times U_i^a} + S_i \tag{5.6}$$

In Eq. 5.6, the ratio of number of sub-streams delivered by a peer to the total number it could have delivered (using its aggregate upload capacity) indicates the upload capacity utilization. A peer replaces its partners to connect with those who have better stability, streaming quality and upload capacity utilization.

The adaptation process is triggered either when a peer retrieves more than fifty percent of chunks from its backup partners instead of sub-stream partners or the servers trigger adaptation based on the peer arrival rate. In the former case, the peer initiates the adaptation process whereas, in the latter case, the server sends a trigger message to few of the newly joined and geographically diverse peers each time after ten percent peers join its sub-overlay. The newly joined peers forward the trigger message to their parent partners and the receiving partners initiate the adaptation process. The steps of adaptation process are described as follows:

- **Step-1:** Each peer initiating the adaptation first calculates its own utility value (using Eq. 5.6), which is advertised to its parents up to two hops. The utility advertisement has sender address, utility value and a hopcount flag set to two.

- **Step-2:** When a peer receives a utility advertisement, it first decrements the hopcount flag by one and then forwards it to parents, unless the flag becomes zero. Every time a peer receives a utility advertisement, it compares its own utility with the utility of sender. If its utility is smaller, then it starts adaptation process and advertises its utility value using the same procedure. Otherwise if the sender is not its child, it compares the utility values of its children with that of sender's. If it finds a child with smaller utility value, then it sends a *JoinMe* message to the sender with its utility value. Else, it discards the received utility advertisement.

- **Step-3:** When a peer receives *JoinMe* message(s), it compares the received utility values with that of its parents. It selects peers with highest utility value as its parent and replaces the existing parents if they have smaller utility value. The peer then sends a *JoinedYou* message to the new parents and retrieves best quality sub-streams from them. The *JoinedYou* message also consists of a list of peers rejected by the peer as parents.

- **Step-4:** When a peer receives a *JoinedYou* message in response to a *JoinMe* message, it abandons its child with least utility value and adds the sender of the *JoinedYou* message as a new child. It also sends the list of its current children and the list of peers available in the *JoinedYou* message to the abandoned child to help it find another parent.

- **Step-5:** Every time a peer replaces a parent, it starts a new adaptation and continues to do so until it receives no *JoinMe* messages in response.

The topology adaptation strategy maintains the streaming quality throughout the session by updating the partners. It also relocates peers with higher serviceability and upload capacity utilization closer to the server to ensure better QoS to all the peers. The message passing overhead of this strategy is also lower compared to the schemes where each peer triggers adaptation periodically irrespective of churn rate and peer population. This strategy frequently updates the overlay when the peer arrival rate is high. As the peer population increases and the peer arrival rate decreases, the frequency of adaptation is reduced to lower the message overhead.

5.2.6 Overhead and message passing complexity

When a new peer joins, it receives a message from the joined CDN server(s) which contains information such as streaming rate in terms of number of chunks per unit time (C), number of sub-streams generated by the source (M), total stream duration (l) and elapsed stream duration (l_e). The new peer also informs server about its aggregate or allocated upload and download capacity to the CDN server(s) of joining sub-overlays. This helps CDN servers as well as peers in calculating values of various parameters in Eq. 5.1 to 5.5, such as β, $q_{i,j}$, Q_i and S_i. Suppose there are total N number of peers join the system. Then the number of messages exchanged between peers and CDN servers to find out the above mentioned information and perform the calculations are $O(N)$.

Each peer also exchange messages with its prospective partners while selecting partners to find out their serviceability. The new peer requests its prospective partners to provide information such as session duration (l_i), number of chunks received before playback deadline (N_i) and download capacity (D_i). This information is used by new peer in predicting the serviceability of partners using Eq. 5.4 and Eq. 5.5. Later, peers also calculate and exchange their own utility value with their partners to maintain the streaming quality. The number of prospective partners and partners of each peer depends on its download capacity and are negligible with respect to total number of peers in the system. Therefore, the number of messages exchanged among peers are also $O(N)$. Overall, the cost and overhead of collecting all the information needed to implement proposed overlay management strategy is $O(N)$ when total N number of peers join the system.

Computational Complexity When a new peer joins, the CDN load manager performs first computation to find out the most suitable CDN servers for the new peer. If the number of CDN servers in the system are Z, then it may take $O(Z)$ units of time. Next, the CDN server adds the new peer either as tree peer or as mesh peer in the overlay after comparing the upload capacity of the new peer with the directly connected children peers of the server. If the outdegree of a server is Z_c, then it may take $O(Z_c)$ units of time to add a new peer in the sub-overlay. Further, when a new peer joins as mesh peer, it selects its partners to receive sub-streams using serviceability based peer selection algorithm. For each new mesh peer it takes $O(K)$ units of time to select its partners out of K prospective partners.

Here, outdegree of the CDN servers (Z_c), the number of CDN servers (Z) and number of prospective partners (K) of a peer are negligible compared to number of peers (N). Therefore, it takes $O(N)$ time for N number of peers to join and maintain the overlay.

5.3 Performance Evaluation

We evaluate the proposed strategy through simulations and compare its performance with two existing peer-assisted live streaming systems *i.e.*, PROSE [57] and LiveSky [14].

5.3.1 Evaluation Scenarios

The proposed strategy is implemented in OMNeT++ [97]. We use OverSim [98] framework to simulate P2P overlay and INET [99] framework to simulate underlay

TCP/IP network. The simulated network consists of a source server, a CDN load manager, six CDN servers and several peers. 20,000 peers join the system during a 100-minute streaming session. Each CDN server has an upload capacity of 200 Mbps [14]. The upload capacity distribution of peers is shown in Table 5.2, where approximately thirty percent of the peers are free riders and the total upload contribution of peers is approximately forty percent of the upload capacity of the system [14]. The source server uses Scalable Video Codec (SVC) standard [108] to encode the video stream and supports streaming rate upto 1 Mbps. The source generates five sub-streams. The peers download different number of sub-streams according to the desired streaming quality. The peers are categorized into three groups based on the desired streaming rate and the download capacity, as reported in Table 5.3 [101].

Table 5.2: Upload capacity distribution of peers

Peers %	**Upload capacity**
30	0 Kbps
50	512 Kbps
8	1.4 Mbps
12	1.6 Mbps

Table 5.3: Download capacity and desired streaming rate of peers

Peers %	**Download capacity**	**Desired streaming rate**
56	512 Kbps	503 Kbps
30	1 Mbps	705.3 Kbps
14	3 Mbps	905.8 Kbps

We simulate flash crowd during the initial ten percent duration of the session and peer departure throughout [32]. Peers arrive following Poisson distribution with mean inter-arrival time between 0.5 and 5 seconds [96, 101]. The session duration of peers follows Pareto distribution [56]. Peers depart either due to unacceptable QoS or after staying for its specified session duration. A peer waits for one minute after QoS deterioration and leaves the system afterwards if it cannot recover QoS. The peers are set to randomly tolerate 10 to 20 percent degradation in desired streaming rate and 15 to 45 seconds of startup delay [30, 32, 103].

5.3.2 Evaluation Metrics

The performance of the proposed overlay management strategy is evaluated using metrics primarily related to QoS and upload capacity utilization. We also assess the stream delivery load of servers since it affects the QoS and affected by the upload capacity utilization of peers. The main QoS parameters for evaluation are startup delay and streaming quality. In addition, the stability of peers, upload contribution of peers, and early departure rate are used to understand the impact of QoS. We define the metrics used for evaluation below.

- **Upload capacity utilization:** It is the ratio of total number of chunks uploaded by peers to the maximum number that could have been uploaded.
- **Startup delay:** It is the duration for which peers wait to start playback after joining the system.
- **Streaming quality:** It is the percentage of chunks delivered to a peer before the playback deadline compared to the total number that should have been received for a desired streaming rate.
- **Server stream delivery load:** It is the percentage of peers, not directly connected to the server, that retrieve chunks from the CDN server either for quick startup or for better quality.

- **Stream recovery time:** It is the duration for which peers experience degradation in streaming quality before recovering the desired quality.
- **Peer stability:** It is the ratio of actual session duration to the maximum session duration of the peer. The maximum session duration is the time span between the peer joining time and the end of streaming session.
- **Early departure rate:** It is the percentage of peers leaving before the end of the session due to unacceptable startup delay and/or streaming quality.
- **Peer upload contribution:** It is the ratio of total upload capacity contributed by peers to the total upload capacity of the system.

5.3.3 Results and Discussions

First, we examine the upload capacity utilization of peers and then discuss its impact on QoS as well as the stream delivery load of servers. Next, we examine the impact of QoS on upload contribution and stability of peers.

Fig. 5.4 shows the upload capacity utilization of peers with the proposed strategy, PROSE and LiveSky. The proposed strategy exhibits better upload capacity utilization as compared to PROSE and LiveSky, because it organizes peers in the overlay considering their serviceability. The peers with higher upload capacities and longer session duration are used to become part of extended CDN tree and peer partners, whereas others are used to create virtual sources. On the other hand, PROSE and LiveSky do not arrange peers in the overlay so that they utilize upload capacity of all the peers. The distribution of peers among sub-overlays also results in better utilization in the proposed strategy. No such mechanism exists in PROSE and LiveSky.

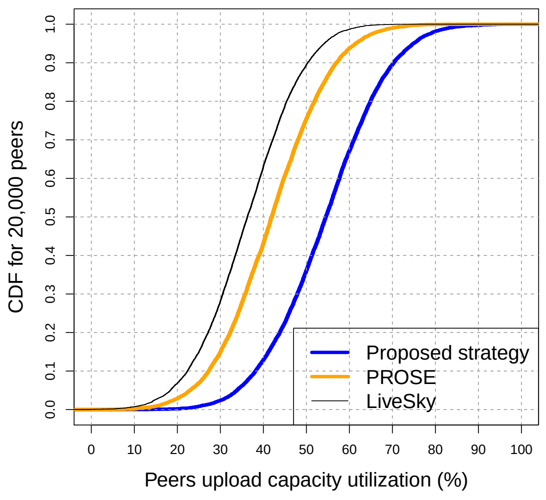


Figure 5.4: Upload capacity utilization of peers

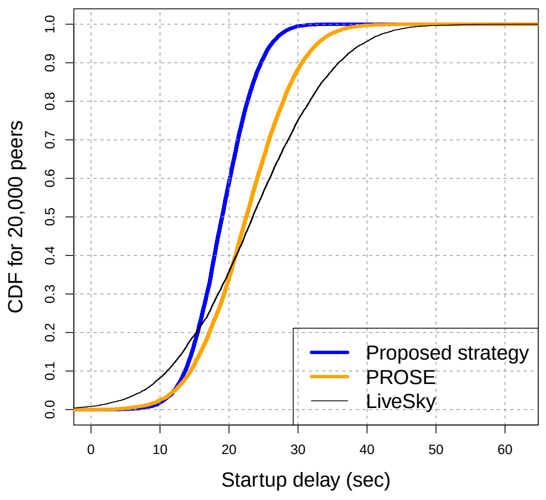


Figure 5.5: Startup delay experienced by peers

Fig. 5.5 and Fig. 5.6 show the startup delay and streaming quality perceived by the peers, respectively. The startup delay increases with the number of peers due to increase in stream delivery load of CDN servers and existing peers. It is also observed that when the number of peers is smaller, the startup delay of LiveSky is better because, the CDN servers initially provide startup chunks to new peers. In PROSE and the proposed strategy new peers receive startup chunks from other existing peers of the system. The proposed strategy still exhibits lower startup delay than PROSE because, it creates a dedicated virtual source for each new peer to provide startup chunks to the new peer.

Fig. 5.5 also shows the proposed strategy exhibits lower startup delay when a large number of peers join the system. LiveSky exhibits higher startup delay because all the new peers get startup chunks from the CDN server that causes overload. In PROSE, the existing peers do not provide the desired streaming quality to new peers during startup, because the CDN server suggests peers randomly without considering upload capacity and data availability in the buffer. In the proposed strategy, the startup chunks are served by the virtual sources to avoid the CDN server overloading. The selection of CDN servers considering the fraction of peers experiencing higher joining delay and poorer streaming quality also ensured quick startup with desired streaming quality.

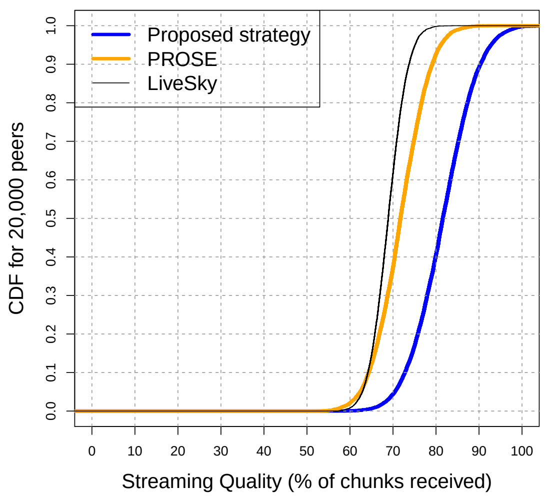


Figure 5.6: Streaming quality experienced by peers

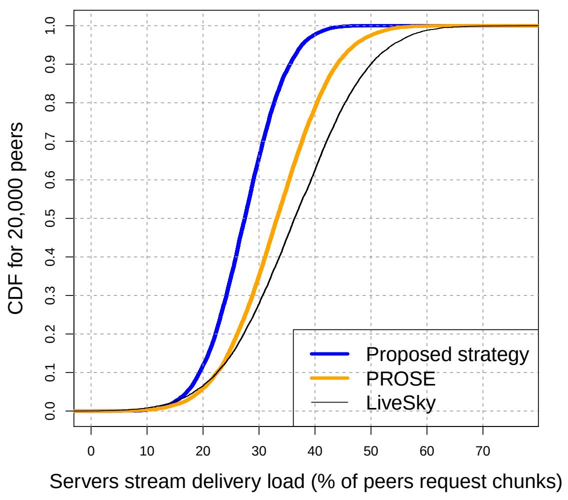


Figure 5.7: Stream delivery load of CDN servers

Fig. 5.6 compares the streaming quality perceived by the peers with the proposed strategy, PROSE and LiveSky. It is observed that PROSE gives better streaming quality than LiveSky because, CDN servers pro-actively forward the stream to peers with high upload capacity. The proposed strategy exhibits better streaming quality than both because it generates more high capacity seeders in the overlay (with extended CDN tree) in addition to what is done in PROSE. It also selects peers considering their serviceability whereas, partners are selected randomly in LiveSky and PROSE. The proposed strategy also prevents quality degradation by getting missing chunks from the backup partners and topology adaptation to re-select partners that give better streaming quality. In LiveSky and PROSE, the peers depend on only CDN servers that increases the stream delivery load on the CDN servers further. It is found that the proposed strategy improves perceived streaming quality by efficiently utilizing the upload capacities of peers.

To study the improvement in QoS due to better upload capacity utilization of peers, we measured the stream delivery load of CDN servers, which is reported in Fig. 5.7. It is evident from the results that stream delivery load of CDN servers is significantly reduced using proposed strategy as compared to LiveSky and PROSE. This is because responsibility of providing startup chunks is delegated to peers by creating virtual sources, whereas peers in PROSE and LiveSky receive startup from the CDN server. Retrieval of missing chunks from CDN servers due to stream quality degradation is reduced with serviceability-based peer selection, backup partners and topology adaptation. Additionally, the upload and download capacity allocation mechanism of peers and CDN server selection strategy also ensure to evenly distribute the stream deliver load of servers based on supply-demand ratio of upload capacities in the overlay.

The departure of peers due to unacceptable startup delay is termed joining failure. Fig. 5.8 shows percentage of peers departing (due to joining failure). It is observed that this follows similar trend as startup delay because lower the startup delay experienced by peers, lower is the departure of peers. Fig. 5.9 compares the percentage of peers leaving due to stream quality degradation in all the three strategies. It is observed that the proposed strategy results in lower departure rate of peers due to stream quality degradation, following the exactly opposite trend as streaming quality experienced by peers.

It is also found that percentage of peers leaving due to joining failure is higher than that due to quality degradation with the proposed strategy. This is because when peers do not provide desired streaming quality, the backup partners are used

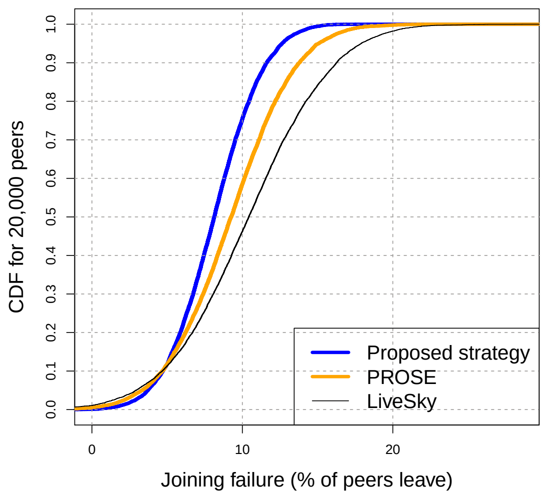


Figure 5.8: Percentage of peers facing joining failure

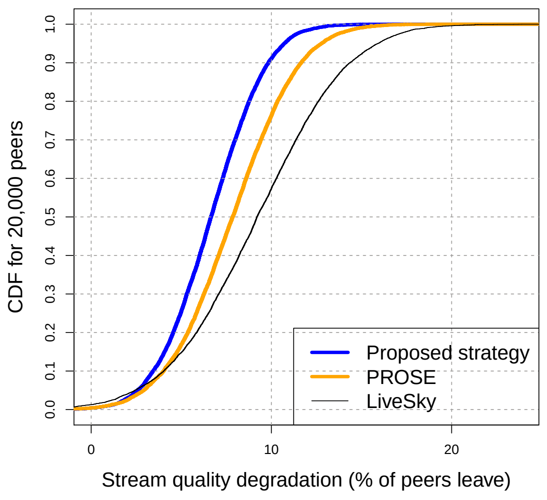


Figure 5.9: Peer departure due to stream quality degradation

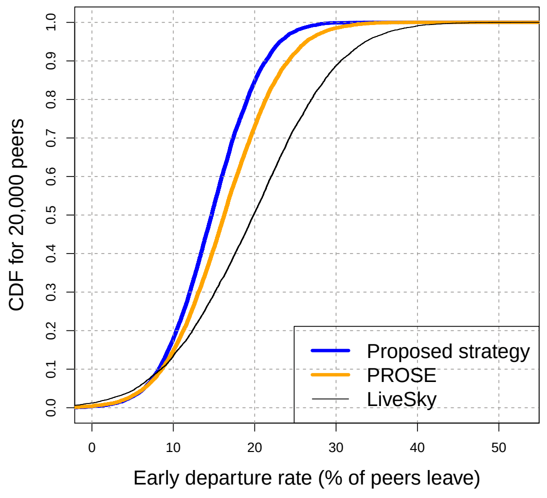


Figure 5.10: Early departure rate of peers due to poor QoS

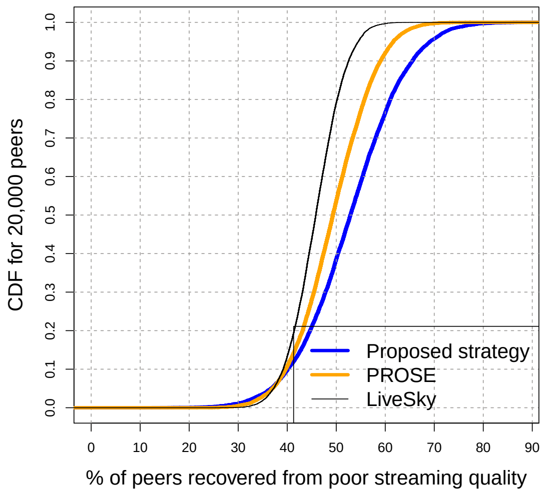


Figure 5.11: Percentage of peers recovered from poor streaming quality

to retrieve chunks and topology adaptation replaces the partners. Moreover, lower

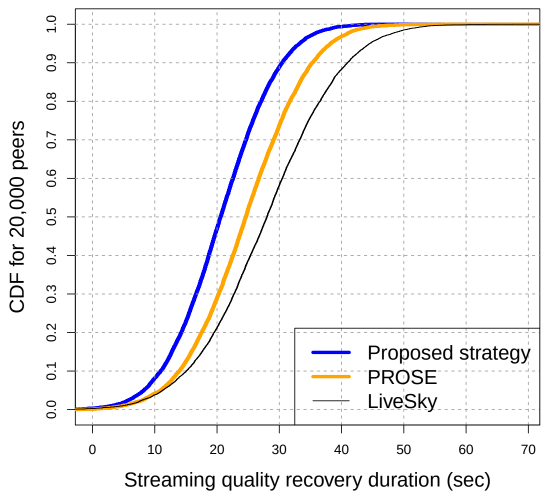
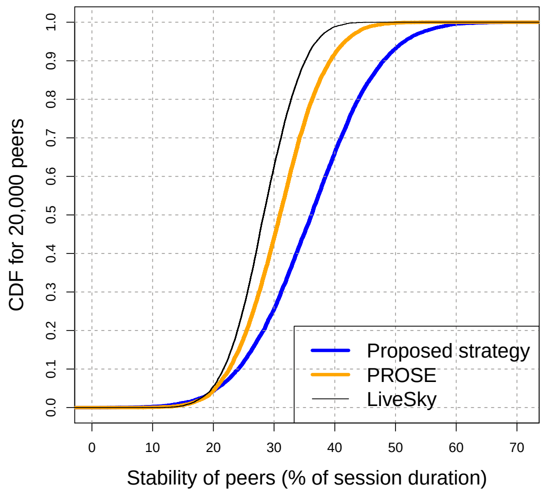


Figure 5.12: Time taken for recovery from degradation of streaming quality

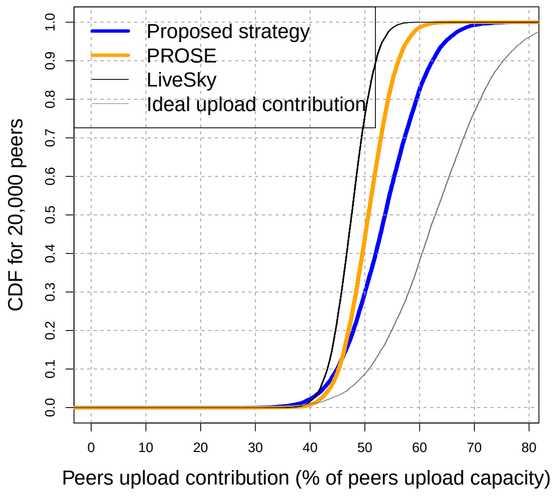
stream delivery load of CDN servers helps in reducing early departure rate of peers. We see that the proposed strategy results in significant reduction in early departure rate of peers, which is reported in Fig. 5.10.

To substantiate these claims, we show the percentage of peers that recovered from unacceptable streaming quality and the time taken for recovery in Fig. 5.11 and Fig. 5.12, respectively. We see in Fig. 5.11 that an additional 12% peers recover from unacceptable streaming quality with the proposed strategy compared to PROSE and LiveSky. Fig. 5.12 shows that the time taken by peers to recover the desired streaming quality is also smallest with our strategy. These improvements are due to lower stream delivery load of CDN servers, quick service from backup partners and topology adaptation.

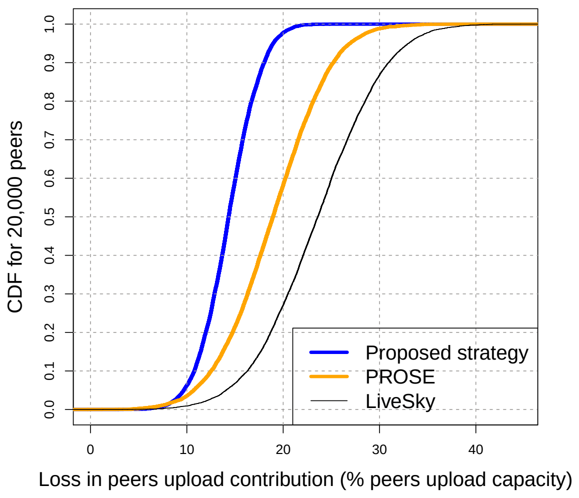
We also show the impact of QoS perceived on the stability and upload contribution of peers in Fig. 5.13. Fig. 5.13a shows that the stability of peers is higher with the proposed strategy because, peers stay longer due to improved QoS. This also improves the total upload contribution of peers in the system. Fig. 5.13b



(a) CDF of stability of peers



(b) CDF of upload contribution of peers



(c) CDF of loss in upload contribution

Figure 5.13: Comparing stability, loss in upload capacity contribution of peers due to poor QoS and peer departure

compares the upload capacity contribution of peers with the ideal contribution of peers, which is the total upload capacity contributed if the peers stay till the end.

It is clear that the upload contribution of peers is higher as well as closer to the ideal contribution. We also found that upload capacity contributed by the tree peers is higher than that by the mesh peers because, tree peers get better QoS being closer to the CDN servers and they also stay longer. Fig. 5.13c compares the loss in upload contribution of peers due to poor QoS and departure of peers. It is observed that proposed strategy shows a loss of approximately 22%, whereas PROSE and LiveSky show a loss of approximately 32% and 38%, respectively. Therefore, both the stability of peers and upload contribution of peers are improved using the proposed strategy.

Overall, Table 5.4 compares the performance of the proposed strategy with existing strategies. It can be seen that proposed strategy improves upload capacity utilization by 30% with higher value of standard deviation. This resulted in reducing startup delay by 20% and improving the streaming quality by 25%. It also reduces the stream delivery load of servers by 20%. We have also observed that proposed strategy takes 15% lesser time in recovering streaming quality of peers during churn. The better QoS also improved the stability of peers by 20%. Increase in stability of peers is further substantiated by measuring early departure rate of peers, which is found to be lower by about 25%. Better stability of peers also increased the upload contribution of peers to the system by about 15%.

5.4 Summary

In this work, we proposed an overlay management strategy for peer-assisted live streaming systems which exploited the serviceability of peers to improve QoS and offload CDN servers. In the proposed strategy, the peers are arranged to form a hybrid tree-mesh overlay topology based on their serviceability. It utilizes peers

Table 5.4: Comparing performance of the proposed strategy and existing strategies

Evaluation Metrics	**Proposed Strategy Mean (SD)**	**LiveSky Mean (SD)**	**PROSE Mean (SD)**	**Improvement %**
Upload capacity utilization	54.48 (12.5)	36.26 (10.8)	42.12 (11.7)	**30%**
Startup delay	18.58 (4.3)	23.44 (9.7)	22.41 (6.3)	**20%**
Streaming quality	83.94 (6.7)	66.95 (3.7)	70.7 (5.6)	**25%**
Server stream delivery load	27.41 (6.2)	36.45 (10.5)	33.19 (8.4)	**20%**
Stream recovery time	21.68 (7.6)	26.8 (10.1)	24.5 (8.3)	**15%**
Peer stability	36.12 (9.3)	28.31 (5.2)	30.77 (6.3)	**20 %**
Early departure rate	14.76 (5.1)	19.6 (8.6)	16.36 (6.1)	**25%**
Peer upload contribution	54.64 (6.7)	47.4 (3.4)	50.53 (4.2)	**15%**

with heterogeneous upload capacity as well as session duration to provide streaming services. The peers with higher upload capacity are added to the extended CDN tree which generated stable and high capacity seeders. The peers with lower upload capacity and/or shorter session duration are used to create virtual sources, that provide startup chunks to new peers. This reduced startup delay experienced by peers and offloaded CDN servers from providing startup chunks to new peers. The mesh peers selected peers with long session duration and high upload capacity as their parent partners to improve streaming quality. The topology adaptation strategy used by mesh peers helped in maintaining QoS during churn and offloading CDN servers from providing missing chunks. The CDN server selection strategy and bandwidth allocation mechanism of peers also balanced the stream delivery load of CDN servers and peers in the overlay and improved QoS.

The performance of proposed strategy is evaluated by comparing it with PROSE and LiveSky. Results show that the proposed strategy improved startup delay and streaming quality by 20% and 25%, respectively. It also improved the stability of peers by 20% leading to an improvement in their upload contribution by approximately 15%. The utilization of upload capacity of peers is enhanced by 30%, which also reduced the stream delivery load of CDN servers by approximately 20%.

Considering the better upload contribution and upload capacity utilization of peers with the proposed strategy, future research could focus on optimizing number of CDN servers while maintaining QoS during peer churn and flash crowd.

Chapter 6

Conclusions and Future Work

The primary objective of the work in this thesis was to improve the QoS of peer-assisted live streaming systems. The major challenge in achieving this objective was to deal with heterogeneity in bandwidth and lifetime of peers, while utilizing their resources to provide streaming service. We addressed this issue by proposing better overlay management strategies.

6.1 Conclusions

First, we explored the strengths and limitations of different overlay topologies using trace-based simulations. We studied the impact of flash crowd, peer churn and heterogeneous upload capacity of peers on the QoS and stability of peers. We found that the multi-tree overlay has highest startup delay due to longer peer selection time, whereas in the mesh and hybrid overlays, the startup delay is higher due to unavailability of chunks and free upload capacity. The playback delay is highest in the mesh overlay due to per-hop latency, but the streaming quality, upload capacity utilization, and resilience are better during peer churn. In the hybrid overlay, the mesh sub-overlay leads to larger playback delay, while the tree sub-overlay suffers from poor resilience. We concluded that the playback delay of mesh overlays needs to be minimized and streaming quality of hybrid overlays needs to be improved.

Towards this, we proposed a peer selection strategy for mesh overlays to minimize the playback delay. In the proposed strategy, peer selection is done at three different stages of overlay construction. The parameters considered during partner selection are upload capacity, propagation delay, buffering level and chunk buffering duration. We evaluated the performance of our proposed strategy to show that the playback delay is significantly minimized while handling flash crowd and peer churn. It also reduced the startup delay marginally, which needs to be addressed further.

Finally, we addressed the problem of improving streaming quality, startup delay, and upload capacity utilization considering heterogeneous lifetime and bandwidth of peers. We proposed an overlay management strategy for hybrid tree-mesh overlays that helped in organizing peers in the overlay based on their serviceability. As a part of the overlay management, we proposed a CDN server selection strategy and a bandwidth allocation mechanism for peers to evenly distribute the load on the servers and peers. The strategy builds a hybrid tree-mesh overlay with stable and high upload capacity seeders. We introduced the notion of virtual sources to provide quick startup to new peers and a serviceability-based peer selection strategy for better streaming quality. The topology is also continuously adapted to maintain QoS during peer churn. We compared the performance of the proposed strategy with the existing systems and found that it improved streaming quality and startup delay of peers significantly. The improvement in QoS also resulted in better stability and upload contribution of peers. The proposed strategy also enhanced the utilization of upload capacity of peers significantly, leading to the offloading of CDN servers.

6.2 Future Work

The work presented in this thesis can be extended in various directions and leaves ample scope for future research in this area. We present a few immediate extensions to this work below.

- In our work, we assumed that the aggregate bandwidth of a peer allocated to the system remains constant. In future, the impact of dynamic aggregate bandwidth of peers on the QoS and their upload capacity utilization can be accounted for in the proposed strategies.
- Our work improves the bandwidth contribution and upload capacity utilization of peers in peer-assisted live streaming systems. Taking into account these improvements, future work could focus on optimizing the number of CDN servers that needs to be deployed to provide a guaranteed QoS while handling flash crowd and peer churn.
- In our work, we considered peer-assisted systems with traditional CDNs. Recently, federated content deliver networks (Federated CDNs or Telco-CDNs) are proposed to offload the traffic of core network and to increase the availability of video data closer to the users. The overlay management strategies proposed in this work can be extended to create a peer-assisted Telco-CDN system for live streaming services that considers the constraints and policies associated with the inter-ISP traffic management.

Bibliography

[1] Kostya, "Skype does away with random supernodes," http://expertmiami.blogspot.com/2012/05/skype-does-away-with-random-supernodes.html, 2012.

[2] M. KIRKPATRICK, "The numbers are in, live video online is blowing up," http://www.readwriteweb.com/archives/live_video_big.php, 2008.

[3] L. GANNES, "The Obama Inauguration Live Stream Stats," http://newteevee.com/2009/01/20/the-obama-inauguration-live-stream-stats/, 2009.

[4] "Cisco Visual Networking Index: Forecast and Methodology, 2016 - 2021," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, 15 September, 2017.

[5] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec, "Peer-assisted Content Distribution in Akamai Netsession," in *Proc. of Internet Measurement Conference*, 2013, pp. 31–42.

[6] Y. Kim, Y. Kim, H. Yoon, and I. Yeom, "Peer-assisted Multimedia Delivery Using Periodic Multicast," *Information Sciences*, vol. 298, no. C, pp. 425–446, 2015.

[7] M. Elkotob and K. Andersson, “Challenges and Opportunities in Content Distribution Networks: A Case Study,” in *Proc. of IEEE Globecom Workshops*, 2012, pp. 1021–1026.

[8] S. Okada and S. Fujita, “P2P Overlay for CDN-P2P Being Aware of the Upload Capacity of Participants,” in *Proc. of IEEE International Conference on Parallel and Distributed Systems*, 2014, pp. 823–828.

[9] P. Wendell and M. J. Freedman, “Going viral: Flash crowds in an open cdn,” in *Proc. of ACM SIGCOMM Internet Measurement Conference*, 2011, pp. 549–558.

[10] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, “A Case for a Coordinated Internet Video Control Plane,” *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 359–370, 2012.

[11] J. Rckert, T. Knierim, and D. Hausheer, “Clubbing with the Peers: A Measurement Study of BitTorrent Live,” in *Proc. of IEEE International Conference on Peer-to-Peer Computing*, 2014, pp. 1–10.

[12] J. Simmons, “Broadcasting the World Cup and Wimbledon in UHD - the full story,” http://www.bbc.co.uk/blogs/internet/entries/5c5b8f80-891d-4b51-babd-8814c1511b4e, 19 July 2018.

[13] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, and U. C. Berkeley, “Understanding the Impact of Video Quality on User Engagement,” in *Proc. of SIGCOMM*, 2011, pp. 362–373.

[14] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, “LiveSky: Enhancing CDN with P2P,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 6, no. 3, pp. 16:1–16:19, 2010.

[15] M. Garmehi and M. Analoui, "Envy-Free Resource Allocation and Request Routing in Hybrid CDN-P2P Networks," *Journal of Network and Systems Management*, vol. 24, no. 4, pp. 884–915, 2016.

[16] B. Li, Z. Wang, J. Liu, and W. Zhu, "Two Decades of Internet Video Streaming: A Retrospective View," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 9, no. 1s, pp. 33:1–33:20, 2013.

[17] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on Peer-assisted Content Delivery Networks," *Computer Networks*, vol. 116, no. C, pp. 79–95, 2017.

[18] J. Rckert, B. Richerzhagen, E. Lidanski, R. Steinmetz, and D. Hausheer, "TOPT: Supporting Flash Crowd Events in Hybrid Overlay-based Live Streaming," in *Proc. of IFIP Networking Conference*, 2015, pp. 1–9.

[19] X. Zhang and H. Hassanein, "A Survey of Peer-to-peer Live Video Streaming Schemes - An Algorithmic Perspective," *Computer Networks*, vol. 56, no. 15, pp. 3548–3579, 2012.

[20] C. Huang, J. Li, and K. W. Ross, "Can Internet Video-on-demand Be Profitable?" *SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 133–144, 2007.

[21] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, 2007.

[22] B. Li, S. Xie, Y. Qu, G. Y. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the New Coolstreaming: Principles, Measurements and Performance Implications," in *Proc. of IEEE INFOCOM*, 2008, pp. 1031–1039.

[23] "SopCast," http://www.sopcast.com/, 2017.

[24] X. Lou and K. Hwang, "Quality of Data Delivery in Peer-to-Peer Video Streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 8, no. 1S, pp. 12:1–12:23, 2012.

[25] B. Li, S. Xie, G. Y. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang, "An Empirical Study of the Coolstreaming+ System," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1627–1639, 2007.

[26] D. Karamshuk, N. Sastry, A. Secker, and J. Chandaria, "ISP-friendly Peer-assisted On-demand Streaming of Long Duration Content in BBC iPlayer," in *Proc. of IEEE INFOCOM*, 2015, pp. 289–297.

[27] M. Afergan, T. Leighton, and J. Parikh, "Hybrid Content Delivery Network (CDN) and Peer-to-Peer (P2P) Network," *US8332484B2*, 2012.

[28] A. Balachandran, V. Sekar, A. Akella, and S. Seshan, "Analyzing the Potential Benefits of CDN Augmentation Strategies for Internet Video Workloads," in *Proc. of the Internet Measurement Conference*, 2013, pp. 43–56.

[29] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding Hybrid CDN-P2P: Why Limelight Needs Its Own Red Swoosh," in *Proc.of NOSSDAV*, 2008, pp. 75–80.

[30] K. Park, D. Chang, J. Kim, W. Yoon, and T. Kwon, "An Analysis of User Dynamics in P2P Live Streaming Services," in *Proc. of IEEE International Conference on Communications (ICC)*, 2010, pp. 1–6.

[31] Z. Liu, C. Wu, B. Li, and S. Zhao, "Why are Peers Less Stable in Unpopular P2P Streaming Channels?" in *Proc. of IFIP-TC 6 Networking Conference*, 2009, pp. 274–286.

[32] S. Agarwal, J. P. Singh, A. Mavlankar, P. Baccichet, and B. Girod, "Performance and Quality-of-Service Analysis of a Live P2P Video Multicast Session on the Internet," in *Proc. of Interntional Workshop on Quality of Service*, 2008, pp. 11–19.

[33] Y. Tang, L.-F. Sun, K.-Y. Zhang, S.-Q. Yang, and Y.-Z. Zhong, "Longer, Better: On Extending User Online Duration to Improve Quality of Streaming Service in P2P Networks," in *Proc. of IEEE International Conference on Multimedia and Expo*, 2007, pp. 2158–2161.

[34] I. Ullah, G. Doyen, and D. Gaiti, "Towards User-aware Peer-to-Peer Live Video Streaming Systems," in *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, 2013, pp. 920–926.

[35] S. A. Baset and H. G. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," in *Proc. of IEEE INFOCOM*, 2006, pp. 1–11.

[36] N. Zeilemaker, M. Capotă, A. Bakker, and J. Pouwelse, "Tribler: P2P Media Search and Sharing," in *Proc. of the 19th ACM International Conference on Multimedia*, 2011, pp. 739–742.

[37] J. Mercier, "Skype Numerology:50 million concurrent users online!" http://skypenumerology.blogspot.com/2013/01/50-million-concurrent-users-online.html, 2013.

[38] A. Passarella, "A Survey on Content-centric Technologies for the Current Internet: CDN and P2P Solutions," *Computer Communications*, vol. 35, no. 1, pp. 1 – 32, 2012.

[39] S. Guha, N. Daswani, and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *Proc. of International Workshop on Peer-to-Peer Systems*, 2006, pp. 1–6.

[40] Z. Whittaker, “Skype Ditched Peer-to-peer Supernodes for Scalability, Not Surveillance,” https://www.zdnet.com/article/skype-ditched-peer-to-peer-supernodes-for-scalability-not-surveillance, 2013.

[41] N. Unuth, “Skype Changes From P2P to Client-Server Model,” https://www.lifewire.com/skype-changes-from-p2p-3426522, 2019.

[42] S. A. Baset, “Protocols and System Design, Reliability, and Energy Efficiency in Peer-to-Peer Communication Systems,” *Columbia University Computer Science Technical Reports*, 2011.

[43] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, “TRIBLER: A Social-based Peer-to-peer System,” *Concurrency and Computation: Recent Advances in Peer-to-Peer Systems and Security*, vol. 20, no. 2, pp. 127–138, 2008.

[44] Tribler, “HTTP and P2P Download Merge: Zero-delay P2P Video Streaming,” https://www.tribler.org/HTTP2PDownload/, 2016.

[45] A. Rose, “Evolution of the BBC iPlayer,” *EBU Technical Review*, pp. 1–14, 2008.

[46] ——, “Introducing BBC iPlayer Desktop for Mac, Linux and PC,” https://www.bbc.co.uk/blogs/bbcinternet/2008/12/introducing_iplayer_deskto.html, 2008.

[47] N. Lanxon, “iPlayer uncovered: What powers the BBC’s epic creation?” https://www.cnet.com/news/iplayer-uncovered-what-powers-the-bbcs-epic-creation/, 2009.

[48] M. Castro, P. Druschel, A. Rowstron, and A.-m. Kermarrec, "SplitStream : High-Bandwidth Multicast in Cooperative Environments," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 298–313, 2003.

[49] O. Abboud, K. Pussep, A. Kovacevic, K. Mohr, S. Kaune, and R. Steinmetz, "Enabling Resilient P2P Video Streaming: Survey and Analysis," *Multimedia Systems*, vol. 17, no. 3, pp. 177–197, 2011.

[50] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 379–392, 2010.

[51] D. Ren, Y. H. Li, and S. . G. Chan, "Fast-Mesh : A Low-Delay High-Bandwidth Mesh for Peer-to-Peer Live Streaming," *IEEE Transactions on Multimedia*, vol. 11, no. 8, pp. 1446–1456, 2009.

[52] A. T. Nguyen, B. Li, and F. Eliassen, "Quality and Context-Aware Neighbor Selection for Layered Peer-to-Peer Streaming," in *Proc. of IEEE International Conference on Communications*, 2010, pp. 1–6.

[53] L. Wang, D. Zhang, and H. Yang, "Qos-Awareness Variable Neighbor Selection for Mesh-based P2P Live Streaming System," in *Proc. of IEEE International Conference on Information Science and Technology*, 2013, pp. 1197–1201.

[54] A. Paula, E. Leonardi, M. Mellia, and M. Meo, "Chunk Distribution in Mesh-Based Large-Scale P2P Streaming Systems: A Fluid Approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 3, pp. 451–463, 2011.

[55] M. L. Merani, L. Natali, and C. Barcellona, "An IP-TV P2P Streaming System that Improves the Viewing Quality and Confines the Startup Delay of Regular

Audience," *Peer-to-Peer Networking and Applications*, vol. 9, no. 1, pp. 209–222, 2016.

[56] F. Wang, J. Liu, and Y. Xiong, "On Node Stability and Organization in Peer-to-Peer Video Streaming Systems," *IEEE Systems Journal*, vol. 5, no. 4, pp. 440–450, 2011.

[57] Z.-H. Lv, L.-J. Chen, J. Wu, D. Deng, S.-J. Huang, and Y. Huang, "PROSE: Proactive, Selective CDN Participation for P2P Streaming," *Journal of Computer Science and Technology*, vol. 28, no. 3, pp. 540–552, 2013.

[58] H. Shen, Z. Li, Y. Lin, and J. Li, "SocialTube: P2P-Assisted Video Sharing in Online Social Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2428–2440, 2014.

[59] H. Deng and J. Xu, "CorePeer: A P2P Mechanism for Hybrid CDN-P2P Architecture," in *Proc. of International Conference on Web-Age Information Management*, 2013, pp. 278–286.

[60] C. Hammami, I. Jemili, A. Gazdar, A. Belghith, and M. Mosbah, "HLPSP: A Hybrid Live P2P Streaming Protocol," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 3, pp. 1035–1056, 2015.

[61] Z. H. Lu, X. H. Gao, S. J. Huang, and Y. Huang, "Scalable and Reliable Live Streaming Service through Coordinating CDN and P2P," in *Proc. of IEEE International Conference on Parallel and Distributed Systems*, 2011, pp. 581–588.

[62] Z. Lu, Y. Wang, and Y. Yang, "An Analysis and Comparison of CDN-P2P-hybrid Content Delivery System and Model," *Journal of Communications*, vol. 7, no. 3, pp. 232–245, 2012.

[63] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," in *Proc. of NOSSDAV*, 2002, pp. 177–186.

[64] V. Venkataraman, K. Yoshida, and P. Francis, "ChunkySpread: Heterogeneous Unstructured Tree-based Peer-to-Peer Multicast," in *Proc. of IEEE International Conference on Network Protocols*, 2006, pp. 2–11.

[65] Q. Huang, H. Jin, and X. Liao, "P2P Live Streaming with Tree-Mesh based Hybrid Overlay," in *Proc. of International Conference on Parallel Processing Workshops*, 2007, pp. 55–55.

[66] S. Awiphan, Z. Su, and J. Katto, "ToMo: A Two-Layer Mesh/Tree Structure for Live Streaming in P2P Overlay Network," in *Proc. of IEEE Consumer Communications and Networking Conference*, 2010, pp. 1–5.

[67] S. Birrer and F. E. Bustamante, "A Comparison of Resilient Overlay Multicast Approaches," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1695–1705, 2007.

[68] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches," in *Proc. of IEEE INFOCOM*, 2007, pp. 1424–1432.

[69] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru, "Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective," in *Proc. of IEEE Conference on Local Computer Networks*, 2008, pp. 20–27.

[70] X. Zhang and H. Hassanein, "Understanding the Impact of Neighboring Strategy in Peer-to-Peer Multimedia Streaming Applications," *Computer Communications*, vol. 35, no. 15, pp. 1893 – 1901, 2012.

[71] B. Biskupski, M. Schiely, P. Felber, and R. Meier, "Tree-Based Analysis of Mesh Overlays for Peer-to-Peer Streaming," in *Proc. of IFIP: Distributed Applications and Interoperable Systems*, 2008, pp. 126–139.

[72] A. Ouali, B. Kerherve, and B. Jaumard, "Toward Improving Scheduling Strategies in Pull-based Live P2P Streaming Systems," in *Proc. of IEEE Consumer Communications and Networking Conference*, 2009, pp. 1–5.

[73] Z. Ouyang, M. Wang, L. Xu, and B. Ramamurthy, "On Providing Bounded Delay Service to Subscribers in P2P Live Streaming Systems," in *Proc. of IEEE Globecom*, 2012, pp. 2012–2017.

[74] H. Wu, J. Liu, H. Jiang, Y. Sun, J. Li, and Z. Li, "Bandwidth-Aware Peer Selection for P2P Live Streaming Systems Under Flash Crowds," in *Proc. of IEEE International Performance Computing and Communications Conference*, 2012, pp. 360–367.

[75] S. Li, J. Zhao, and X. Wang, "Elite : Differentiating the Playback Lag for Peer-Assisted Live Video Streaming," in *Proc. of IEEE International Workshop on Quality of Service*, 2012, pp. 1–9.

[76] X. Wu, X. Chen, and H. Wang, "An Effective Scheme for Performance Improvement of P2P Live Streaming Systems," *Journal of Networks*, vol. 9, no. 4, pp. 1067–1073, 2014.

[77] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic Live Streaming: Optimal Performance Trade-Offs," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1, pp. 325–336, 2008.

[78] R. Birke, E. Leonardi, M. Mellia, A. Bakay, T. Szemethy, C. Kiraly, R. L. Cigno, F. Mathieu, L. Muscariello, S. Niccolini *et al.*, "Architecture of a

Network-aware P2P-TV Application: the NAPA-WINE Approach," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 154–163, 2011.

[79] A. P. C. da Silva, E. Leonardi, M. Mellia, and M. Meo, "A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems," in *Proc. of International Conference on Peer-to-Peer Computing*, 2008, pp. 279–288.

[80] I. Chatzidrossos, G. Dn, and V. Fodor, "Delay and Playout Probability Trade-off in Mesh-based Peer-to-Peer Streaming with Delayed Buffer Map Updates," *Peer-to-Peer Networking and Applications*, vol. 3, no. 3, pp. 208–221, 2010.

[81] Y. Zhou, D.-M. Chiu, and J. C. Lui, "A Simple Model for Chunk-Scheduling Strategies in P2P Streaming," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 42–54, 2011.

[82] S. Heidari, N. Dehghani, M. S. Talebi, and A. Khonsari, "Exploring Playback Continuity and Delay Trade-off in Peer-to-Peer Streaming," in *Proc. of IEEE Symposium on Computers and Communications*, 2012, pp. 518–523.

[83] Y. Liu, "Delay Bounds of Chunk-Based Peer-to-Peer Video Streaming," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1195–1206, 2010.

[84] A. P. C. da Silva, E. Leonardi, M. Mellia, and M. Meo, "Exploiting Heterogeneity in P2P Video Streaming," *IEEE Transactions on Computers*, vol. 60, no. 5, pp. 667–679, 2011.

[85] Z. Liu, C. Wu, B. Li, and S. Zhao, "Distilling Superior Peers in Large-scale P2P Streaming Systems," in *Proc. of IEEE INFOCOM*, 2009, pp. 82–90.

[86] C. Hu, M. Chen, C. Xing, and B. Xu, "EUE Principle of Resource Scheduling for Live Streaming Systems Underlying CDN-P2P Hybrid Architecture," *Peer-to-Peer Networking and Applications*, vol. 5, no. 4, pp. 312–322, 2012.

[87] T. T. Ha, J. Kim, and J. Nam, "Design and Deployment of Low-Delay Hybrid CDN-P2P Architecture for Live Video Streaming Over the Web," *Wireless Personal Communications*, vol. 94, no. 3, pp. 513–525, 2017.

[88] Z. Shen, J. Luo, R. Zimmermann, and A. V. Vasilakos, "Peer-to-Peer Media Streaming: Insights and New Developments," *Proceedings of the IEEE*, vol. 99, no. 12, pp. 2089–2109, 2011.

[89] N. Ramzan, H. Park, and E. Izquierdo, "Video streaming over P2P networks: Challenges and opportunities," *Signal Processing: Image Communication*, vol. 27, no. 5, pp. 401 – 411, 2012.

[90] Y. Liu, Y. Guo, and C. Liang, "A Survey on Peer-to-peer Video Streaming Systems," *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, 2008.

[91] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer Receiver-driven Mesh-based Streaming," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1052–1065, 2009.

[92] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating Trees from Overlay Multicast," in *Proc. of International Conference on Peer-to-Peer Systems*, 2005, pp. 127–140.

[93] X. Zhang, J. Liu, B. Li, and Y. S. P. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," in *Proc. of INFOCOM*, 2005, pp. 2102–2111.

[94] H. Byun and M. Lee, "HyPO: A Peer-to-Peer based Hybrid Overlay Structure," in *Proc. of International Conference on Advanced Communication Technology*, 2009, pp. 840–844.

[95] B. U. Maheswari and T. S. B. Sudarshan, "Reputation Based Mesh-Tree-Mesh Cluster Hybrid Architecture for P2P Live Streaming," in *Proc. of International Conference on Devices, Circuits and Systems*, 2016, pp. 240–243.

[96] M. Wichtlhuber, B. Richerzhagen, J. Rckert, and D. Hausheer, "TRANSIT: Supporting Transitions in Peer-to-Peer Live Video Streaming," in *Proc. of IFIP Networking Conference*, 2014, pp. 1–9.

[97] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proc. of Simutools*, 2008, pp. 1–10.

[98] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proc. of IEEE Global Internet Symposium in conjunction with INFOCOM*, 2007, pp. 79–84.

[99] T. Steinbach, H. D. Kenfack, F. Korf, and T. C. Schmidt, "An Extension of the OMNeT++ INET Framework for Simulating Real-time Ethernet with High Accuracy," in *Proc. of International Conference on Simulation Tools and Techniques*, 2011, pp. 375–382.

[100] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The Feasibility of Supporting Large-scale Live Streaming Applications with Dynamic Application End-points," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 107–120, 2004.

[101] H. Hu, Y. Guo, and Y. Liu, "Peer-to-Peer Streaming of Layered Video: Efficiency, Fairness and Incentive," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 8, pp. 1013–1026, 2011.

[102] C. Vassilakis and I. Stavrakakis, "Minimizing Node Churn in Peer-to-peer Streaming," *Computer Communications*, vol. 33, no. 14, pp. 1598–1614, 2010.

[103] S. Xie, G. Y. Keung, and B. Li, "A Measurement of a Large-scale Peer-to-Peer Live Video Streaming System," in *Proc. of International Conference on Parallel Processing Workshops*, 2007, pp. 57–57.

[104] B. Zhang, G. Kreitz, M. Isaksson, J. Ubillos, G. Urdaneta, J. A. Pouwelse, D. Epema, and A. Trace, "Understanding User Behavior in Spotify," in *Proc. of IEEE INFOCOM*, 2013, pp. 220–224.

[105] I. Ullah, G. Doyen, G. Bonnet, and D. Gaiti, "A Survey and Synthesis of User Behavior Measurements in P2P Streaming Systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 3, pp. 734–749, 2012.

[106] G. Bonnet, I. Ullah, G. Doyen, L. Fillatre, D. Gaiti, and I. Nikiforov, "A Semi-Markovian Individual Model of Users for P2P Video Streaming Applications," in *Proc. of IFIP International Conference on New Technologies, Mobility and Security*, 2011, pp. 1–5.

[107] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet ," in *Proc. of ACM Internet Measurement Conference*, 2004, pp. 41–54.

[108] H. Schwarz and M. Wien, "The Scalable Video Coding Extension of the H. 264/AVC Standard," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 135–141, 2008.

Publications Related to Thesis

Journal Papers

- **Shilpa Budhkar** and Venkatesh Tamarapalli, “Delay Management in Mesh-Based P2P Live Streaming Using a Three-Stage Peer Selection Strategy”, *Journal of Network and Systems Management*, vol. 26, no. 2, pp. 401-425, 2018.
- **Shilpa Budhkar** and Venkatesh Tamarapalli, “An Overlay Management Strategy to Improve QoS in CDN-P2P Live Streaming Systems”, *Peer-to-Peer Networking and Applications*, pp. 1-17, 2019.

Conference Papers

- **Shilpa Budhkar** and Venkatesh Tamarapalli, “An Overlay Management Strategy to Improve Peer Stability in P2P Live Streaming Systems”, *10th IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 16, IISc Bangalore, India, November 2016.
- **Shilpa Budhkar** and Venkatesh Tamarapalli, “Two-Tier Peer Selection Strategy to Minimize Delay in P2P Live Streaming Systems”, *22nd IEEE National Conference on Communications (NCC)*, pp. 16, IIT Guwahati, India, March 2016.

Author's Biography

Shilpa Budhkar joined Ph.D. programme in the Department of Computer Science and Engineering at Indian Institute of Technology Guwahati, India in July 2010. Prior to joining PhD, she completed her Bachelor of Engineering (B.E.) degree with HONORS in Information Technology from Rajiv Gandhi Technological University, Bhopal, India in 2010. Currently, she is working as an Assistant Professor with the Department of Computer Science & Engineering and Information Technology, Jaypee Institute Of Information Technology, Noida, India. Her research interests include P2P overlay networks, Multimedia streaming over Internet, Content delivery networks (CDN) and Distributed Systems.

Contact Information

E-mail: shilpab.iitg@gmail.com, b.shilpa@iitg.ac.in

Permanent Address: Plot No.-508, C-sector, Sarva Dharm, Kolar Road,

Bhopal, Madhya Pradesh (M.P.), India-462042.