A QUANTITATIVE FRAMEWORK FOR CDN-BASED OVER-THE-TOP VIDEO STREAMING SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Abubakr O. Al-Abbasi

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Vaneet Aggarwal, Chair

School of Industrial Engineering

Dr. Xiaojun Lin

School of Electrical and Computer Engineering

Dr. Tian Lan

School of Electrical and Computer Engineering

Dr. Christopher Quinn

School of Electrical and Computer Engineering

Approved by:

Dr. Abhijit Deshmukh

Head of the Graduate Program

This thesis is dedicated to my beloved wife, parents, and my lovely daughter Lames and the new comer, my little kid, Hashim!

ACKNOWLEDGMENTS

"Praise be to God for guiding us to this. We would have never been guided if God had not guided us.", Quran 7:43.

I would like to express my deepest appreciation and gratitude to my doctoral advisor Professor Vaneet Aggarwal. I consider myself as a lucky person working with him. I have learned a lot, both in scientific and personal aspects. Actually, this work would not have been developed in this form without his guidance, scientific support, and fruitful discussions. I would like to thank Professor Xiaojun Lin, Professor Tian Lan and Professor Christopher Quinn for their invaluable criticism and fruitful feedback for my dissertation. I would like to show my appreciation to Professor Tian Lan for his comments, suggestions, and feedback which results in having two well established journal papers at IEEE Transaction on Cloud Computing and IEEE Transaction on Network Science and Management. I also want to thank Prof. Xiaojun Lin for teaching me ECE695 course. This course had a great impact on my theoretical and practical understanding of wireless communication networks. I sincerely acknowledge my colleagues at Purdue, in particular Cloud Computing, Machine Learning, And Networking Research (CLAN) Lab for their continuous encouragement, support and sharing experiences and knowledge. Finally, I would like to express my profound gratitude to my lovely wife, my parents and my kids (Lames and Hashim).

I believe that PhD is not the end of my lifelong learning journey. It is the beginning of my pursuit of knowledge.

"My Lord! Increase me in knowledge.", Quran 20:114

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

SYMBOLS

r	Number of video files in system
m	Number of storage nodes
L_i	Number of segments for video file
$G_{i,j}$	Segment j of video file i
(n_i, k_i)	Erasure code parameters for file i
$C_{i,j}^{(q)}$	q^{th} coded chunk of segment j in file i
λ_i	Possion arrival rate of file i
π_{ij}	Probability of retrieving chunk of file i from node j using probabilistic scheduling algorithm
$\mathcal{S}_i$	Set of storage nodes having coded chunks of file i
$\mathcal{A}_i$	Set of storage nodes used to access chunks from file i
$(\alpha_j,\, \beta_j)$	Parameters of Shifted Exponential distribution
X_j	Service time distribution of a chunk at node j
$M_j(t)$	Moment generating function for the service time of a chunk at node j $M_j(t) = \mathbb{E}\left[e^{tX_j}\right]$
σ	Parameter indexing stall duration tail probability
$D_{i,j}^{(q)}$	Download time for coded chunk $q \in \{1, \ldots, L_i\}$ of file i from storage node j
R_j	Service time of the video files
$\overline{R}_j$	Laplace-Stieltjes Transform of R_j, $\overline{R}_j = \mathbb{E}\left[e^{-sR_j}\right]$

$B_j(t)$	Moment generating function for the service time of video files $B_j(t) = \mathbb{E}\left[e^{tR_j}\right]$
μ_j	Mean service time of a chunk from storage node j
Λ_j	Aggregate arrival rate at node j
ρ_j	Video file request intensity at node j
$T_i^{(q)}$	The time at which the segment $G_{i,q}$ is played back
d_s	Start-up delay
τ	Chunk size in seconds
$\Gamma^{(i)}$	Stall duration tail probability for file i
θ	Trade off factor between mean stall duration and stall duration tail probability
$L_{j,i}$	Number of cached chunks for video file i, at server j
$\pi_{i,j,\ell}$	Probability of retrieving chunk of file i from node j from edge-router ℓ using probabilistic scheduling algorithm
$p_{i,j,\nu_j\ell}$	Probability of retrieving chunk of file i from server j and PS ν_j, if requested through edge-router ℓ.
$q_{i,j,\beta_j\ell}$	Probability of retrieving chunk of file i from server j and PS β_j, if requested through edge-router ℓ.
$\mathrm{PS}_{\beta_j,\ell}^{(d,j)}, \forall \beta_j$	Set of parallel streams between data center and cache server j which serves edge router ℓ
$\mathrm{PS}_{\beta_j,\ell}^{(\bar{d},j)}, \forall \beta_j$	Set of parallel streams between cache server j and edge-router ℓ
$\mathrm{PS}_{\nu_j,\ell}^{(e,j)}, \forall \nu_j$	Set of parallel streams between cache server j and edge-router ℓ assigned to serve cached segments
$\left(\alpha_j^{(d)}, \eta_j^{(d)}\right)$	Parameters of Shifted Exponential distribution of service time from data center to cache server j
$\left(\alpha_{j,\ell}^{(f_j)}, \eta_{j,\ell}^{(f_j)}\right)$	Parameters of Shifted Exponential distribution of service time from cache server j to edge-router ℓ
$M_{j,\beta_j,\ell}^{(d)}$	Moment generating function for the service time of the parallel stream $\mathrm{PS}_{\beta_j,\ell}^{(d,j)}$

$M^{(\bar{d})}_{j,\beta_j,\ell}$	Moment generating function for the service time of the parallel stream $\mathrm{PS}^{(\bar{d},j)}_{\beta_j,\ell}$
$M^{(e)}_{j,\beta_j,\ell}$	Moment generating function for the service time of the parallel stream $\mathrm{PS}^{(e,j)}_{\nu_j,\ell}$
$\Lambda^{(d)}_{j,\beta_j,\ell}$	aggregate arrival rate at parallel stream $\mathrm{PS}^{(d,j)}_{\beta_j,\ell}$
$\Lambda^{(\bar{d})}_{j,\beta_j,\ell}$	aggregate arrival rate at parallel stream $\mathrm{PS}^{(\bar{d},j)}_{\beta_j,\ell}$
$\Lambda^{(e,j)}_{j,\beta_j,\ell}$	aggregate arrival rate at parallel stream $\mathrm{PS}^{(e,j)}_{\beta_j,\ell}$
$X_{i,\ell}$	Random variable corresponding to amount of space in the edge-cache ℓ for video file i
$T^{(u)}_{i,j,\beta_j,\nu_j}$	Time that chunk u begins to play at client i given that it is downloaded from β_j and ν_j.
$\Gamma^{i,j,\beta_j,\nu_j}_{U}$	Stall duration for the request of file i from β_j and ν_j queues (not cached at the edge-router).
$\Gamma^{i,j,\beta_j,\nu_j}_{tot}$	Total stall duration for the request of file i from either β_j and ν_j queues or from the edge-cache.
$D^{(u)}_{i,j,\beta_j,\nu_j}$	Download time fo chunk $u \in \{1,\ldots,L_i\}$ of file i from node j, from β_j and ν_j queues.
$R^{(e)}_{j,\nu_j}$	Service time of the video files at the parallel streams $\mathrm{PS}^{(e,j)}_{(\nu_j)}$
$B^{(d)}_{j,\beta_j}$	MGF of the Service time of all video files from the parallel stream $\mathrm{PS}^{(d,j)}_{(\beta_j)}$
$\rho^{(e)}_{j,\nu_j}$	load intensity at the parallel stream $\mathrm{PS}^{(e,j)}_{(\nu_j)}$.
$\rho^{(d)}_{j,\beta_j}$	load intensity at the parallel stream $\mathrm{PS}^{(d,j)}_{(\beta_j)}$.
$\rho^{(\bar{d})}_{j,\beta_j}$	load intensity at the parallel stream $\mathrm{PS}^{(\bar{d},j)}_{(\beta_j)}$.

ABBREVIATIONS

VSOC	Video streaming over cloud
CDN	Content delivery (or distribution) network
QoE	Quality of experience
OTT	Over-the-top
PSs	Parallel streams
IP	Internet protocol
vCDN	virtual CDN
VoD	Vedio on demand
FOTA	Firmware-over-the-air
LRU	Least recently used
FCDN	fully qualified domain name (FQDN).
iDNS	domain name service (called iDNS)
SDTP	Stall duration tail probability
Pofd	power-of-d
JSQ	join shortest queue
LL	Least Load
NOVA	iNner cOnVex Approximation.
MDS	Maximum Distance Separable
VM	Virtual machine
FIFO	First-in-first-out
RP-OA	Random Placement, Optimized Access
RP-PEA	Random Placement, Projected Equal Access
RP-PEA	Random Placement, Projected Equal Access

OP-PSP	Optimized Placement-Projected Service-Rate Proportional Allocation
RP-PSP	Random Placement-PSP
PEA-QTB	Projected Equal Access, Optimized Quality Probabilities, Auxiliary variables and Bandwidth Wights
PEB-QTA	Bandwidth, Optimized Quality Probabilities, Auxiliary variables and Server Access
PEQ-BTA	Projected Equal Quality, Optimized Bandwidth Wights, Auxiliary variables and Server Access
PSP-QTB	Projected Proportional Service-Rate, Optimized Quality, Auxiliary variables and Bandwidth Wights
PLQ-BTA	Projected Lowest Quality, Optimized Bandwidth Wights, Auxiliary variables and Server Access
PHQ-BTA	Projected Highest Quality, Optimized Bandwidth Wights, Auxiliary variables and Server Access

ABSTRACT

Al-Abbasi, Abubakr O. PhD, Purdue University, May 2020. A Quantitative Framework for CDN-based Over-the-top Video Streaming Systems . Major Professor: Vaneet Aggarwal.

The demand for global video has been burgeoning across industries. With the expansion and improvement of video-streaming services, cloud-based video is evolving into a necessary feature of any successful business for reaching internal and external audiences. Over-the-top (OTT) video streaming, e.g., Netflix and YouTube, has been dominating the global IP traffic in recent years. More than 50% of OTT video traffic are now delivered through content distribution networks (CDNs).

Even though multiple solutions have been proposed for improving congestion in the CDN system, managing the ever-increasing traffic requires a fundamental understanding of the system and the different design flexibilities (control knobs) to make the best use of the hardware limitations. In Addition, there is no analytical understanding for the key quality of experience (QoE) attributes (stall duration, average quality, etc.) for video streaming when transmitted using CDN-based multi-tier infrastructure, which is the focus of this thesis. The key contribution of this thesis is to provide a white-box analytical understanding of the key QoE attributes of the end-user in cloud storage systems, which can be used to systematically address the choppy user experience and have optimized system designs. The first key design involves the scheduling strategy, that chooses the subset of CDN servers to obtain the content. The second key design involves the quality of each video chunk. The third key design involves deciding which contents to cache at the edge routers and which content needs to be stored at the CDN. Towards solving these challenges, this dissertation is divided into three parts. Part 1 considers video streaming over distributed systems where the

video segments are encoded using an erasure code for better reliability. Part 2 looks at the problem of optimizing the tradeoff between quality and stall of the streamed videos. In Part 3, we consider caching partial contents of the videos at the CDN as well as at the edge-routers to further optimize video streaming services.

We present a model for describing a today's representative multi-tier system architecture for video streaming applications, typically composed of a centralized origin server, several CDN sites and edge-caches. Our model comprehensively considers the following factors: limited caching spaces at the CDN sites and edge-routers, allocation of CDN for a video request, choice of different ports from the CDN, and the central storage and bandwidth allocation. With this model, we optimize different quality of experience (QoE) measures and present novel, yet efficient, algorithms to solve the formulated optimization problems. Our extensive simulation results demonstrate that the proposed algorithms significantly outperform the state-of-the-art strategies. We take one step further and implement a small-scale video streaming system in a real cloud environment, managed by Openstack, and validate our results

1. INTRODUCTION

1.1 Motivation

The demand of video streaming services have been skyrocketing over these years, with the global video streaming market expected to grow annually at a rate of 18.3% [1]. With the proliferation and advancement of video-streaming services, cloud-based video has become an imperative feature of any successful business. This can also be seen as IBM estimates cloud-based video will be a $105 billion market opportunity by 2019 [2].

Many industry observers believe that Content Delivery Networks (CDNs), which play a critical role in the delivery of high-quality video, provide an excellent use case for deployment within the evolving frameworks [8]. Increased popularity of OTT (over-the-top) content from Hulu, Netflix, Amazon, Youtube, and others, increasing premium resolution offerings (HD, 8K, 360, VR), and more connected homes and mobile devices have been playing a significant role in the consumption of online video thus helping drive the shift to cloud environments. This proposal focuses on efficient control and end-to-end management of these OTT video streaming systems.

In cloud storage systems, erasure coding has seen itself quickly emerged as a promising technique to reduce the storage cost for a given reliability as compared to the replicated systems [3, 4]. It has been widely adopted in modern storage systems by companies like Facebook [5], Microsoft [6], and Google [7]. We further note that replication is a special case of erasure coding. Thus, the proposed research using erasure-coded content on the servers can also be used when the content is replicated on the servers.

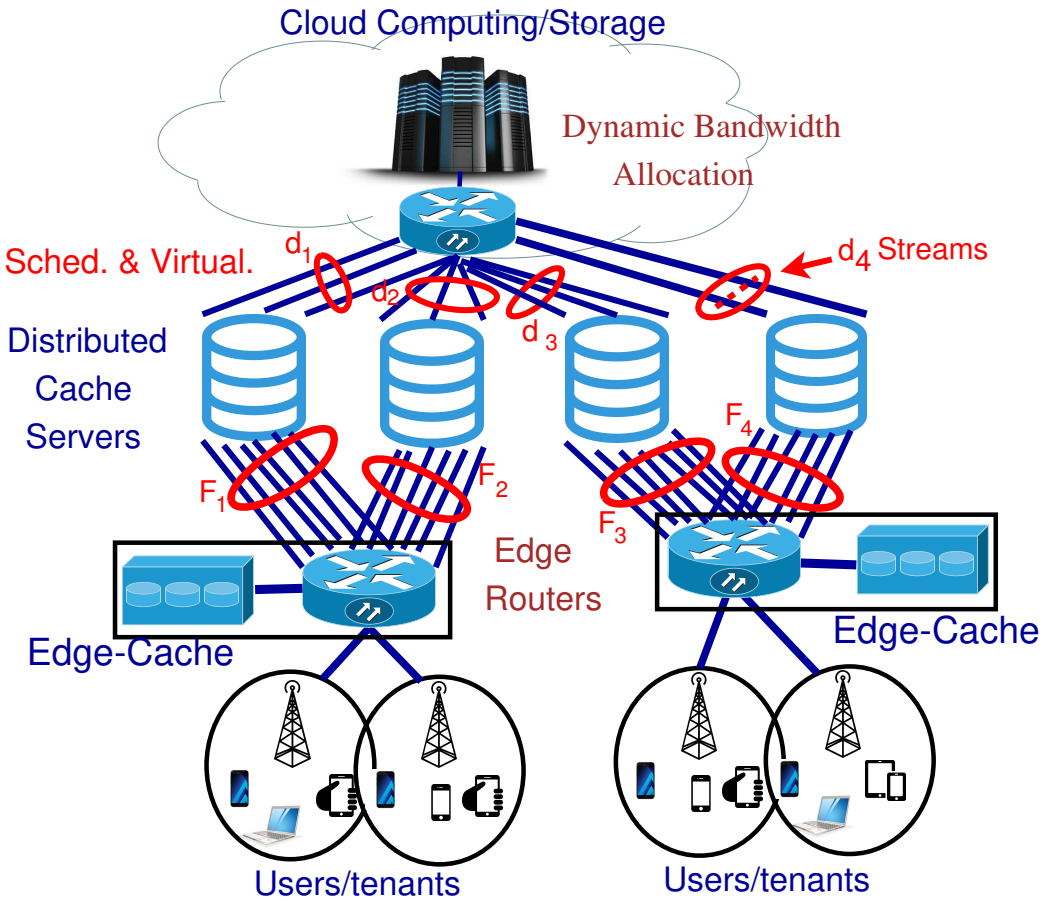

Fig. 1.1.: An illustration of our system model for video content delivery, consisting of a datacenter, four cache servers ($m = 4$), and 2 edge routers. d_j and F_j parallel connections are assumed between datacenter and cache server j, and datacenter and edge router, respectively.

In cloud-based-video, the users are connected to an edge router, which fetch the contents from the distributed storage servers (as depicted in Fig. 1.1[1]). Multiple parallel streams (PSs) between a server and the edge router are considered, which provides the ability to get multiple streams simultaneously. Unlike the case of file download, the later video-chunks do not have to be downloaded as fast as possible to

[1]Detailed explaination for the system model and vCDN will be provided later.

improve the QoE and thus multiple parallel streams help achieve better QoE. This is because later chunks can be downloaded while earlier chunks are streamed. The key differences in streaming of video content as compared to the file download include

1. video can be streamed at different quality which gives an additional choice of the quality of each streamed video,
2. different video-chunks can be obtained from different set of servers,
3. stall duration accounts for time of delivery of each video-chunk rather than just the last video-chunk, and
4. caching policy has to make decisions for every video segment rather than entire video file, since video files are generally larger in size as compared to other files. Thus, the choice among different qualities, caching, and server selection for each video segment makes the problem challenging.

This thesis seeks to improve the QoE of the clients, key attributes of which include (i) Mean stall duration, (ii) Tail probability of stall duration, (iii) Average quality of streamed videos, and (iv) Cache miss (resp. hit) rate of edge-caches.

OTT video streaming, such as Netflix and YouTube, has been dominating the global IP traffic in recent years. It is shown in [8] that video streaming applications in North America now represents 62% of the Internet traffic, and this figure will continue to grow due to the introduction of even higher resolution video formats such as 4K on the horizon. With the growing popularity of video services, increased congestion and latency related to retrieving content from remote datacenters can lead to degraded end customer experience. Service and content providers often seek to mitigate such performance issues by employing caching at the network edge and by pushing content closer to their customers using content distribution networks (CDNs). More than 50% of over-the-top video traffic are now delivered through CDNs [9].

Caching of video content has to address a number of crucial challenges that differ from caching of web objects, see for instance [10] and the references therein. First,

video streaming services such as Netflix [11] often adopt a proactive caching strategy, which conciously pushes video files into local caches during off-peak hours, while cache content is updated according to changes in predicted demand. Due to the correlation in user preferences within different regions [10], it calls for new solutions that take into account both regional and global popularity of video files, for jointly optimizing cache content and performance. Second, video files are significantly larger in size than web objects. In order to minimize congestion and latency, caching of video files must be optimized together with network resource allocation and request scheduling, which however, is currently under-explored. Finally, while recent work have considered video-streaming over distributed storage systems [10], they normally focus on network performance metrics similar to those considered by web object caching (e.g., packet delay and cache hit rate), rather than QoE metrics that are more relevant to end user experience in over-the-top video streaming.

This thesis considers video streaming when the content is placed on cloud servers, where coding is used. We consider two different coding techniques: erasure and repetition coding. The key QoE metric for video streaming is the duration of stalls at the clients. This work gives bounds on the stall durations, and uses that to propose an optimized streaming service that minimizes average QoE for the clients.

1.2 Target System

Our work is motivated by the architecture of a production system with a Virtualized Content Distribution Network (vCDN), as depicted in Fig. 1.1. Such services, for instance, include video-on-demand (VoD), live linear streaming services (also referred to as over-the-top video streaming services), firmware over the air (FOTA) Android updates to mobile devices, etc. The main role of this CDN infrastructure is not only to provide users with lower response time and higher bandwidth, but also to distribute the load (especially during peak time) across many edge locations. Consequently, the core backbone network will have reduced network load and better

response time. The origin server has original data and CDN sites have only part of those data. Each CDN site is composed of multiple cache servers each of which is typically implemented as a VM backed by multiple directly attached solid state drives (SSDs) for higher throughput. The cache servers store video segments and a typical duration of each segment covers $5 - 11$ seconds of playback time.

Further, the typical vCDN architecture includes an additional cache at the edge, called edge cache. This edge cache allows for saving some recently accessed videos. This cache can also help multicasting content to another user connected to the same edge router. One of the typical policy that is used in edge cache is based on least-recently-used (LRU) caching policy [12]. In this work, we will consider a modification of this strategy to weigh the eviction policy of contents dependent on their weight, placement, and access rates and thus can be optimized.

When a client such as VoD/LiveTV app requests a certain content, it goes through multiple steps. First, it sees whether the content is in edge cache. If so, the content is directly accessed from the edge cache. Then, it sees whether the content has been requested by someone connected to the same edge router and is being sent to them. In this case, the content already received at the edge router is sent to the user and the remaining content is passed as received (equivalent to a multicast stream setup). If the content cannot be obtained in the two steps, the client then contacts *CDN manager*, choose the best CDN service to use and retrieve a fully qualified domain name (FQDN). Fourth, with the acquired FQDN, it gets a cache server's IP address from a content routing service (called iDNS). Then we use the IP address to connect to one of the cache servers. The cache server will directly serve the incoming request if it has data in its local storage (cache-hit). If the requested content is not on the cache server (i.e., cache-miss), the cache server will fetch the content from the origin server and then serve the client.

In this thesis, we will present a generic mathematical model applicable to not only our considered system but also other video streaming systems that implement CDN-like two-tier caching structure.

1.3 Thesis Contributions

The key contribution of this thesis is to provide a white-box analytical understanding of the key QoE attributes of the end-user in cloud storage systems, which can be used to systematically address the choppy user experience and have optimized system designs. The first key design involves the scheduling strategy, that chooses the subset of CDN servers to obtain the content. The second key design involves the quality of each video chunk. The third key design involves deciding which contents to cache at the edge routers and which content needs to be stored at the CDN. Towards solving these challenges, the thesis is divided into four parts.

Part 1 considers video streaming over distributed systems where the video segments are encoded using an erasure code for better reliability thus being the first work to our best knowledge that considers video streaming over erasure-coded distributed cloud systems. The download time of each coded chunk of each video segment is characterized and ordered statistics over the choice of the erasure-coded chunks is used to obtain the playback time of different video segments. Using the playback times, bounds on the moment generating function on the stall duration is used to bound the mean stall duration. Moment generating function based bounds on the ordered statistics are also used to bound the stall duration tail probability which determines the probability that the stall time is greater than a pre-defined number. These two metrics, mean stall duration and the stall duration tail probability, are important QoE measures for the end users. Based on these metrics, we formulate an optimization problem to jointly minimize the convex combination of both the QoE metrics averaged over all requests over the placement and access of the video content. The non-convex problem is solved using an efficient iterative algorithm. Numerical results show significant improvement in QoE metrics for cloud-based video as compared to the considered baselines.

Part 2 looks at the problem of optimizing the quality of streamed video. Given multiple parallel streams between each server and the edge router, we determine, for

each client request, the subset of servers to stream the video, as well as one of the parallel streams from each chosen server. In order to have this scheduling, we propose a two-stage probabilistic scheduling. The selection of video quality is also chosen with a certain probability distribution, that is optimized in our algorithm. With these parameters, the playback time of video segments is determined by characterizing the download time of each coded chunk for each video segment. Using the playback times, a bound on the moment generating function of the stall duration is used to bound the mean stall duration. Based on this, we formulate an optimization problem to jointly optimize the convex combination of mean stall duration and average video quality for all requests, where the two-stage probabilistic scheduling, video quality selection, bandwidth split among parallel streams, and auxiliary bound parameters can be chosen. This non-convex problem is solved using an efficient iterative algorithm. Based on the offline version of our proposed algorithm, an online policy is developed where servers selection, quality, bandwidth split, and parallel streams are selected in an online manner. Experimental results show significant improvement in QoE metrics for cloud-based video as compared to the considered baselines.

Different from the previous two parts, in **Part 3** we present a model for describing a today's representative system architecture for video streaming applications, typically composed of a centralized origin server and several CDN sites. Our model comprehensively considers the following factors: limited caching spaces at the CDN sites, allocation of CDN for a video request, choice of different ports from the CDN, and the central storage and bandwidth allocation. With the model, we focus on minimizing a performance metric, stall duration tail probability (SDTP), and present a novel, yet efficient, algorithm to solve the formulated optimization problem. The theoretical bounds with respect to the SDTP metric are also analyzed and presented. Our extensive simulation results demonstrate that the proposed algorithms can significantly improve the SDTP metric, compared to the state-of-the-art strategies. We take one step further and implement a small-scale video streaming systems in a real cloud environment and validate our results.

Since streaming services can include multiple caching tiers, at the distributed servers and the edge routers, efficient content management at these locations improves the QoE of the end users. In this part of the thesis, besides the several CDN caches, edge-caches are placed close to the end users to further improve the QoE of users. The theoretical bounds with respect to the SDTP metric are analyzed and presented. The implementation on a virtualized cloud system manged by Openstack demonstrate that the proposed algorithms can significantly improve the SDTP metric, compared to the baseline strategies.

1.3.1 Thesis Outcomes and Publications

The work throughout my PhD study has resulted in the following publications:

- **Abubakr Alabbasi**, Vaneet Aggarwal, Tian Lan, Yu Xiang, Moo-Ryong Ra, and Yih-Farn R. Chen, "FastTrack: Minimizing Stalls for CDN-based Over-the-top Video Streaming Systems," IEEE Transactions on Cloud Computing, Jun 2019.

- **Abubakr Alabbasi**, Vaneet Aggarwal, and Moo-Ryong Ra, "Multi-tier Caching Analysis in CDN-based Over-the-top Video Streaming Systems," IEEE/ACM Transactions on Networking, vol. 27, no. 2, pp. 835-847, April 2019.

- **Abubakr Alabbasi** and Vaneet Aggarwal, "Video Streaming in Distributed Erasure-coded Storage Systems: Stall Duration Analysis," IEEE/ACM Transactions on Networking, vol. 26, no. 4, pp. 1921-1932, Aug. 2018.

- **Abubakr Al-Abbasi** and Vaneet Aggarwal, "Optimized Video Streaming over Cloud: A Stall-Quality Trade-off," Submitted to ACM Tompecs, Aug 2018 (Major Rev., Revised Apr 2019, v2).

- **Abubakr Alabbasi** and Vaneet Aggarwal, "Joint Information Freshness and Completion Time Optimization for Vehicular Networks," Submitted to IEEE Transactions on Service Computing, May 2018 (Major Rev, under revision, v3).

- **Abubakr Alabbasi** and Vaneet Aggarwal, "Stall-Quality Tradeoff for Cloud-based Video Streaming," in Proc. IEEE SPCOM, Jul 2018

- **Abubakr Alabbasi** and Vaneet Aggarwal, "EdgeCache: An Optimized Algorithm for CDN-based Over-the-top Video Streaming Services," in Proc. Infocom Workshop (International Workshop on Integrating Edge Computing, Caching, and Offloading in Next Generation Networks (IECCO)), Apr 2018.

- **Abubakr Alabbasi** and Vaneet Aggarwal, "Mean Latency Optimization in Erasure-coded Distributed Storage Systems," in Proc. Infocom Workshop (International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA)), Apr 2018.

Besides working towards my PhD thesis, I have worked on some other research areas which, as a result, produces the following papers:

- **Abubakr Al-Abbasi**, Arnob Ghosh, and Vaneet Aggarwal, "DeepPool: Distributed Model-free Algorithm for Ride-sharing using Deep Reinforcement Learning," Accepted to IEEE Transactions on Intelligent Transportation Systems, Jul 2019.
- **Abubakr Alabbasi**, Vaneet Aggarwal, and Tian Lan, "TTLoC: Taming Tail Latency for Erasure-coded Cloud Storage Systems," Accepted to IEEE TNSM, May 2019.
- **Abubakr Alabbasi**, and Vaneet Aggarwal, "BitFedEx: Quantifying Data Freshness and Tail Latency in Multi-path Wireless Networks ," submitted to INFOCOM 2020.
- **Abubakr Alabbasi**, and Vaneet Aggarwal, "Swift: Joint Information Freshness and Completion Time Optimization for Cloud Applications," submitted to INFOCOM 2020.

- Ashutosh Singh, **Abubakr Alabbasi**, and Vaneet Aggarwal, "A Reinforcement Learning Based Algorithm for Multi-hop Ride-sharing: Model-free Approach," in Proc. Neurips Workshop, Dec 2019.
- **Abubakr Alabbasi**, Ali Elghariani, Anis Elgabli, and Vaneet Aggarwal, "On the Information Freshness and Tail Latency Trade-off in Mobile Networks," in Proc. Globecomm, Dec 2019.
- Ashutosh Singh, **Abubakr Alabbasi**, and Vaneet Aggarwal, "A Distributed Model-Free Algorithm for Multi-hop Ride-sharing using Deep Reinforcement Learning," Submitted to IEEE Transactions on Intelligent Trasnportation Systems, Oct 2019.
- Ashwin Kumar Boddeti, **Abubakr Alabbasi**, Vaneet Aggarwal, and Zubin Jacob, "Spectral domain inverse design for accelerating nanocomposite metamaterials discovery," submitted to Optical Materials Express, (Major Rev., Nov. 2019 Major Rev.)
- **Abubakr Alabbasi** and Vaneet Aggarwal, "TTLCache: Minimizing Latency in Erasure-coded Storage through Time To Live Caching," Submitted to IEEE Transactions on Network and Service Management, May 2019 (Major Rev., under revision).
- **Abubakr Alabbasi**, Ali Elghariani, Anis Elgabli, and Vaneet Aggarwal, "PSS: Joint Information Freshness and Tail Latency Optimization for Real-time Applications," Submitted to IEEE/ACM TON, Mar 2019 (Major Rev., Revised Sept 2019, v2).

1.3.2 Thesis Organization

The structure of this thesis is organized as follows. Chapter 2 provides related work for this work. We first provide the related work in distributed storage systems and highlight how our work advances them. Then, we present the related work in video streaming and scheduling over parallel servers.

In Chapter 3, we propose a framework for optimizing the video over erasure distributed storage system. In this chapter, we provide expressions for the download and play times of the chunks which are used to find the upper bounds on the QoE metrics of the mean stall duration and video stall latency. Then, based on these expressions, we formulate the QoE optimization problem as a weighted combination of the two QoE metrics and propose the iterative algorithmic solution of this problem. At the end of this chapter, numerical results are provided and a summary for the main results are presented.

Chapter 4 extends the model presented in Chapter 3 to the scenarios where videos can be streamed over parallel links with different quality levels. We first describe the system model with a description of video streaming over cloud storage. We then derive the download and play times of the chunks in closed forms. Based on this analysis, we formulate a generic QoE optimization problem as a weighted combination of the two QoE metrics (quality and stall) and propose efficient algorithms to solve this problem. Simulation results show the superiority of our approach and the key conclusion outcome are provided at the end of this chapter.

Chapter 5 presents a model for describing a today's representative system architecture for video streaming applications, typically composed of a centralized origin server, several CDN sites, and edge-caches. We start by describing the system model used in the work with a description of CDN-based Over-the-top video streaming systems. We then provide an upper bound on the mean stall duration which is then used to formulate the QoE optimization problem as a weighted sum of all SDTP of all files and propose simple, yet efficient, solution for our problem. Experimental results are also. We conclude this chapter by highlighting the key remarks concluded out of this work.

The key observations and the major conclusion remarks are presented in Chapter 6.

2. BACKGROUND AND LITERATURE REVIEW

2.1 Latency in Erasure-coded Storage

While latency in erasure coded storage systems has been widely studied, to our best knowledge, quantifying exact latency for erasure-coded storage system in data-center network is an open problem. Prior works focusing on asymptotic queuing delay behaviors [13, 14] are not applicable because redundancy factor in practical data centers typically remains small due to storage cost concerns. Due to the lack of analytic latency models, most of the literature is focused on reliable distributed storage system design, and latency is only presented as a performance metric when evaluating the proposed erasure coding scheme, e.g., [15, 16], which demonstrate latency improvement due to erasure coding in different system implementations. Related design can also be found in data access scheduling [17, 18], access collision avoidance [19], and encoding/decoding time optimization [20] and there is also some work using the LT erasure codes to adjust the system to meet user requirements such as availability, integrity and confidentiality [21].

Recently, there has been a number of attempts at finding latency bounds for an erasure-coded storage system [22–26]. The key scheduling approaches include *block-one-scheduling* policy that only allows the request at the head of the buffer to move forward [27], fork-join queue [25, 28] to request data from all server and wait for the first k to finish, and the probabilistic scheduling [22, 23] that allows choice of every possible subset of k nodes with certain probability. Mean latency and tail latency have been characterized in [22, 23] and [29] respectively for a system with multiple files using probabilistic scheduling. This work considers video streaming rather than file downloading. The metrics for video streaming does not only account for the end of the download of the video but also of the download of each of the segment. Thus,

the analysis for the content download cannot be extended to the video streaming directly and the analysis approach in this work is very different from the prior works in the area.

2.2 Video Streaming over Cloud

Servicing Video on Demand and Live TV Content from cloud servers have been studied widely [30–34]. The placement of content and resource optimization over the cloud servers have been considered. To the best of our knowledge, reliability of content over the cloud servers have not been considered for video streaming applications. In the presence of erasure-coding, there are novel challenges to characterize and optimize the QoE metrics at the end user. Adaptive streaming algorithms have also been considered for video streaming [35, 36], which are beyond the scope of this work and are left for future work.

In [37], authors utilize the social information propagation pattern to improve the efficiency of social video distribution. Further, they used replication and user request dispatching mechanism in the cloud content delivery network architecture to reduce the system operational cost, while maintaining the averaged service latency. However, this work considers only video download. The benefits of delivering videos at the edge network is shown in [38]. Authors show that bringing videos at the edge network can significantly improve the content item delivery performance, in terms of improving quality experienced by users as well as reducing content item delivery costs. To the best of our knowledge, reliability of content over the cloud servers have not been considered for video streaming applications. There are novel challenges to characterize and optimize the QoE metrics at the end user. In [39–41] a predictive model of video, apps, and web QoE is developed using machine-learning algorithms. However, these works do not model (or quantify) the impact of the control parameters on quality metrics or engagement. In [42], the effects of end-user mobility on the perceived popularity distribution and join strategy across different

CDN sites are considered. The data flow between different CDNs are investigated in [43]. However, none of these works quantify the SDTP and/or optimize the resources or chunk placement. Adaptive streaming algorithms have also been considered for video streaming [36, 44, 45] which are beyond the scope of this work and are left for future work.

2.3 Scheduling in Distributed Systems

Different approaches have been proposed to schedule task on different servers. Some examples of these approaches are d-choose-2, or power-of-2 (pof(d)), In this policy, d servers are randomly selected and then the request is sent to the shortest two queues/links. Similar approaches like join shortest queue (JSQ) and Least Load-d LL(d) are proposed in [46–48]. However, these approaches are queue dependent, hence have to keep tracking the instantaneous queue levels, which increases the complexity of servers selection. Further, unlike our policy, these approaches do not differentiate between the different updates. We give more priority to the updates with higher weights (e.g., higher arrival rates) in order to minimize the overall performance.

2.4 Caching Analysis

Performance of caching mechanisms is hard to analyze. This is because the Markov chain associated with a single cache (adopting LRU for example) has exponential number of states [49]. Several approximation methods have been investigated, in the literature. Two key types of analysis techniques are considered: the characteristic time approximation based approaches and the network calculus based approaches. One of the key metrics to quantify caching systems is the hit ratio, which describes the probability of finding a file in the cache given a popularity distribution on the set of available content. Authors in [50] proposed a method for approximating the hit rates for an LRU caching system assuming that all files are of identical size. However, in most cases, files are of different sizes. Further work in [51] extends the previous

work to accommodate the case of multiple file sizes. Several variants of LRU have been proposed, including q-LRU, k-LRU, RANDOM, and k-RANDOM [52]. In order to have better performance with realistic file sizes, multiple approaches have been proposed. In [53], an admission control strategy is used to decrease the probability that a large file size is added in the cache, thus reducing the possibilities that a large file arrival can evict multiple small files.

TTL-based caching has been extensively studied in the literature. TTL caching model has connections to the popular caching policies such as LRU. Even though TTL caching has been widely studied, they have not been analyzed for characterizing latency in erasure-coded storage systems which is the focus of this paper. Further, our model differs from previous models by associating TTL window to files (ω_i for file i) and thus a file is evicted if not requested in the last ω_i time, if needed.

3. VIDEO STREAMING OVER DISTRIBUTED STORAGE SYSTEMS

3.1 Introduction

In this part, we consider two measures of QoE metrics in terms of stall duration. The first is the mean stall duration. Almost every viewer can relate to the quality of experiences for watching videos being the stall duration and is thus one of the key focus in the studied streaming algorithms [54, 55]. The second is the probability that the stall duration is greater than a fixed number x, which determines the stall duration tail probability. It has been shown that in modern Web applications such as Bing, Facebook, and Amazon's retail platform, the long tail of latency is of particular concern, with 99.9th percentile response times that are orders of magnitude worse than the mean [11,56]. Thus, the QoE metric of stall duration tail probability becomes important. This work characterizes an upper bound on both QoE metrics.

We note that quantifying service latency for erasure-coded storage is an open problem [26], and so is tail latency [29]. This thesis takes a step forward and explores the notions for video streaming rather than video download. Thus, finding the exact QoE metrics is an open problem. This work finds the bounds on the QoE metrics. The data chunk transfer time in practical systems follows a shifted exponential distribution [23, 24] which motivates the choice that the service time distribution for each video server is a shifted exponential distribution. Further, the request arrival rates for each video is assumed to be Poisson. The video segments are encoded using an (n, k) erasure code and the coded segments are placed on n different servers. When a video is requested, the segments need to be requested from k out of n servers. Optimal strategy of choosing these k servers would need a Markov approach similar to that in [26] and suffers from a similar state explosion problem, because states of

the corresponding queuing model must encapsulate not only a snapshot of the current system including chunk placement and queued requests but also past history of how chunk requests have been processed by individual nodes.

In this work, we use the probabilistic scheduling proposed in [22, 23] to access the k servers, where each possibility of k servers is chosen with certain probability and the probability terms can be optimized. Using this scheduling mechanism, the random variables corresponding to the times for download of different video segments from each server are found. Using ordered statistics over the k servers, the random variables corresponding to the playback time of each video segment are characterized. These are then used to find bounds on the mean stall duration and the stall duration tail probability. Moment generating functions of the ordered statistics of different random variables are used in the bounds. We note that the problem of finding latency for file download is very different from the video stall duration for streaming. This is because the stall duration accounts for download time of each video segment rather than only the download time of the last video segment. Further, the download time of segments are correlated since the download of chunks from a server are in sequence and the playback time of a video segment are dependent on the playback time of the last segment and the download time of the current segment. Taking these dependencies into account, this work characterizes the bounds on the two QoE metrics. We note that for the special case when each video has a single segment, the bounds on mean stall duration and stall duration tail probability reduce to that for file download. Further, the bounds based on the approach in this work have been shown to outperform the results for mean file download latency in [22, 23].

The proposed framework provides a mathematical crystallization of the engineering artifacts involved and illuminates key system design issues through optimization of QoE. The average QoE metric over different requests can be optimized over the placement of the video files, the access of the video files from the servers, and the bound parameters. The tradeoff in the two QoE metrics is captured by defining the objective function which is a convex combination of the two QoE metrics. Varying

the parameter trading off the two metrics can be used to get a tradeoff region between the two metrics helping the system designer to choose an appropriate point. An efficient algorithm is proposed to solve the proposed non-convex problem. The proposed algorithm does an alternating optimization over the placement, access, and the bound parameters. The optimization over probabilistic scheduling access parameters help reduce the mean and tail of the stall durations by differentiating video files thus providing more flexibility as compared to choosing the lowest queue servers.

The sub-problems have been shown to have convex constraints and thus can be efficiently solved using iNner cOnVex Approximation (NOVA) algorithm proposed in [57]. The proposed algorithm is shown to converge to a local optimal. Numerical results demonstrate significant improvement of QoE metrics as compared to the baselines.

Today, cloud-based video does not use erasure coding. One of the key reason is the additional decoding latency from multiple coded streams. Since the computing has been growing exponentially [58], it is only a matter of time when the computation of decoding will not limit the latencies in delay sensitive video streaming and the networking latency will govern the system designs. Further, we note that replication is a special case of erasure coding. Thus, the proposed research using erasure-coded content on the servers can also be used when the content is replicated on the servers.

The key contributions of this part of the thesis include:

- This chapter formulates video streaming over erasure-coded cloud storage system.

- The random variable corresponding to the download time of a chunk of each video segment from a server is characterized. Using ordered statistics, the random variable corresponding to the playback time of each video segment is found. These are further used to derive upper bounds on the mean stall duration of the video and the video stall duration tail probability.

- The QoE metrics are used to formulate system optimization problems over the choice of the placement of video segments, probabilistic scheduling access policy and the bound parameters which are related to the moment generating function. Efficient iterative solutions are provided for these optimization problems.

- Numerical results show that the proposed algorithms converges within a few iterations. Further, the QoE metrics are shown to have significant improvement as compared to the considered baselines. For instance, the mean stall duration for the proposed algorithm is 60% smaller and the stall duration tail probability is orders of magnitude better as compared to random placement and projected equal access probability strategy.

3.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 2 provides related work for this work. In Section 3 of this chapter, we describe the system model used in the work with a description of video streaming over cloud storage. Section 4 derives expressions on the download and play times of the chunks which are used in Sections 5 and 6 to find the upper bounds on the QoE metrics of the mean stall duration and video stall latency, respectively. Section 7 formulates the QoE optimization problem as a weighted combination of the two QoE metrics and proposes the iterative algorithmic solution of this problem. Numerical results are provided in Section 8. Section 9 concludes the work.

3.3 System Model

We consider a distributed storage system consisting of m heterogeneous servers (also called storage nodes), denoted by $\mathcal{M} = 1, 2, ..., m$. Each video file i, where $i = 1, 2, ...r$, is divided into L_i equal segments, $G_{i,1}, \cdots, G_{i,L_i}$, each of length τ sec. Then, each segment $G_{i,j}$ for $j \in \{1, 2, \ldots, L_i\}$ is partitioned into k_i fixed-size chunks

and then encoded using an (n_i, k_i) Maximum Distance Separable (MDS) erasure code to generate n_i distinct chunks for each segment $G_{i,j}$. These coded chunks are denoted as $C_{i,j}^{(1)}, \cdots, C_{i,j}^{(n_i)}$. The encoding setup is illustrated in Figure 4.2.

The encoded chunks are stored on the disks of n_i distinct storage nodes. These storage nodes are represented by a set $\mathcal{S}_i$, such that $\mathcal{S}_i \subseteq \mathcal{M}$ and $n_i = |\mathcal{S}_i|$. Each server $z \in \mathcal{S}_i$ stores all the chunks $C_{i,j}^{(g_z)}$ for all j and for some $g_z \in \{1, \cdots, n_i\}$. In other words, each of the n_i storage nodes stores one of the coded chunks for the entire duration of the video. The placement on the servers is illustrated in Figure 3.2, where the server 1 is shown to store first coded chunks of file i, third coded chunks of file u and first coded chunks for file v.

The use of (n_i, k_i) of MDS erasure code introduces a redundancy factor of n_i/k_i which allows the video to be reconstructed from the video chunks from any subset of k_i-out-of-n_i servers. We note that the erasure-code can also help in recovery of the content i as long as k_i of the servers containing file i are available [4]. Note that replication along n servers is equivalent to choosing $(n, 1)$ erasure code. Hence, when a video i is requested, the request goes to a set $\mathcal{A}_i$ of the storage nodes, where $\mathcal{A}_i \subseteq \mathcal{S}_i$ and $k_i = |\mathcal{A}_i|$. From each server $z \in \mathcal{A}_i$, all chunks $C_{i,j}^{(g_z)}$ for all j and the value of g_z corresponding to that placed on server z are requested. The request is illustrated in Figure 3.2. In order to play a segment q of video i, $C_{i,q}^{(g_z)}$ should have been downloaded from all $z \in \mathcal{A}_i$. We assume that an edge router which is a combination of multiple users is requesting the files. Thus, the connections between the servers and the edge router is considered as the bottleneck. Since the service provider only has control over this part of the network and the last hop may not be under the control of the provider, the service provider can only guarantee the quality-of-service till the edge router.

We assume that the files at each server are served in order of the request in a first-in-first-out (FIFO) policy. Further, the different chunks are processed in order of the duration. This is depicted in Figure 5.1, where for a server q, when a file i is

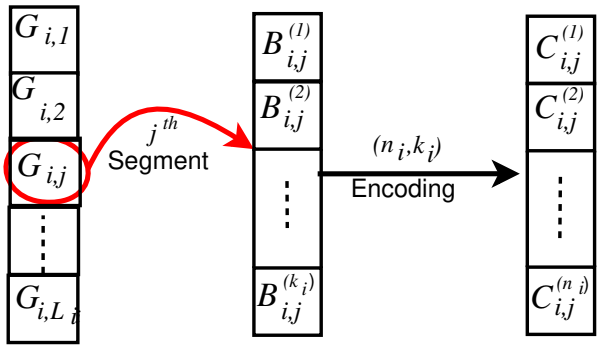

Fig. 3.1.: A schematic illustrates video fragmentation and erasure-coding processes. Video i is composed of L_i segments. Each segments is partitioned into k_i chunks and then encoded using an (n_i, k_i) MDS code.

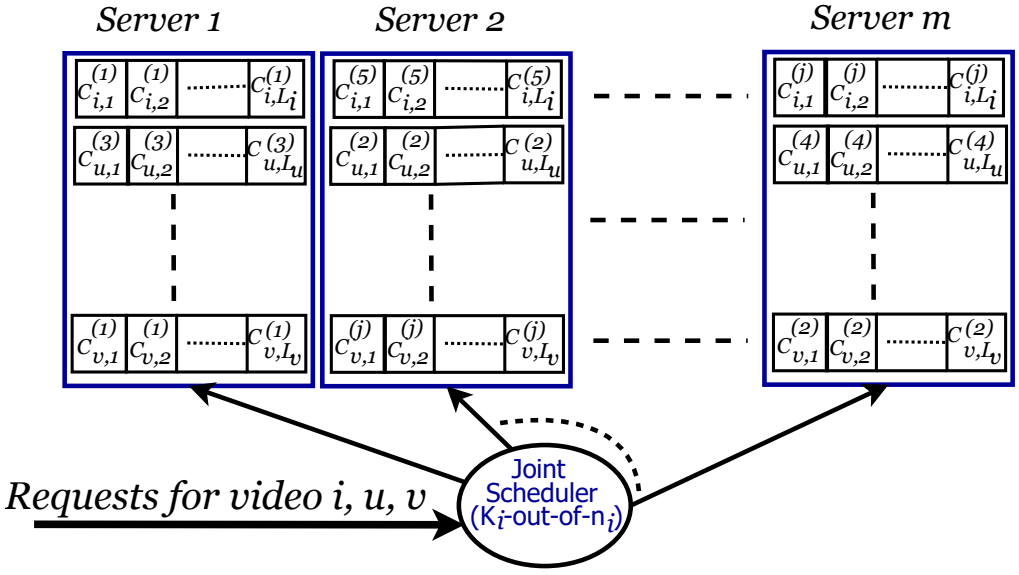

Fig. 3.2.: An Illustration of a distributed storage system equipped with m nodes and storing 3 video files assuming (n_i, k_i) erasure codes.

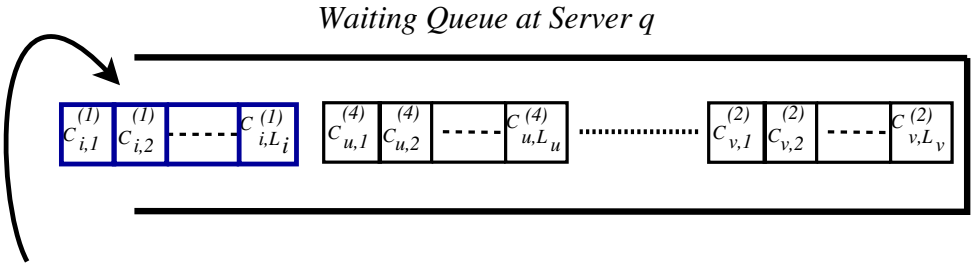

Fig. 3.3.: An Example of the instantaneous queue status at server q, where $q \in 1, 2, ..., m$.

requested, all the chunks are placed in the queue where other video requests before this that have not yet been served are waiting.

In order to schedule the requests for video file i to the k_i servers, the choice of k_i-out-of-n_i servers is important. Finding the optimal choice of these servers to compute the latency expressions is an open problem to the best of our knowledge. Thus, this work uses a policy, called Probabilistic Scheduling, which was proposed in [22, 23]. This policy allows choice of every possible subset of k_i nodes with certain probability. Upon the arrival of a video file i, we randomly dispatch the batch of k_i chunk requests to appropriate a set of nodes (denoted by set $\mathcal{A}_i$ of servers for file i) with predetermined probabilities ($P(\mathcal{A}_i)$ for set $\mathcal{A}_i$ and file i). Then, each node buffers requests in a local queue and processes in order and independently as explained before. The authors of [22, 23] proved that a probabilistic scheduling policy with feasible probabilities $\{P(\mathcal{A}_i) : \forall_i, \mathcal{A}_i\}$ exists if and only if there exists conditional probabilities $\pi_{ij} \in [0, 1]\ \forall i, j$ satisfying

$$\sum_{j=1}^{m} \pi_{ij} = k_i \quad \forall i \qquad \text{and} \quad \pi_{ij} = 0 \quad \text{if } j \notin \mathcal{S}_i.$$

In other words, selecting each node j with probability π_{ij} would yield a feasible choice of $\{P(\mathcal{A}_i) : \forall_i, \mathcal{A}_i\}$. Thus, we consider the request probabilities π_{ij} as the probability that the request for video file i uses server j. While the probabilistic scheduling have been used to give bounds on latency of file download, this work uses the scheduling to give bounds on the QoE for video streaming.

We note that it may not be ideal in practice for a server to finish one video request before starting another since that increases delay for the future requests. However, this can be easily alleviated by considering that each server has multiple queues (streams) to the edge router which can all be considered as separate servers. These multiple streams can allow multiple parallel videos from the server. The probabilistic scheduling can choose k_i of the overall queues to access the content. Possible approaches of extension to accommodate such scenarios are shown in the Appendix A.10.

We now describe a queuing model of the distributed storage system. We assume that the arrival of client requests for each video i form an independent Poisson process with a known rate λ_i. The arrival of file requests at node j forms a Poisson Process with rate $\Lambda_j = \sum_i \lambda_i \pi_{i,j}$ which is the superposition of r Poisson processes each with rate $\lambda_i \pi_{i,j}$.

We assume that the chunk service time for each coded chunk $C_{i,l}^{(g_j)}$ at server j, X_j, follows a shifted exponential distribution as has been demonstrated in realistic systems [23, 24]. The service time distribution for the chunk service time at server j, X_j, is given by the probability distribution function $f_j(x)$, which is

$$f_j(x) = \begin{cases} \alpha_j e^{-\alpha_j (x-\beta_j)}, & x \geq \beta_j \\ 0, & x < \beta_j \end{cases} . \tag{3.1}$$

We note that exponential distribution is a special case with $\beta_j = 0$. We note that the constant delays like the networking delay, and the decoding time can be easily factored into the shift of the shifted exponential distribution. Let $M_j(t) = \mathbb{E}\left[e^{tX_j}\right]$ be the moment generating function of X_j. Then, $M_j(t)$ is given as

$$M_j(t) = \frac{\alpha_j}{\alpha_j - t} e^{\beta_j t} \qquad t < \alpha_j \tag{3.2}$$

We note that the arrival rates are given in terms of the video files, and the service rate above is provided in terms of the coded chunks at each server. The client plays the video segment after all the k_i chunks for the segment have been downloaded and the previous segment has been played. We also assume that there is a start-up delay of d_s (in seconds) for the video which is the duration in which the content can be buffered but not played. This work will characterize the stall duration and stall duration tail probability for this setting.

3.4 Download and Play Times of the Chunks

In order to understand the stall duration, we need to see the download time of different coded chunks and the play time of the different segments of the video.

3.4.1 Download Times of the Chunks from each Server

In this subsection, we will quantify the download time of chunk for video file i from server j which has chunks $C_{i,q}^{(g_j)}$ for all $q = 1, \cdots L_i$. We consider download of q^{th} chunk $C_{i,q}^{(g_j)}$. As seen in Figure 5.1, the download of $C_{i,q}^{(g_j)}$ consists of two components - the waiting time of all the video files in queue before file i request and the service time of all chunks of video file i up to the q^{th} chunk. Let W_j be the random variable corresponding to the waiting time of all the video files in queue before file i request and $Y_j^{(q)}$ be the (random) service time of coded chunk q for file i from server j. Then, the (random) download time for coded chunk $q \in \{1, \cdots, L_i\}$ for file i at server $j \in \mathcal{A}_i$, $D_{i,j}^{(q)}$, is given as

$$D_{i,j}^{(q)} = W_j + \sum_{v=1}^{q} Y_j^{(v)}. \tag{3.3}$$

We will now find the distribution of W_j. We note that this is the waiting time for the video files whose arrival rate is given as $\Lambda_j = \sum_i \lambda_i \pi_{i,j}$. Since the arrival rate of video files is Poisson, the waiting time for the start of video download from a server j, W_j, is given by an M/G/1 process. In order to find the waiting time, we would need to find the service time statistics of the video files. Note that $f_j(x)$ gives the service time distribution of only a chunk and not of the video files.

Video file i consists of L_i coded chunks at server j ($j \in \mathcal{S}_i$). The total service time for video file i at server j if requested from server j, $ST_{i,j}$, is given as

$$ST_{i,j} = \sum_{v=1}^{L_i} Y_j^{(v)}. \tag{3.4}$$

The service time of the video files is given as

$$R_j = \begin{cases} ST_{i,j} & \text{with probability } \frac{\pi_{ij}\lambda_i}{\Lambda_j} \quad \forall i, \end{cases} \tag{3.5}$$

since the service time is $ST_{i,j}$ when file i is requested from server j. Let $\overline{R}_j(s) = \mathbb{E}[e^{-sR_j}]$ be the Laplace-Stieltjes Transform of R_j.

Lemma 1 *The Laplace-Stieltjes Transform of* R_j, $\overline{R}_j(s) = \mathbb{E}\left[e^{-s\overline{R}_j}\right]$ *is given as*

$$\overline{R}_j(s) = \sum_{i=1}^{r} \frac{\pi_{ij}\lambda_i}{\Lambda_j} \left(\frac{\alpha_j e^{-\beta_j s}}{\alpha_j + s}\right)^{L_i} \tag{3.6}$$

Proof The proof is provided in Appendix A.1. ■

Corollary 1 *The moment generating function for the service time of video files when requested from server* j, $B_j(t)$, *is given by*

$$B_j(t) = \sum_{i=1}^{r} \frac{\pi_{ij}\lambda_i}{\Lambda_j} \left(\frac{\alpha_j e^{\beta_j t}}{\alpha_j - t}\right)^{L_i} \tag{3.7}$$

for any $t > 0$, *and* $t < \alpha_j$.

Proof This corollary follows from (4.10) by setting $t = -s$. ■

The server utilization for the video files at server j is given as $\rho_j = \Lambda_j \mathbb{E}[R_j]$. Since $\mathbb{E}[R_j] = B_j'(0)$, using Lemma 4.10, we have

$$\rho_j = \sum_i \pi_{ij}\lambda_i L_i \left(\beta_j + \frac{1}{\alpha_j}\right). \tag{3.8}$$

Having characterized the service time distribution of the video files via a Laplace-Stieltjes Transform $\overline{R}_j(s)$, the Laplace-Stieltjes Transform of the waiting time W_j can be characterized using Pollaczek-Khinchine formula for M/G/1 queues [59], since the request pattern is Poisson and the service time is general distributed. Thus, the Laplace-Stieltjes Transform of the waiting time W_j is given as

$$\mathbb{E}\left[e^{-sW_j}\right] = \frac{(1-\rho_j)\, s}{s - \Lambda_j\left(1 - \overline{R}_j(s)\right)} \tag{3.9}$$

Having characterized the Laplace-Stieltjes Transform of the waiting time W_j and knowing the distribution of $Y_j^{(v)}$, the Laplace-Stieltjes Transform of the download time $D_{i,j}^{(q)}$ is given as

$$\mathbb{E}[e^{-sD_{i,j}^{(q)}}] = \frac{(1-\rho_j)\, s}{s - \Lambda_j\left(1 - \overline{R}_j(s)\right)} \left(\frac{\alpha_j}{\alpha_j + s}\, e^{-\beta_j s}\right)^{q}. \tag{3.10}$$

We note that the expression above holds only in the range of s when $s-\Lambda_j\left(1-\overline{R}_j(s)\right) > 0$ and $\alpha_j + s > 0$. Further, the server utilization ρ_j must be less than 1. The overall download time of all the chunks for the segment $G_{i,q}$ at the client, $D_i^{(q)}$, is given by

$$D_i^{(q)} = \max_{j \in \mathcal{A}_i} D_{i,j}^{(q)}. \tag{3.11}$$

3.4.2 Play Time of Each Video Segment

Let $T_i^{(q)}$ be the time at which the segment $G_{i,q}$ is played (started) at the client. The startup delay of the video is d_s. Then, the first segment can be played at the maximum of the time the first segment can be downloaded and the startup delay. Thus,

$$T_i^{(1)} \;=\; \max\left(d_s,\, D_i^{(1)}\right). \tag{3.12}$$

For $1 < q \leq L_i$, the play time of segment q of file i is given by the maximum of the time it takes to download the segment and the time at which the previous segment is played plus the time to play a segment (τ seconds). Thus, the play time of segment q of file i, $T_i^{(q)}$ can be expressed as

$$T_i^{(q)} \;=\; \max\left(T_i^{(q-1)} + \tau,\, D_i^{(q)}\right). \tag{3.13}$$

Equation (4.18) gives a recursive equation, which can yield

$$\begin{aligned} T_i^{(L_i)} &= \max\left(T_i^{(L_i-1)} + \tau,\, D_i^{(L_i)}\right) \\ &= \max\left(T_i^{(L_i-2)} + 2\tau,\, D_i^{(L_i-1)} + \tau,\, D_i^{(L_i)}\right) \\ &= \max\left(d_s + (L_i-1)\tau,\right. \\ &\qquad \left.\max_{z=2}^{L_i+1} D_i^{(z-1)} + (L_i - z + 1)\tau\right) \end{aligned} \tag{3.14}$$

Since $D_i^{(q)} = \max_{j \in \mathcal{A}_i} D_{i,j}^{(q)}$ from (4.16), $T_i^{(L_i)}$ can be written as

$$T_i^{(L_i)} = \max_{z=1}^{L_i+1} \max_{j \in \mathcal{A}_i} \left(p_{i,j,z} \right), \tag{3.15}$$

where

$$p_{i,j,z} = \begin{cases} d_s + (L_i - 1)\,\tau & , \; z = 1 \\ \\ D_{i,j}^{(z-1)} + (L_i - z + 1)\tau & , \; 2 \leq z \leq (L_i + 1) \end{cases} \tag{3.16}$$

We next give the moment generating function of $p_{i,j,z}$ that will be used in the calculations of the QoE metrics in the next sections. Hence, we define the following lemma.

Lemma 2 *The moment generating function for $p_{i,j,z}$, is given as*

$$\mathbb{E}\left[e^{tp_{i,j,z}}\right] = \begin{cases} e^{t(d_s+(L_i-1)\tau)} & , \; z = 1 \\ e^{t(L_i+1-z)\tau} Z_{i,j}^{(z-1)}(t) & , 2 \leq z \leq L_i + 1 \end{cases} \tag{3.17}$$

where

$$Z_{i,j}^{(\ell)}(t) = \mathbb{E}[e^{tD_{i,j}^{(\ell)}}] = \frac{(1-\rho_j)\, t\, (M_j(t))^{\ell}}{t - \Lambda_j\,(B_j(t) - 1)} \tag{3.18}$$

Proof The proof is provided in Appendix A.2. ■

Ideally, the last segment should be completed by time $d_s + L_i\tau$. The difference between $T_i^{(L_i)}$ and $d_s + (L_i - 1)\tau$ gives the stall duration. Note that the stalls may occur before any segment. This difference will give the sum of durations of all the stall periods before any segment. Thus, the stall duration for the request of file $\delta^{(i)}$ is given as

$$\Gamma^{(i)} = T_i^{(L_i)} - d_s - (L_i - 1)\tau. \tag{3.19}$$

In the next two sections, we will use this stall time to determine the bounds on the mean stall duration and the stall duration tail probability.

3.5 Mean Stall Duration

In this section, we will provide a bound for the first QoE metric, which is the mean stall duration for a file i. We will find the bound by probabilistic scheduling and since probabilistic scheduling is one feasible strategy, the obtained bound is an upper bound to the optimal strategy.

Using (4.24), the expected stall time for file i is given as follows

$$\begin{aligned} \mathbb{E}\left[\Gamma^{(i)}\right] &= \mathbb{E}\left[T_i^{(L_i)} - d_s - (L_i - 1)\,\tau\right] \\ &= \mathbb{E}\left[T_i^{(L_i)}\right] - d_s - (L_i - 1)\,\tau \end{aligned} \tag{3.20}$$

An exact evaluation for the play time of segment L_i is hard due to the dependencies between p_{jz} random variables for different values of j and z, where $z \in (1, 2, ..., L_i + 1)$ and $j \in \mathcal{A}_i$. Hence, we derive an upper-bound on the playtime of the segment L_i as follows. Using Jensen's inequality [60], we have for $t_i > 0$,

$$e^{t_i \mathbb{E}\left[T_i^{(L_i)}\right]} \leq \mathbb{E}\left[e^{t_i T_i^{(L_i)}}\right]. \tag{3.21}$$

Thus, finding an upper bound on the moment generating function for $T_i^{(L_i)}$ can lead to an upper bound on the mean stall duration. Thus, we will now bound the moment generating function for $T_i^{(L_i)}$.

$$\begin{aligned}
\mathbb{E}\left[e^{t_i T_i^{(L_i)}}\right] &\overset{(a)}{=} \mathbb{E}\left[\max_z \max_{j\in\mathcal{A}_i} e^{t_i p_{ijz}}\right] \\
&= \mathbb{E}_{\mathcal{A}_i}\left[\mathbb{E}\left[\max_z \max_{j\in\mathcal{A}_i} e^{t_i p_{ijz}} \,|\, \mathcal{A}_i\right]\right] \\
&\overset{(b)}{\leq} \mathbb{E}_{\mathcal{A}_i}\left[\sum_{j\in\mathcal{A}_i} \mathbb{E}\left[\max_z e^{t_i p_{ijz}}\right]\right] \\
&= \mathbb{E}_{\mathcal{A}_i}\left[\sum_j F_{ij}\mathbf{1}_{\{j\in\mathcal{A}_i\}}\right] \\
&= \sum_j F_{ij}\,\mathbb{E}_{\mathcal{A}_i}\left[\mathbf{1}_{\{j\in\mathcal{A}_i\}}\right] \\
&= \sum_j F_{ij}\,\mathbb{P}\left(j\in\mathcal{A}_i\right) \\
&\overset{(c)}{=} \sum_j F_{ij}\pi_{ij} \qquad (3.22)
\end{aligned}$$

where (a) follows from (4.21), (b) follows by upper bounding $\max_{j\in\mathcal{A}_i}$ by $\sum_{j\in\mathcal{A}_i}$, (c) follows by probabilistic scheduling where $\mathbb{P}\left(j\in\mathcal{A}_i\right)=\pi_{ij}$, and $F_{ij}=\mathbb{E}\left[\max_z e^{t_i p_{ijz}}\right]$. We note that the only inequality here is for replacing the maximum by the sum. Since this term will be inside the logarithm for the mean stall latency, the gap between the term and its bound becomes additive rather than multiplicative.

Substituting (A.74) in (A.73), we have

$$\mathbb{E}\left[T_i^{(L_i)}\right] \leq \frac{1}{t_i}\log\left(\sum_{j=1}^{m}\pi_{ij}F_{ij}\right). \qquad (3.23)$$

Let $H_{ij}=\sum_{\ell=1}^{L_i} e^{-t_i(d_s+(\ell-1)\tau)}Z_{i,j}^{(\ell)}(t_i)$, where $Z_{i,j}^{(\ell)}(t)$ is defined in equation (4.23). We note that H_{ij} can be simplified using the geometric series formula as follows.

Lemma 3 *Let*

$$H_{ij}=\frac{e^{-t_i(d_s-\tau)}\left(1-\rho_j\right)t_i\widetilde{M}_j(t_i)}{t_i-\Lambda_j\left(B_j(t_i)-1\right)}\frac{1-\left(\widetilde{M}_j(t_i)\right)^{L_i}}{\left(1-\widetilde{M}_j(t_i)\right)}, \qquad (3.24)$$

where $\widetilde{M}_j(t_i)=M_j(t_i)e^{-t_i\tau}$, $M_j(t_i)$ *is given in (4.5), and* $B_j(t_i)$ *is given in (4.12).*

Proof The proof is provided in Appendix A.3. ■

Substituting (4.35) in (A.72) and some manipulations, the mean stall duration is bounded as follows.

Theorem 3.5.1 *The mean stall duration time for file i is bounded by*

$$\mathbb{E}\left[\Gamma^{(i)}\right] \leq \frac{1}{t_i} log\left(\sum_{j=1}^{m} \pi_{ij}\left(1+H_{ij}\right)\right) \tag{3.25}$$

for any $t_i > 0$, $\rho_j = \sum_i \pi_{ij}\lambda_i L_i\left(\beta_j + \frac{1}{\alpha_j}\right)$, $\rho_j < 1$, and
$\sum_{f=1}^{r} \pi_{fj}\lambda_f \left(\frac{\alpha_j e^{-\beta_j t_i}}{\alpha_j - t_i}\right)^{L_f} - (\Lambda_j + t_i) < 0$, $\forall j$.

Proof The proof is provided in Appendix A.4. ■

Note that Theorem A.19.1 above holds only in the range of t_i when $t_i - \Lambda_j\left(B_j(t_i) - 1\right) > 0$ which reduces to $\sum_{f=1}^{r} \pi_{fj}\lambda_f \left(\frac{\alpha_j e^{-\beta_j t_i}}{\alpha_j - t_i}\right)^{L_f} - (\Lambda_j + t_i) < 0$, $\forall i, j$, and $\alpha_j - t_i > 0$. Further, the server utilization ρ_j must be less than 1 for stability of the system.

We note that for the scenario, where the files are downloaded rather than streamed, a metric of interest is the mean download time. This is a special case of our approach when the number of segments of each video is one, or $L_i = 1$. Thus, the mean download time of the file follows as a special case of Theorem A.19.1. We note that the authors of [22, 23] gave an upper bound for mean file download time using probabilistic scheduling. However, the bound in this work is different since we use moment generating function based bound. The two bounds are compared in Section 5.6, and the bounds in this work are shown to outperform those in [22, 23].

3.6 Stall Duration Tail Probability

The stall duration tail probability of a file i is defined as the probability that the stall duration tail $\Gamma^{(i)}$ is greater than (or equal) to x. Since evaluating $\Pr\left(\Gamma^{(i)} \geq x\right)$ in closed-form is hard [22–27], we derive an upper bound on the stall duration tail probability considering Probabilistic Scheduling as follows.

$$\begin{aligned}\Pr\left(\Gamma^{(i)} \geq x\right) &\overset{(a)}{=} \Pr\left(T_i^{(L_i)} \geq x + d_s + (L_i - 1)\,\tau\right) \\ &= \Pr\left(T_i^{(L_i)} \geq \overline{x}\right) \end{aligned} \tag{3.26}$$

where (a) follows from (A.72) and $\overline{x} = x + d_s + (L_i - 1)\,\tau$. Then,

$$\begin{aligned}\Pr\left(T_i^{(L_i)} \geq \overline{x}\right) &\overset{(b)}{=} \Pr\left(\max_z \max_{j\in\mathcal{A}_i} p_{ijz} \geq \overline{x}\right) \\ &= \mathbb{E}_{\mathcal{A}_i, p_{ijz}}\left[\mathbf{1}_{\left(\max\limits_z \max\limits_{j\in\mathcal{A}_i} p_{ijz} \geq \overline{x}\right)}\right] \\ &\overset{(c)}{=} \mathbb{E}_{\mathcal{A}_i, p_{ijz}}\left[\max_{j\in\mathcal{A}_i} \mathbf{1}_{\left(\max\limits_z p_{ijz} \geq \overline{x}\right)}\right] \\ &\overset{(d)}{\leq} \mathbb{E}_{\mathcal{A}_i, p_{ijz}} \textstyle\sum_{j\in\mathcal{A}_i} \mathbf{1}_{\left(\max\limits_z p_{ijz} \geq \overline{x}\right)} \\ &\overset{(e)}{=} \textstyle\sum_j \pi_{ij} \mathbb{E}_{pijz}\left[\mathbf{1}_{\left(\max\limits_z p_{ijz} \geq \overline{x}\right)}\right] \\ &= \textstyle\sum_j \pi_{ij} \mathbb{P}\left(\max\limits_z\, p_{ijz} \geq \overline{x}\right)\end{aligned} \tag{3.27}$$

where (b) follows from (4.21), (c) follows as both max over z and max over $\mathcal{A}_j$ are discrete indicies (quantities) and do not depend on other so they can be exchanged, (d) follows by replacing the max by $\sum_{\mathcal{A}_i}$, (e) follows from probabilistic scheduling. Using Markov Lemma, we get

$$\mathbb{P}\left(\max_z\, p_{ijz} \geq \overline{x}\right) \leq \frac{\mathbb{E}\left[e^{t_i\left(\max\limits_z\, p_{ijz}\right)}\right]}{e^{t_i \overline{x}}} \tag{3.28}$$

We further simplify to get

$$\begin{aligned}\mathbb{P}\left(\max_z\, p_{ijz} \geq \overline{x}\right) &\leq \frac{\mathbb{E}\left[e^{t_i\left(\max\limits_z\, p_{ijz}\right)}\right]}{e^{t_i \overline{x}}} \\ &= \frac{\mathbb{E}\left[\max\limits_z\, e^{t_i p_{ijz}}\right]}{e^{t_i \overline{x}}} \\ &\overset{(f)}{=} \frac{F_{ij}}{e^{t_i \overline{x}}}\end{aligned} \tag{3.29}$$

where (f) follows from (A.3). Substituting (3.29) in (3.27), we get the stall duration tail probability as described in the following theorem (details are provided in Appendix A.5).

Theorem 3.6.1 *The stall distribution tail probability for video file i is bounded by*

$$\sum_j \frac{\pi_{ij}}{e^{t_i x}} \left(1 + e^{-t_i(d_s+(L_i-1)\tau)} H_{ij}\right) \tag{3.30}$$

for any $t_i > 0$, $\rho_j = \sum_i \pi_{ij}\lambda_i L_i \left(\beta_j + \frac{1}{\alpha_j}\right)$, $\rho_j \leq 1$,
$\sum_{f=1}^{r} \pi_{fj}\lambda_f \left(\frac{\alpha_j e^{-\beta_j t_i}}{\alpha_j - t_i}\right)^{L_f} - (\Lambda_j + t_i) < 0$, $\forall i, j$, and H_{ij} is given by (4.42).

We note that for the scenario, where the files are downloaded rather than streamed, a metric of interest is the latency tail probability which is the probability that the file download latency is greater than x. This is a special case of our approach when the number of segments of each video is one, or $L_i = 1$. Thus, the latency tail probability of the file follows as a special case of Theorem 3.6.1. In this special case, the result reduces to that in [29].

3.7 Optimization Problem Formulation and Proposed Algorithm

3.7.1 Problem Formulation

Let $\boldsymbol{\pi} = (\pi_{ij} \forall i = 1, \cdots, r \text{ and } j = 1, \cdots, m)$, $\boldsymbol{\mathcal{S}} = (\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r)$, and $\boldsymbol{t} = \left(\widetilde{t}_1, \widetilde{t}_2, \ldots, \widetilde{t}_r; \overline{t}_1, \overline{t}_2, \ldots, \overline{t}_r\right)$. Note that the values of t_i's used for mean stall duration and the stall duration tail probability can be different and the parameters $\widetilde{t}$ and $\overline{t}$ indicate these parameters for the two cases, respectively. We wish to minimize the two proposed QoE metrics over the choice of scheduling and access decisions. Since this is a multi-objective optimization, the objective can be modeled as a convex combination of the two QoE metrics.

Let $\overline{\lambda} = \sum_i \lambda_i$ be the total arrival rate. Then, $\lambda_i/\overline{\lambda}$ is the ratio of video i requests. The first objective is the minimization of the mean stall duration, averaged over all the file requests, and is given as $\sum_i \frac{\lambda_i}{\overline{\lambda}} \mathbb{E}\left[\Gamma^{(i)}\right]$. The second objective is the minimization of stall duration tail probability, averaged over all the file requests, and is given as $\sum_i \frac{\lambda_i}{\overline{\lambda}} \Pr\left(\Gamma^{(i)} \geq x\right)$. Using the expressions for the mean stall duration and the stall

duration tail probability in Sections A.19 and 3.6, respectively, optimization of a convex combination of the two QoE metrics can be formulated as follows.

$$\min \quad \sum_i \frac{\lambda_i}{\overline{\lambda}} \left[\theta \frac{1}{\widetilde{t}_i} \log \left(\sum_{j=1}^{m} \pi_{ij} \left(1 + \widetilde{H}_{ij} \right) \right) \right.$$

$$\left. + (1-\theta) \sum_j \frac{\pi_{ij}}{e^{\overline{t}_i x}} \left(1 + e^{-\overline{t}_i (d_s + (L_i - 1)\tau)} \, \overline{H}_{ij} \right) \right] \tag{3.31}$$

$$\text{s.t.} \quad \widetilde{H}_{ij} = \frac{e^{-\widetilde{t}_i (d_s - \tau)} \, (1 - \rho_j) \, \widetilde{t}_i}{\widetilde{t}_i - \Lambda_j \left(B_j(\widetilde{t}_i) - 1 \right)} \widetilde{Q}_{ij} \ , \tag{3.32}$$

$$\overline{H}_{ij} = \frac{e^{-\overline{t}_i (d_s - \tau)} \, (1 - \rho_j) \, \overline{t}_i}{\overline{t}_i - \Lambda_j \left(B_j(\overline{t}_i) - 1 \right)} \overline{Q}_{ij} \ , \tag{3.33}$$

$$\widetilde{Q}_{ij} = \left[\frac{\widetilde{M}_j(\widetilde{t}_i) \left(1 - \left(\widetilde{M}_j(\widetilde{t}_i) \right)^{L_i} \right)}{1 - \widetilde{M}_j(\widetilde{t}_i)} \right], \tag{3.34}$$

$$\overline{Q}_{ij} = \left[\frac{\widetilde{M}_j(\overline{t}_i) \left(1 - \left(\widetilde{M}_j(\overline{t}_i) \right)^{L_i} \right)}{1 - \widetilde{M}_j(\overline{t}_i)} \right], \tag{3.35}$$

$$\widetilde{M}_j(t) = \frac{\alpha_j e^{(\beta_j - \tau)t}}{\alpha_j - t}, \tag{3.36}$$

$$B_j(t) = \sum_{f=1}^{r} \frac{\lambda_f \pi_{fj}}{\Lambda_j} \left(\frac{\alpha_j e^{\beta_j t}}{\alpha_j - t} \right)^{L_f} \ , \tag{3.37}$$

$$\widetilde{M}_j(t) = \frac{\alpha_j e^{(\beta_j - \tau)t}}{\alpha_j - t}, \tag{3.38}$$

$$B_j(t) = \sum_{f=1}^{r} \frac{\lambda_f \pi_{fj}}{\Lambda_j} \left(\frac{\alpha_j e^{\beta_j t}}{\alpha_j - t} \right)^{L_f} \ , \tag{3.39}$$

$$\rho_j = \sum_{f=1}^{r} \pi_{fj} \lambda_f L_f \left(\beta_j + \frac{1}{\alpha_j} \right) < 1 \quad \forall j \tag{3.40}$$

$$\Lambda_j = \sum_{f=1}^{r} \lambda_f \pi_{f,j} \quad \forall j \tag{3.41}$$

$$\sum_{j=1}^{m} \pi_{i,j} = k_i \tag{3.42}$$

$$\pi_{i,j}{=}0 \text{ if } j \notin S_i\,, \pi_{i,j} \in [0,1] \tag{3.43}$$

$$|\mathcal{S}_i| = n_i,\ \forall i \tag{3.44}$$

$$0 < \widetilde{t}_i < \alpha_j,\ \forall j \tag{3.45}$$

$$0 < \bar{t}_i < \alpha_j,\ \forall j \tag{3.46}$$

$$\alpha_j \left(e^{(\beta_j - \tau)\widetilde{t}_i} - 1 \right) + \widetilde{t}_i < 0\,, \forall j \tag{3.47}$$

$$\alpha_j \left(e^{(\beta_j - \tau)\bar{t}_i} - 1 \right) + \bar{t}_i < 0\,, \forall j \tag{3.48}$$

$$\sum_{f=1}^{r} \pi_{fj} \lambda_f \left(\frac{\alpha_j e^{\beta_j \widetilde{t}_i}}{\alpha_j - \widetilde{t}_i} \right)^{L_f} - \left(\Lambda_j + \widetilde{t}_i \right) < 0,\ \forall i, j \tag{3.49}$$

$$\sum_{f=1}^{r} \pi_{fj} \lambda_f \left(\frac{\alpha_j e^{\beta_j \bar{t}_i}}{\alpha_j - \bar{t}_i} \right)^{L_f} - \left(\Lambda_j + \bar{t}_i \right) < 0,\ \forall i, j \tag{3.50}$$

$$\text{var.} \qquad \boldsymbol{\pi}, \boldsymbol{t}, \boldsymbol{S} \tag{3.51}$$

Here, $\theta \in [0,1]$ is a trade-off factor that determines the relative significance of mean and tail probability of the stall durations in the minimization problem. Varying $\theta = 0$ to $\theta = 1$, the solution for (4.45) spans the solutions that minimize the mean stall duration to ones that minimize the stall duration tail probability. Note that constraint (4.48) gives the load intensity of server j. Constraint (4.49) gives the aggregate arrival rate Λ_j for each node for the given probabilistic scheduling probabilities π_{ij} and arrival rates λ_i. Constraints (4.51)-(3.44) guarantees that the scheduling probabilities are feasible. Constraints (3.45)-(4.57) ensure that $\widetilde{M}_j(t)$ exist for each $\widetilde{t}_i$ and $\bar{t}_i$. Finally, Constraints (3.49)-(4.58) ensure that the moment generating function given in (4.23) exists. We note that the the optimization over $\boldsymbol{\pi}$ helps decrease the objective function and gives significant flexibility over choosing the lowest-queue servers for accessing the files. The placement of the video files $\boldsymbol{S}$ helps separate the highly accessed files on different servers thus reducing the objective. Finally, the optimization over the

auxiliary variables $\boldsymbol{t}$ gives a tighter bound on the objective function. We note that the QoE for file i is weighed by the arrival rate λ_i in the formulation. However, general weights can be easily incorporated for weighted fairness or differentiated services.

Note that the proposed optimization problem is a mixed integer non-convex optimization as we have the placement over n servers and the constraints (3.49) and (4.58) are non-convex in $(\boldsymbol{\pi}, \boldsymbol{t})$. We also note the placement may be decided for multiple aggregation VMs simultaneously and may not be a parameter for single aggregation VM. In that case, the proposed algorithm can still be used without an optimization over the placement of video files. In the next subsection, we will describe the proposed algorithm.

3.7.2 Proposed Algorithm

The joint mean-tail stall duration optimization problem given in (4.45)-(4.59) is optimized over three set of variables: scheduling probabilities $\boldsymbol{\pi}$, auxiliary parameters $\boldsymbol{t}$, and chunk placement $\boldsymbol{\mathcal{S}}$. Since the problem is non-convex, we propose an iterative algorithm to solve the problem. The proposed algorithm divides the problem into three subproblems that optimize one variable fixing the remaining two. The three sub-problems are labeled as (i) Access Optimization optimizes $\boldsymbol{\pi}$ for given $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{t}$, (ii) Auxiliary Variables Optimization optimizes $\boldsymbol{t}$ for given $\boldsymbol{\pi}$ and $\boldsymbol{\mathcal{S}}$, and (iii) Placement Optimization optimizes $\boldsymbol{\mathcal{S}}$ for given $\boldsymbol{\pi}$ and $\boldsymbol{t}$. This algorithm is summarized as follows.

1. **Initilization**: Initialize $\boldsymbol{t}, \boldsymbol{\mathcal{S}}$, and $\boldsymbol{\pi}$ in the feasible set.
2. **While Objective Converge**
   (a) Run Access Optimization using current values of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{t}$ to get new values of $\boldsymbol{\pi}$
   (b) Run Auxiliary Variables Optimization using current values of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\pi}$ to get new values of $\boldsymbol{t}$

(c) Run Placement Optimization using current values of $\boldsymbol{\pi}$ and $\boldsymbol{t}$ to get new values of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\pi}$.

We first initialize $\mathcal{S}_i$, π_{ij} and t_i $\forall$ i, j such that the choice is feasible for the problem. Then, we do alternating minimization over the three sub-problems defined above. We will describe the three sub-problems along with the proposed solutions for the sub-problems in Appendix A.6. Each of the three sub-problems are solved by iNner cOnVex Approximation (NOVA) algorithm proposed in [57], and is guaranteed to converge to a stationary point. Since each sub-problem converges (decreasing) and the overall problem is bounded from below, we have the following result.

Theorem 3.7.1 *The proposed algorithm converges to a stationary point.*

3.8 Numerical Results

In this section, we evaluate our proposed algorithm for optimization of mean and tail probability of stall duration and show the effect of the trade-off of parameter θ. We first study the two extremes where only either mean stall duration objective or tail stall duration probability is considered. Then, we show the tradeoff between the two QoE metrics based on the trade-off parameter θ.

Table 3.1.: Storage Node Parameters Used in our Simulation (Shift $\beta = 10msec$ and rate α in 1/s)

	Node 1	**Node 2**	**Node 3**	**Node 4**	**Node 5**	**Node 6**
α_j	18.2298	24.0552	11.8750	17.0526	26.1912	23.9059

	Node 7	**Node 8**	**Node 9**	**Node 10**	**Node 11**	**Node 12**
α_j	27.006	21.3812	9.9106	24.9589	26.5288	21.8067

3.8.1 Numerical Setup

We simulate our algorithm in a distributed storage system of $m = 12$ distributed nodes, where each video file uses an $(10, 4)$ erasure code. These parameters were chosen in [23] in the experiments using Tahoe testbed. Further, $(10, 4)$ erasure code is used in HDFS-RAID in Facebook [61] and Microsoft [6]. Unless otherwise explicitly stated, we consider $r = 1000$ files, whose sizes are generated based on Pareto distribution [62] with shape factor of 2 and scale of 300, respectively. We note that the Pareto distribution is considered as it has been widely used in existing literature [63] to model video files, and file-size distribution over networks. We also assume that the chunk service time follows a shifted-exponential distribution with rate α_j and shift β_j, whose values are shown in Table I, which are generated at random and kept fixed for the experiments (Recall that this distribution has been validated in real experiments demonstrated in realistic systems [23, 24]). Unless explicitly stated, the arrival rate for the first 500 files is $0.002s^{-1}$ while for the next 500 files is set to be $0.003s^{-1}$. Chunk size τ is set to be equal to 4 s. When generating video files, the sizes of the video file sizes are rounded up to the multiple of 4 sec. We note that a high load scenario is considered for the numerical results. In practice, the load will not be that high. However, higher load helps demonstrate the significant improvement in performance as compared to the lightly loaded scenarios where there are almost no stalls. In order to initialize our algorithm, we use a random placement of files on all the servers. Further, we set $\pi_{ij} = k/n$ on the placed servers with $t_i = 0.01$ $\forall i$ and $j \in \mathcal{S}_i$. However, these choices of π_{ij} and t_i may not be feasible. Thus, we modify the initialization of $\boldsymbol{\pi}$ to be closest norm feasible solution given above values of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{t}$. We compare our proposed approach with five strategies:

1. *Random Placement, Optimized Access (RP-OA):* In this strategy, the placement is chosen at random where any n out of m servers are chosen for each file, where each choice is equally likely. Given the random placement, the variables $\boldsymbol{t}$ and

$\boldsymbol{\pi}$ are optimized using the Algorithm in Section 5.5.2, where $\boldsymbol{\mathcal{S}}$-optimization is not performed.

2. *Optimized Placement, Projected Equal Access (OP-PEA):* The strategy utilizes $\boldsymbol{\pi}$, $\boldsymbol{t}$ and $\boldsymbol{\mathcal{S}}$ as mentioned in the setup. Then, alternating optimization over placement and $\boldsymbol{t}$ are performed using the proposed algorithm.

3. *Random Placement, Projected Equal Access (RP-PEA):* In this strategy, the placement is chosen at random where any n out of m servers are chosen for each file, where each choice is equally likely. Further, we set $\pi_{ij} = k/n$ on the placed servers with $t_i = 0.01$ $\forall i$ and $j \in \mathcal{S}_i$. We then modify the initialization of $\boldsymbol{\pi}$ to be closest norm feasible solution given above values of $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{t}$. Finally, an optimization over $\boldsymbol{t}$ is performed to the objective using Algorithm (4).

4. OP-PSP (*Optimized Placement-Projected Service-Rate Proportional Allocation*) Policy: The joint request scheduler chooses the access probabilities to be proportional to the service rates of the storage nodes, i.e., $\pi_{ij} = k_i \frac{\mu_j}{\sum_j \mu_j}$. This policy assigns servers proportional to their service rates. These access probabilities are projected toward feasible region for a uniformly random placed files to ensure stability of the storage system. With these fixed access probabilities, the weighted mean stall duration and stall duration tail probability are optimized over the $\mathbf{t}$, and placement $\boldsymbol{\mathcal{S}}$.

5. RP-PSP (*Random Placement-PSP*) Policy: As compared to the OP-PSP Policy, the chunks are placed uniformly at random. The weighted mean stall duration and stall duration tail probability are optimized over the choice of auxiliary variables $\mathbf{t}$.

3.8.2 Mean Download Time Comparison

We note that when the number of segments, L_i, the mean stall duration is the same as the mean download time of the file. Further, the bounds in this

work are different from those given in [22, 23] even though both the works use probabilistic scheduling. We will now compare our proposed upper-bound on download time of a file with the upper-bound given in [22, 23]. The comparison can be seen in Figure 3.4, where the above service time distributions are used at the servers. We observe that our bound performs better for all values of arrival rate (λ), and the relative performance increases with the arrival rate. For instance, our bound is 30% lower than that given in [22, 23] when the arrival rate equals $0.8 \times \lambda$.

3.8.3 Mean Stall Duration optimization

In this subsection, we focus only on minimizing the mean stall duration of all files by setting $\theta = 1$, *i.e.*, stall duration tail probability is not considered.

Convergence of the Proposed Algorithm

Figure 4.3 shows the convergence of our proposed algorithm, which alternatively optimizes the mean stall duration of all files over scheduling probabilities $\boldsymbol{\pi}$, auxiliary variables $\widetilde{\boldsymbol{t}}$, and placement $\boldsymbol{\mathcal{S}}$. We notice that for $r = 1000$ video files of size 600 sec with $m = 12$ storage nodes, the mean stall duration converges to the optimal value within less than 700 iterations.

Effect of Arrival Rate and Video Length

Figure 3.6 shows the effect of different video arrival rates on the mean stall duration for different-size video length.The different size uses the Pareto-distributed lengths described above. We compare our proposed algorithm with the five baseline policies and we see that the proposed algorithm outperforms all baseline strategies for the QoE metric of mean stall duration. Thus, both access and placement of files are both important for the reduction of mean stall duration. Further, we see that the

mean stall duration increases with arrival rates, as expected. Since the mean stall duration is more significant at high arrival rates, we notice a significant improvement in mean stall duration by about 60% (approximately 700s to about 250s) at the highest arrival rate in Figure 3.6 as compared to the random placement and projected equal access policy. In Figure A.7, Appendix A.9, we studied the effect of increasing the arrival rate when the video-sizes are equal with mean of 600 sec.

3.8.4 Stall Duration Tail Probability Optimization

In this subsection, we consider minimizing the stall duration tail probability, $\mathbb{P}\left(\Gamma^{(i)} \geq x\right)$, by setting $\theta = 0$ in (4.45).

Decrease of Stall Duration Tail Probability with x

Figure 3.7 shows the decay of weighted stall duration tail probability with respect to x (in seconds) for the proposed and the baseline strategies. In order to signify (magnify) the small differences, we plot y-axis in logarithmic scale. We observe that the proposed algorithm gives orders improvement in the stall duration tail probabilities as compared to the baseline strategies.

Effect of the number of video files

Figure 3.8 demonstrates the effect of increase of the number of video files (from 200 files to 1200 files whose sizes are defined based on Pareto) on the stall duration tail probability. The stall duration tail probability increases with the number of video files, and the proposed algorithm manages to significantly improve the QoE as compared to the considered baselines.

3.8.5 Tradeoff between mean stall duration and stall duration tail probability

If the mean stall duration decreases, intuitively the stall duration tail probability also reduces. Thus, a question arises whether the optimal point for decreasing the mean stall duration and the stall duration tail probability is the same. We answer the question in negative since for $r = 1000$ of equal sizes of length 300 sec, we find that at the values of $(\boldsymbol{\pi}, \boldsymbol{\mathcal{S}})$ that optimize the mean stall duration, the stall duration tail probability is 12 times higher as compared to the optimal stall duration tail probability. Similarly, the optimal mean stall duration is 30% lower as compared to the mean stall duration at the value of $(\boldsymbol{\pi}, \boldsymbol{\mathcal{S}})$ that optimizes the stall duration tail probability. Thus, an efficient tradeoff point between the QoE metrics can be chosen based on the point on the curve that is appropriate for the clients.

3.9 Chapter Conclusion

This work considers video streaming over cloud where the content is erasure-coded on the distributed servers. Two QoE metrics related to the stall duration, mean stall duration and stall duration tail probability are characterized with upper bounds. The download and play times of each video segment are characterized to evaluate the QoE metrics. An optimization problem that optimizes the convex combination of the two QoE metrics for the choice of placement and access of contents from the servers is formulated. Efficient algorithm is proposed to solve the optimization problem and the numerical results depict the improved performance of the algorithm as compared to the considered baselines.

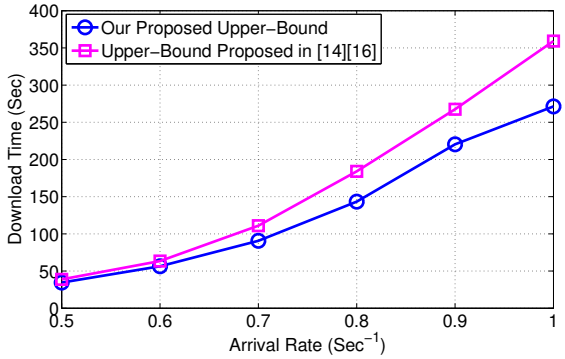

Fig. 3.4.: Comparison between our upper bound on download time and the upper bound proposed in [22, 23].

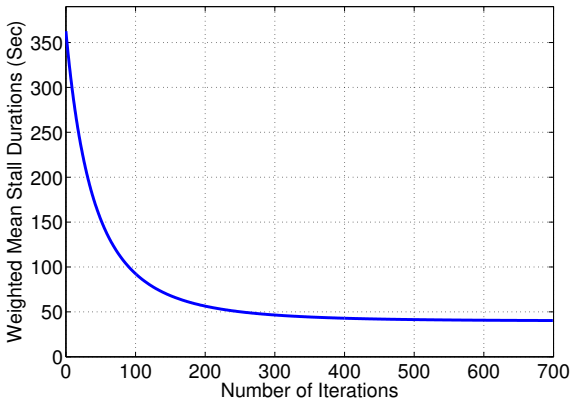

Fig. 3.5.: Convergence of mean stall duration.

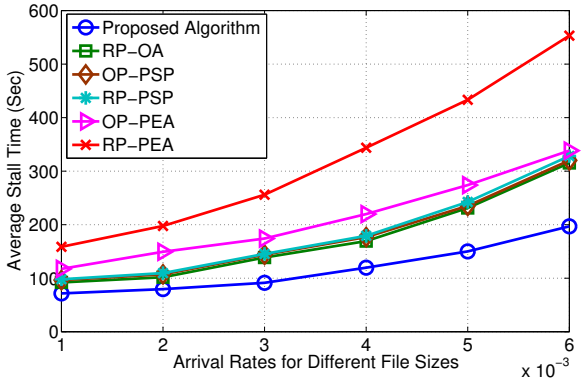

Fig. 3.6.: Mean stall duration for different video arrival rates with different video lengths.

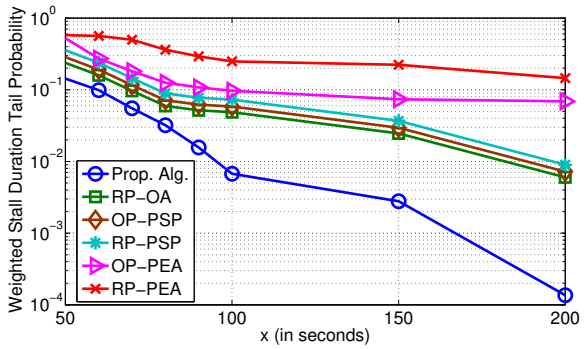

Fig. 3.7.: Stall duration tail probability for different values of x (in seconds).

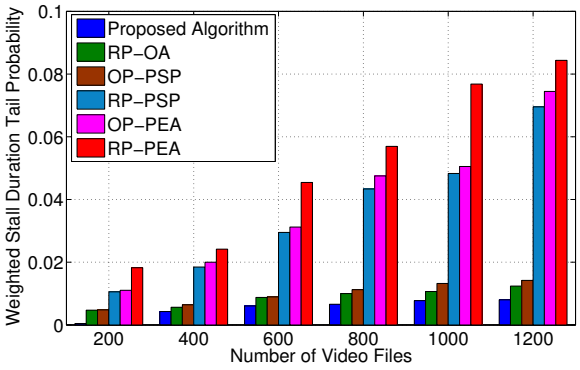

Fig. 3.8.: Stall duration tail probability for varying number of video files ($x = 150$ s).

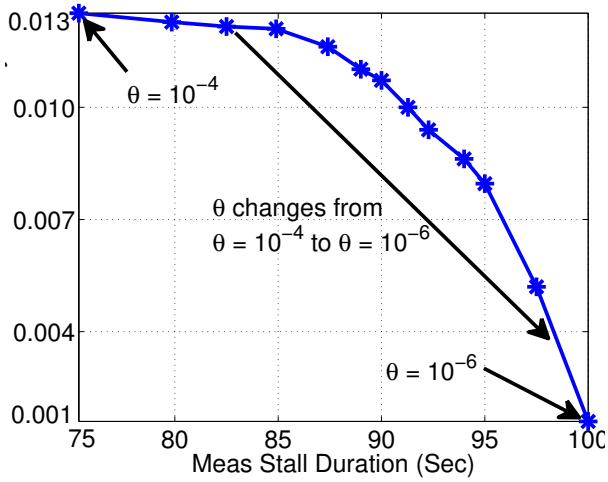

Fig. 3.9.: Tradeoff between mean stall duration and stall duration tail probability obtained by varying θ.

4. STALL-QUALITY TRADEOFF IN VIDEO STREAMING OVER DISTRIBUTED STORAGE SYSTEMS

4.1 Introduction

Cloud computing has changed the way many Internet services are provided and operated. Video-on-Demand (VoD) providers are increasingly moving their streaming services, data storage, and encoding software to cloud service providers [64,65]. With the annual growth of global video streaming at a rate of 18.3% [1], cloud-based video has become an imperative feature of any successful business. For example, IBM estimates cloud-based video will be a $105 billion market opportunity by 2019 [2]. In this chapter, we will give a novel approach to an optimized cloud-based-video streaming.

Since the computing has been growing exponentially [58], the computation of decoding will not limit the latencies in delay sensitive video streaming and the networking latency will govern the system designs. The key advantage of erasure coding is that it reduces storage cost while providing similar reliability as replicated systems [3, 4], and thus has now been widely adopted by companies like Facebook [5], Microsoft [6], and Google [7]. Further, we note that replication is a special case of erasure coding. Thus, the proposed research using erasure-coded content on the servers can also be used when the content is replicated on the servers.

In cloud-based-video, the users are connected to an edge router, which fetch the contents from the distributed storage servers (as depicted in Fig. 4.1). Multiple parallel streams (PSs) between a server and the edge router are considered, which provides the ability to get multiple streams simultaneously. We assume that the connection between users and edge router is not limited. However, our analysis can

be easily generalized to accommodate the last hop between edge-routes and users. Unlike the case of file download, the later video-chunks do not have to be downloaded as fast as possible to improve the QoE and thus multiple parallel streams help achieve better QoE. This is because later chunks can be downloaded while earlier chunks are streamed. The key QoE metrics for video streaming are the duration of stalls at the clients and the streamed average video quality. Every viewer can relate the QoE for watching videos to the stall duration and is thus one of the key focus in the studied streaming algorithms [54, 66]. The average quality of the streamed video is another important QoE metric.

The key challenge in quantification stall duration is the choice of scheduling strategy to choose the storage servers for each request, as well as the parallel streams from the chosen servers. For a single video-chunk and single quality videos, the problem is equivalent to minimizing the download latency Minimizing file download time rather than stall duration follows as a special case of our framework since the stall duration of a single-chunk video is the same as the download time. However, for more than one chunk video, stall duration do not follow from the file download time. This problem of file download is an open problem, since the optimal strategy of choosing these k servers (when file is erasure coded with parameters (n, k)) would need a Markov approach similar to that in [26] which suffers from a state explosion problem. Further, the choice of video quality makes the problem challenging since the selection of video quality would also depend on the current queue states. The authors of [22, 23] proposed a probabilistic scheduling method for file request scheduling, where each possibility of k servers is chosen with a certain probability that can be optimized. In this work, we extend this scheduling approach, in the video context, to a two-stage probabilistic scheduling which chooses k servers and one of the parallel streams from each of these k servers. Further, the choice of video quality is chosen independent of the scheduling and is chosen by a discrete probabilistic distribution. Thus, the proposed scheduling and quality assignment do not account for the current queue state making the approach manageable for analysis. In addition, our

scheduling techniques can assign different weights for different video files (to reflect their importance/popularity) to further prioritize some videos over the other, hence improving the overall QoE metrics.

The data chunk transfer time in practical systems follows a shifted exponential distribution [23, 24] which motivates the choice that the service time distribution for each video server is a shifted exponential distribution. Further, the request arrival rates for each video is assumed to be Poisson. The video segments are encoded using an (n, k) erasure code and the coded segments are placed on n different servers. When a video file is requested, the segments need to be served from k out of n servers as well as one of the parallel streams from each of the k servers. Using the two-stage probabilistic scheduling and probabilistic quality assignment, the random variables corresponding to the download times of different video segments from each server are characterized. By using ordered statistics over the k parallel streams (one from each of the chosen k servers), the random variables corresponding to the playback time of each video segment are obtained. These are then used to find a bound on the mean stall duration. Moment generating functions of the ordered statistics of different random variables are used in the bound. We note that the problem of finding latency for file download is very different from the video stall duration for streaming. This is because the stall duration accounts for download time of each video segment rather than only the download time of the last video segment. Moreover, in video streaming, the download time of segments are correlated since the chunks download times from a server are in sequence and the playback time of a video segment both depend on: the playback time of the last segment and the download time of the current segment. Taking these dependencies into account, this chapter characterizes the bound on the mean stall duration and provides experimental results to show the tightness of this bound.

In this chapter, a convex combination of mean stall duration and average video quality is optimized over the choice of two-stage probabilistic scheduling, video quality assignment, bandwidth allocation among different parallel streams, and the auxiliary

variables in the bounds. Changing the convex combination parameter gives a tradeoff between the mean stall duration and the average video quality. An efficient algorithm is proposed to solve this non-convex problem. The proposed algorithm performs an alternating optimization over the different parameters, where each sub-problem is shown to have convex constraints and thus can be efficiently solved using iNner cOnVex Approximation (NOVA) algorithm proposed in [57]. The proposed algorithm is shown to converge to a stationary point. Based on the offline algorithm, an online version of the algorithm is further developed. Evaluation results demonstrate significant improvement of QoE metrics as compared to the considered baselines and some queue-based online algorithms, e.g., [48], [46], and [47]. The key contributions of our work in this chapter are summarized as follows.

- This chapter proposes a two-stage probabilistic scheduling for the choice of servers and the parallel streams. Further, the video quality is chosen using a discrete probability distribution.

- Two-stage probabilistic scheduling and quality assignment are used to find the distribution of the (random) download time of a chunk of each video segment from a parallel stream. Using ordered statistics, the random variable corresponding to the playback time of each video segment is characterized. This is further used to give bounds on the mean stall duration.

- The QoE metrics of mean stall duration and average video quality are used to formulate an optimization problem over the two-stage probabilistic scheduling access policy, probabilistic quality assignment, the bandwidth allocation weights among the different streams, and the auxiliary bound parameters which are related to the moment generating function. Efficient iterative solutions with low time complexity are provided for these optimization problems. Based on the offline policy, an online algorithm is proposed.

- The experimental results validate our theoretical analysis and demonstrate the efficacy of our proposed algorithm. Further, numerical results show that the

proposed algorithms converge within a few iterations. In addition, the QoE metrics are shown to have significant improvements as compared to the considered baselines and some queue-based policies. Even for the minimum stall point, the proposed algorithm gets better quality than the lowest quality. The tradeoff between stalls and quality can be used by the service provider to effectively find an operating point.

4.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 4.3 presents the system model used in this chapter with a description of video streaming over cloud storage. Section 4.4 derives expressions for the download and play times of the chunks. Section 4.6 formulates the QoE optimization problem as a weighted combination of the two QoE metrics and proposes the iterative algorithmic solution of this problem. Numerical results are provided in Section 4.8. Section 4.8 concludes this chapter with pointing out to the key observations.

4.3 System Model

In this section, we describe the system model and the video encoding parameters. Then, we explain the two-stage probabilistic scheduling and the queuing model.

4.3.1 System Description

We consider a distributed storage system consisting of m heterogeneous servers (also called storage nodes), denoted by $\mathcal{M} = 1, 2, \cdots, m$. Each server j can be split into d_j virtual outgoing parallel streams (queues) to the edge router, where the server bandwidth is split among all d_j parallel streams (PSs). This is depicted in Fig. 4.1. The reason of having d_j PSs is to serve d_j video files simultaneously from a server thus helping one file not to have files wait for the previous long video

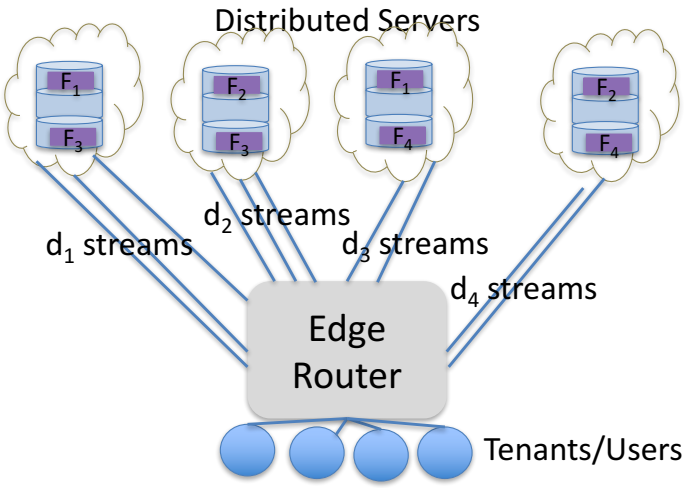

Fig. 4.1.: An Illustration of a distributed storage system equipped with $m = 4$ nodes. Storage server j has d_j streams to the edge router.

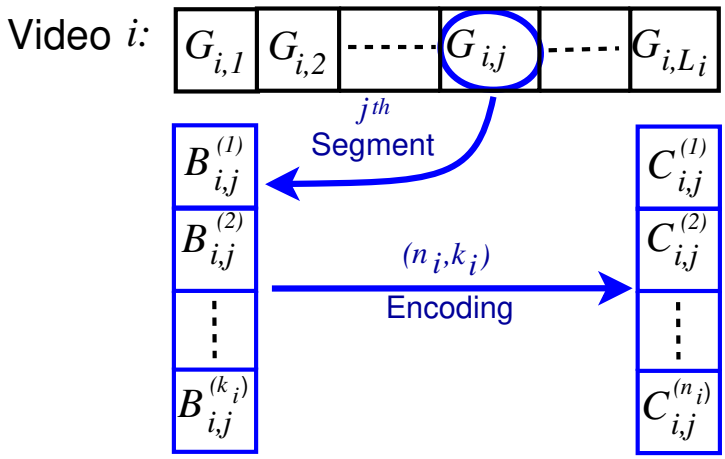

Fig. 4.2.: A schematic illustrates video fragmentation and erasure-coding processes. Video i is composed of L_i segments. Each segment is partitioned into k_i chunks and then encoded using an (n_i, k_i) MDS code. The quality index is omitted in the figure for simplicity.

files. This is a key difference for video streaming as compared to file download since the deadline for the later video chunks are late thus motivating prioritizing earlier chunks. This parallelization helps download multiple files in parallel which also delays the finishing of download of the last chunks of multiple requests. Multiple users are connected to edge-router, where we assume that the connection between user and edge router is infinite and thus only consider the links from the server to the edge router. Thus, we can consider edge router as an aggregation of multiple users. Let $\left\{w_{j,\nu_j}, \forall j = 1, \cdots, m, \nu_j = 1, \cdots d_j\right\}$ be a set of d_j non-negative weights representing the split of bandwidth at server j on the d_j PSs. The weights satisfy $\sum_{\nu_j=1}^{d_j} w_{j,\nu_j} \leq 1 \forall j$. The sum of weights at all PSs can be smaller than 1, representing that the bandwidth may not be completely utilized. By optimizing w_{j,ν_j}, the server bandwidth can be efficiently split among different PSs. Optimizing these weights help avoid bandwidth under-utilization and congestion, for example, assigning larger bandwidth to heavy workload PSs can help reduce mean stall duration.

Each video file i, where $i = 1, 2, \cdots, r$, is divided into L_i equal segments, each of length τ seconds. We assume that each video file is encoded to different qualities, *i.e.*, $\ell \in \{1, 2, \cdots, V\}$, where V are the number of possible choices for the quality level[1]. The L_i segments of video file i at quality ℓ are denoted as $G_{i,\ell,1}, \cdots, G_{i,\ell,L_i}$. Then, each segment $G_{i,\ell,u}$ for $u \in \{1, 2, \ldots, L_i\}$ and $\ell \in \{1, 2, \cdots, V\}$ is partitioned into k_i fixed-size chunks and then encoded using an (n_i, k_i) Maximum Distance Separable (MDS) erasure code to generate n_i distinct chunks for each segment $G_{i,\ell,u}$. These coded chunks are denoted as $C^{(1)}_{i,\ell,u}, \cdots, C^{(n_i)}_{i,\ell,u}$. The encoding setup is illustrated in Figure 4.2. The encoded chunks for all quality levels are stored on the disks of n_i distinct storage nodes. The storage nodes chosen for quality level ℓ are represented by a set $\mathcal{S}^{(\ell)}_i$, such that $\mathcal{S}^{(\ell)}_i \subseteq \mathcal{M}$ and $n_i = \left|\mathcal{S}^{(\ell)}_i\right|$. Each server $z \in \mathcal{S}^{(\ell)}_i$ stores all the chunks $C^{(g)}_{i,\ell,u}$ for all u and for some g. In other words, n_i servers store the entire content, where a server stores coded chunk g for all the video-chunks for some g or does not store any chunk. We will use a probabilistic quality assignment strategy, where a chunk of quality ℓ of size a_ℓ is requested with probability $b_{i,\ell}$ for all $\ell \in \{1, 2, \cdots, V\}$. We further assume all the chunks of the video are fetched at the same quality level. The decision variable of choosing the quality takes into account the network's congestion, link capacity and video weights/popularities. Note that $k_i = 1$ indicates that the video file i is replicated n_i times.

4.3.2 Two-stage Probabilistic Scheduling

In order to serve the incoming request at the edge router, the video can be reconstructed from the video chunks from any subset of k_i-out-of-n_i servers. Further, we need to assign one of the d_j PSs for each server j that is selected. We assume that files at each PS are served in order of the request in a FIFO policy. However, the

[1]While a constant bitrate (quality) is assumed for each video i, our algorithm will first determine whether the downloading of the highest resolution/quality can be supported by the link capacity (resources) with an acceptable downloading delay and/or stalls. If yes, the video would be downloaded and played at the highest quality; otherwise, it would consider a further optimization for the video content with the next lower bitrate/quality.

proposed framework and the scheduling policies remain applicable for other queuing disciplines as long as the waiting time in the queue is characterized (see reference [67] for priority queuing based models). Further, the different video chunks in a video are processed in order. In order to select the different PSs for video i and quality ℓ, the request goes to a set $\mathcal{A}_i^{(\ell)} = \{(j, \nu_j) : j \in \mathcal{S}_i^{(\ell)}, \nu_j \in \{1, \cdots, d_j\}\}$, with $\left|\mathcal{A}_i^{(\ell)}\right| = k_i$ and for every (j, ν_j) and (k, ν_k) in $\mathcal{A}_i^{(\ell)}$, $j \neq k$. Here, the choice of j represents the server to choose and ν_j represents the PS selected. From each choice $(j, \nu_j) \in \mathcal{A}_i^{(\ell)}$, all chunks $C_{i,\ell,u}^{(g)}$ for all u and the value of g corresponding to that placed on server j are requested from PS ν_j. The choice of optimal scheduling strategy, or set $\mathcal{A}_i^{(\ell)}$ is an open problem. In this part, we extend the probabilistic scheduling proposed in [22, 23] to two-stage probabilistic scheduling. The two-stage probabilistic scheduling chooses every possible subset of k_i-out-of-n_i nodes with certain probability, and for every chosen node j, chooses 1-out-of-d_j PSs with certain probability. Let $\pi_{i,j,\nu_j}^{(\ell)}$ is the probability of requesting file i from the PS ν_j that belongs to server j for quality level ℓ. Thus, $\pi_{i,j,\nu_j}^{(\ell)}$ is given by

$$\pi_{i,j,\nu_j}^{(\ell)} = q_{i,j}^{(\ell)} p_{j,\nu_j}^{(\ell)}, \tag{4.1}$$

where $q_{i,j}^{(\ell)}$ is the probability of choosing server j and $p_{j,\nu}^{(\ell)}$ is the probability of choosing PS ν_j at server j. Following [22, 23], it can be seen that the two-stage probabilistic scheduling gives feasible probabilities for choosing k_i-out-of n_i nodes and one-out-of-d_j PSs if and only if there exists conditional probabilities $q_{i,j}^{(\ell)} \in [0, 1]$ and $p_{j,\nu_j}^{(\ell)} \in [0, 1]$ satisfying

$$\sum_{j=1}^{m} q_{i,j}^{(\ell)} = k_i \quad \forall i \qquad \text{and} \qquad q_{i,j}^{(\ell)} = 0 \quad \text{if } j \notin \mathcal{S}_i^{(\ell)}, \tag{4.2}$$

and

$$\sum_{\nu_j=1}^{d_j} p_{j,\nu_j}^{(\ell)} = 1 \quad \forall j. \tag{4.3}$$

4.3.3 Queueing Model

We now describe a queuing model of the distributed storage system. We assume that the arrival of requests at the edge router for each video i form an independent Poisson process with a known rate λ_i. Using the two stage probabilistic scheduling and the quality assignment probability distribution, the arrival of file requests at PS ν_j at node j forms a Poisson Process with rate $\Lambda_{j,\nu_j} = \sum_{i,\ell} \lambda_i \pi^{(\ell)}_{i,j,\nu_j} b_{i,\ell}$ which is the superposition of rd_j Poisson processes each with rate $\lambda_i \pi^{(\ell)}_{i,j,\nu_j} b_{i,\ell}$. We assume that the chunk service time for each coded chunk $C^{(g)}_{i,\ell,u}$ at PS ν_j of server j, $X^{(\ell)}_{j,\nu_j}$, follows a shifted exponential distribution as has been demonstrated in realistic systems [23, 24] and is given by the probability distribution function $f^{(\ell)}_{j,\nu_j}(x)$, which is

$$f^{(\ell)}_{j,\nu_j}(x) = \begin{cases} \alpha^{(\ell)}_{j,\nu_j} e^{-\alpha^{(\ell)}_{j,\nu_j}\left(x-\beta^{(\ell)}_{j,\nu_j}\right)}, & x \geq \beta^{(\ell)}_{j,\nu_j} \\ 0, & x < \beta^{(\ell)}_{j,\nu_j} \end{cases} . \tag{4.4}$$

where $\beta^{(\ell)}_{j,\nu_j}$ represents the shift of quality ℓ at server j and PS ν_j, while $\alpha^{(\ell)}_{j,\nu_j}$ represents the rate of the exponential random part of a video streamed at quality ℓ if it is streamed from server j and PS ν_j. We note that exponential distribution is a special case with $\beta^{(\ell)}_{j,\nu_j} = 0$. Let $M^{(\ell)}_{j,\nu_j}(t) = \mathbb{E}\left[e^{tX^{(\ell)}_{j,\nu_j}}\right]$ be the moment generating function of $X^{(\ell)}_{j,\nu_j}$ whose quality is ℓ. Then, $M^{(\ell)}_{j,\nu_j}(t)$ is given as

$$M^{(\ell)}_{j,\nu_j}(t) = \frac{\alpha^{(\ell)}_{j,\nu_j}}{\alpha^{(\ell)}_{j,\nu_j} - t} e^{\beta^{(\ell)}_{j,\nu_j} t} \qquad t < \alpha^{(\ell)}_{j,\nu_j} \tag{4.5}$$

Note that the value of $\beta^{(\ell)}_{j,\nu_j}$ increases in proportion to the chunk size, and the value of $\alpha^{(\ell)}_{j,\nu_j}$ decreases in proportion to the chunk size in the shifted-exponential service time distribution. Further, the rate $\alpha^{(\ell)}_{j,\nu_j}$ is proportional to the assigned bandwidth w_{j,ν_j}. More formally, the parameters $\alpha^{(\ell)}_{j,\nu_j}$ and $\beta^{(\ell)}_{j,\nu_j}$ are given as

$$\alpha^{(\ell)}_{j,\nu_j} = \alpha_j w_{j,\nu_j}/a_\ell, \ \beta^{(\ell)}_{j,\nu_j} = \beta_j a_\ell, \tag{4.6}$$

where α_j and β_j are constant service time parameters when $a_\ell = 1$ and the entire bandwidth is allocated to one PS. Since $\beta^{(\ell)}_{j,\nu_j}$ mainly represents the read time and other processing times, we assume that all PSs have the same value of $\beta^{(\ell)}_{j,\nu_j}$.

We note that the arrival rates are given for video files, and the service rate above is provided in terms of the coded chunks at each server. Also, the client plays the video segment after all the k_i chunks for the segment have been downloaded and the previous segment has been played. We also assume that there is a start-up delay of d_s (in seconds) for the video which is the duration in which the content for the first chunk can be buffered but not played. This part will characterize the mean stall duration using two-stage probabilistic scheduling and probabilistic quality assignment.

4.4 Download and Play Times of the Chunks

In this section, we derive the expressions for download and play times. In order to understand the stall duration, we need first to get the download time of different coded chunks. Then, the play time of the different segments of the video can be obtained accordingly.

4.4.1 Download Times of the Chunks from each Server

In this subsection, we will quantify the download time of chunk for video file i from server j which has chunks $C_{i,\ell,u}^{(g)}$ for all $u = 1, \cdots L_i$. The download of $C_{i,\ell,u}^{(g)}$ consists of two components - the waiting time of the video files in the queue of the PS before file i request and the service time of all chunks of video file i up to the g^{th} chunk. Let W_{j,ν_j} be the random variable corresponding to the waiting time of all the video files in queue of PS ν_j at server j before file i request and $Y_{j,\nu_j}^{(g,\ell)}$ be the (random) service time of coded chunk g for file i with quality ℓ from PS ν_j at server j. Then, the (random) download time for coded chunk $u \in \{1, \cdots, L_i\}$ for file i at PS ν_j at server $j \in \mathcal{A}_i^{(\ell)}$, $D_{i,j}^{(u,\ell)}$, is given as

$$D_{i,j,\nu_j}^{(u,\ell)} = W_{j,\nu_j} + \sum_{v=1}^{u} Y_{j,\nu_j}^{(v,\ell)}. \tag{4.7}$$

We will now find the distribution of W_{j,ν_j}. We note that this is the waiting time for the video files whose arrival rate is given as $\Lambda_{j,\nu_j} = \sum_{i,\ell} \lambda_i b_{i,\ell} \pi_{i,j,\nu_j}^{(\ell)}$. In order to

find the waiting time, we would need to find the service time statistics of the video files. Note that $f_{j,\nu_j}^{(\ell)}(x)$ gives the service time distribution of only a chunk and not of the video files.

Video file i of quality ℓ consists of L_i coded chunks at PS ν_j at server j ($j \in \mathcal{S}_i^{(\ell)}$). The total service time for video file i with quality ℓ at PS ν_j at server j if requested from server j, $ST_{i,j,\nu_j}^{(\ell)}$, is given as

$$ST_{i,j,\nu_j}^{(\ell)} = \sum_{v=1}^{L_i} Y_{j,\nu_j}^{(v,\ell)}. \tag{4.8}$$

The service time of the video files is given as

$$R_{j,\nu_j} = \begin{cases} ST_{i,j,\nu_j}^{(\ell)} & \text{with probability } \frac{\pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell}}{\Lambda_{j,\nu_j}} \quad \forall i, \ell, j, \nu_j \end{cases} \tag{4.9}$$

since the service time is $ST_{i,j,\nu_j}^{(\ell)}$ when file i is requested at quality ℓ from PS ν_j from server j. Let $\overline{R}_{j,\nu_j}(s) = \mathbb{E}[e^{-sR_{j,\nu_j}}]$ be the Laplace-Stieltjes Transform of R_{j,ν_j}.

Lemma 4 *The Laplace-Stieltjes Transform of* R_{j,ν_j}, $\overline{R}_{j,\nu_j}(s) = \mathbb{E}\left[e^{-s\overline{R}_{j,\nu_j}}\right]$ *is given as*

$$\overline{R}_{j,\nu_j}(s) = \sum_{i=1}^{r} \sum_{\ell=1}^{V} \frac{\pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell}}{\Lambda_{j,\nu_j}} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{-\beta_{j,\nu_j}^{(\ell)} s}}{\alpha_{j,\nu_j}^{(\ell)} + s} \right)^{L_i} \tag{4.10}$$

Proof

$$\begin{aligned}
\overline{R}_{j,\nu_j}(s) &= \sum_{i=1}^{r} \sum_{\ell=1}^{V} \frac{\pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell}}{\Lambda_{j,\nu_j}} \mathbb{E}\left[e^{-s\left(ST_{i,j,\nu_j}^{(\ell)}\right)}\right] \\
&= \sum_{i=1}^{r} \sum_{\ell=1}^{V} \frac{\pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell}}{\Lambda_{j,\nu_j}} \mathbb{E}\left[e^{-s\left(\sum_{\nu=1}^{L_i} Y_{j,\nu_j}^{(\nu,\ell)}\right)}\right] \\
&= \sum_{i=1}^{r} \sum_{\ell=1}^{V} \frac{\pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell}}{\Lambda_{j,\nu_j}} \left(\mathbb{E}\left[e^{-s\left(Y_{j,\nu_j}^{(1,\ell)}\right)}\right]\right)^{L_i} \\
&= \sum_{i=1}^{r} \sum_{\ell=1}^{V} \frac{\pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell}}{\Lambda_{j,\nu_j}} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{-\beta_{j,\nu_j}^{(\ell)} s}}{\alpha_{j,\nu_j}^{(\ell)} + s} \right)^{L_i}
\end{aligned} \tag{4.11}$$

■

Note that the service times of chunks are independent. Since we have L_i chunks for video file i, the MGF of the service time of video file i is the product of the MGF of chunk service times, as shown in (4.11).

Corollary 2 *The moment generating function for the service time of video files when requested from server j and PS ν_j, $B_{j,\nu_j}(t)$, is given as*

$$B_{j,\nu_j}(t) = \sum_{i=1}^{r} \sum_{\ell=1}^{V} \frac{\pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell}}{\Lambda_{j,\nu_j}} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{\beta_{j,\nu_j}^{(\ell)} t}}{\alpha_{j,\nu_j}^{(\ell)} - t} \right)^{L_i} \tag{4.12}$$

for any $t > 0$, and $t < \alpha_{j,\nu_j}$.

Proof This corollary follows from (4.10) by setting $t = -s$. ■

The server utilization for the video files at PS ν_j of server j is given as $\rho_{j,\nu_j} = \Lambda_{j,\nu_j} \mathbb{E}\left[R_{j,\nu_j}\right]$. Since $\mathbb{E}\left[R_{j,\nu_j}\right] = B'_{j,\nu_j}(0)$, using Lemma 4, we have

$$\rho_{j,\nu_j} = \sum_{i=1}^{r} \sum_{\ell=1}^{V} \pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell} L_i \left(\beta_{j,\nu_j}^{(\ell)} + \frac{1}{\alpha_{j,\nu_j}^{(\ell)}} \right). \tag{4.13}$$

Having characterized the service time distribution of the video files via a Laplace-Stieltjes Transform $\overline{R}_{j,\nu_j}(s)$, the Laplace-Stieltjes Transform of the waiting time W_{j,ν_j} can be characterized using Pollaczek-Khinchine formula for M/G/1 queues [59], since the request pattern is Poisson and the service time is general distributed. Thus, the Laplace-Stieltjes Transform of the waiting time W_{j,ν_j} is given as

$$\mathbb{E}\left[e^{-sW_{j,\nu_j}}\right] = \frac{\left(1 - \rho_{j,\nu_j}\right) s}{s - \Lambda_{j,\nu_j}\left(1 - \overline{R}_{j,\nu_j}(s)\right)} \tag{4.14}$$

By characterizing the Laplace-Stieltjes Transform of the waiting time W_{j,ν_j} and knowing the distribution of $Y_{j,\nu_j}^{(v,\ell)}$, the Laplace-Stieltjes Transform of the download time $D_{i,j,\nu_j}^{(u,\ell)}$ is given as

$$\mathbb{E}[e^{-sD_{i,j,\nu_j}^{(u,\ell)}}] = \frac{\left(1 - \rho_{j,\nu_j}\right) s}{s - \Lambda_{j,\nu_j}\left(1 - \overline{R}_{j,\nu_j}(s)\right)} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{-\beta_{j,\nu_j}^{(\ell)} s}}{\alpha_{j,\nu_j}^{(\ell)} + s} \right)^{u}. \tag{4.15}$$

We note that the expression above holds only in the range of s when $s - \Lambda_{j,\nu_j}\left(1 - \overline{R}_{j,\nu_j}(s)\right) > 0$ and $\alpha_{j,\nu_j}^{(\ell)} + s > 0$. Further, the server utilization ρ_{j,ν_j} must be less than 1. The overall download time of all the chunks for the segment $G_{i,u,\ell}$ at the client, $D_i^{(u,\ell)}$, is given by

$$D_i^{(u,\ell)} = \max_{(j,\nu_j)\in\mathcal{A}_i} D_{i,j,\nu_j}^{(u,\ell)}. \tag{4.16}$$

4.4.2 Play Time of Each Video Segment

Let $T_i^{(u,\ell)}$ be the time at which the segment $G_{i,\ell,u}$ is played (started) at the client. The startup delay of the video is d_s. Then, the first segment can be played at the maximum of the time the first segment can be downloaded and the startup delay. Thus,

$$T_i^{(1,\ell)} \;=\; \max\left(d_s,\, D_i^{(1,\ell)}\right). \tag{4.17}$$

For $1 < u \leq L_i$, the play time of segment u of file i is given by the maximum of the time it takes to download the segment and the time at which the previous segment is played plus the time to play a segment (τ seconds). Thus, the play time of segment u of file i, $T_i^{(u,\ell)}$ can be expressed as

$$T_i^{(u,\ell)} \;=\; \max\left(T_i^{(u-1,\ell)} + \tau,\, D_i^{(u,\ell)}\right). \tag{4.18}$$

Equation (4.18) gives a recursive equation, which can yield

$$\begin{aligned} T_i^{(L_i,\ell)} &= \max\left(T_i^{(L_i-1,\ell)} + \tau,\, D_i^{(L_i,\ell)}\right) \\ &= \max\left(T_i^{(L_i-2,\ell)} + 2\tau,\, D_i^{(L_i-1,\ell)} + \tau,\, D_i^{(L_i,\ell)}\right) \\ &= \max\left(\mathcal{F}_{j,1,\nu_j,\ell}, \max_{z=2}^{L_i+1} D_i^{(z-1,\ell)} + (L_i - z + 1)\tau\right) \end{aligned} \tag{4.19}$$

where

$$\mathcal{F}_{j,z,\nu_j,\ell} = \begin{cases} d_s + (L_i - 1)\,\tau & ,\ z = 1 \\ & \\ D_{i,j,\nu_j}^{(z-1,\ell)} + (L_i - z + 1)\tau & ,\ 2 \le z \le (L_i + 1) \end{cases} . \tag{4.20}$$

Since $D_i^{(u,\ell)} = \max_{(j,\nu_j)\in\mathcal{A}_i^{(\ell)}}^{(\ell)} D_{i,j,\nu_j}^{(u,\ell)}$ from (4.16), $T_i^{(L_i,\ell)}$ can be written as

$$T_i^{(L_i,\ell)} = \max_{z=1}^{L_i+1} \max_{(j,\nu_j)\in\mathcal{A}_i} \left(\mathcal{F}_{j,z,\nu_j,\ell}\right). \tag{4.21}$$

We next give the moment generating function of $\mathcal{F}_{j,z,\nu_j,\ell}$ that will be used in the calculations of the mean stall duration in the next section.

Lemma 5 *The moment generating function for $\mathcal{F}_{j,z,\nu_j,\ell}$, is given as*

$$\mathbb{E}\left[e^{t\mathcal{F}_{j,z,\nu_j,\ell}}\right] = \begin{cases} e^{t(d_s+(L_i-1)\tau)} & ,\ z = 1 \\ e^{t(L_i+1-z)\tau} Z_{D_{i,j,\nu_j}^{(z-1,\ell)}}(t) & ,\ 2 \le z \le L_i + 1 \end{cases} \tag{4.22}$$

where

$$Z_{D_{i,j,\nu_j}^{(u,\ell)}}(t) = \mathbb{E}[e^{tD_{i,j,\nu_j}^{(u,\ell)}}] = \frac{\left(1-\rho_{j,\nu_j}\right) t_i \left(M_{j,\nu_j}^{(\ell)}(t_i)\right)^u}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i) - 1\right)} \tag{4.23}$$

Proof This follows by substituting $t = -s$ in (4.15) and $B_{j,\nu_j}(t_i)$ is given by (4.12) and $M_{j,\nu_j}^{(\ell)}(t_i)$ is given by (4.5). This expression holds when $t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i) - 1\right) > 0$ and $t_i < 0\,\forall j, \nu_j$, since the moment generating function does not exist if the above do not hold. ■

Ideally, the last segment should have started played by time $d_s + (L_i - 1)\tau$. The difference between $T_i^{(L_i,\ell)}$ and $d_s + (L_i - 1)\tau$ gives the stall duration. We note that $T_i^{(L_i,\ell)}$ is not the download time of the last segment, but the play time of the last segment and accounts for the download of all the L_i segments. This is a key difference as compared to the file download since the download time of each segment of the

video has to be accounted for computing stall duration. Also, note that the stalls may occur before any segment. This difference will give the sum of durations of all the stall periods before any segment. Thus, the stall duration for the request of video file i of quality ℓ, i.e., $\Gamma^{(i,\ell)}$, is given as

$$\Gamma^{(i,\ell)} = T_i^{(L_i,\ell)} - d_s - (L_i - 1)\tau. \tag{4.24}$$

In the next section, we will use this stall time to determine the bound on the mean stall duration of the streamed video.

4.5 Mean Stall Duration

In this section, we will provide a bound on the mean stall duration for a file i. We will find the bound by two-stage probabilistic scheduling and since this scheduling is one feasible strategy, the obtained bound is an upper bound to the optimal strategy. Using (4.24), the expected stall time for file i is given as follows

$$\begin{aligned} \mathbb{E}\left[\Gamma^{(i,\ell)}\right] &= \mathbb{E}\left[T_i^{(L_i,\ell)} - d_s - (L_i - 1)\,\tau\right] \\ &= \mathbb{E}\left[T_i^{(L_i,\ell)}\right] - d_s - (L_i - 1)\,\tau \end{aligned} \tag{4.25}$$

Exact evaluation for the play time of segment L_i is hard due to the dependencies between $\mathcal{F}_{j,z,\nu_j,\ell}$ random variables for different values of j, ν_j, z, and ℓ, where $z \in (1, 2, ..., L_i + 1)$ and $(j, \nu_j) \in \mathcal{A}_i^{(\ell)}$. Hence, we derive an upper-bound on the playtime of the segment L_i as follows. Using Jensen's inequality [60], we have for $t_i > 0$,

$$e^{t_i \mathbb{E}\left[T_i^{(L_i,\ell)}\right]} \leq \mathbb{E}\left[e^{t_i T_i^{(L_i,\ell)}}\right]. \tag{4.26}$$

Thus, finding an upper bound on the moment generating function for $T_i^{(L_i,\ell)}$ can lead to an upper bound on the mean stall duration. Thus, we will now bound the moment generating function for $T_i^{(L_i,\ell)}$.

$$\mathbb{E}\left[e^{t_i T_i^{(L_i,\ell)}}\right] \overset{(a)}{=} \mathbb{E}\left[\max_{z} \max_{(j,\nu_j)\in\mathcal{A}_i^{(\ell)}} e^{t_i \mathcal{F}_{j,z,\nu_j,\ell}}\right] \tag{4.27}$$

$$= \mathbb{E}_{\mathcal{A}_i^{(\ell)}}\left[\mathbb{E}\left[\max_{z} \max_{(j,\nu_j)\in\mathcal{A}_i{}^{(\ell)}} e^{t_i \mathcal{F}_{j,z,\nu_j,\ell}} \middle| \mathcal{A}_i^{(\ell)}\right]\right] \tag{4.28}$$

$$\overset{(b)}{\leq} \mathbb{E}_{\mathcal{A}_i^{(\ell)}}\left[\sum_{(j,\nu_j)\in\mathcal{A}_i^{(\ell)}} \mathbb{E}\left[\max_{z} e^{t_i \mathcal{F}_{j,z,\nu_j,\ell}}\right]\right] \tag{4.29}$$

$$= \mathbb{E}_{\mathcal{A}_i^{(\ell)}}\left[\sum_{(j,\nu_j)} F_{i,j,\nu_j,\ell} \mathbf{1}_{\left\{(j,\nu_j)\in\mathcal{A}_i^{(\ell)}\right\}}\right] \tag{4.30}$$

$$= \sum_{(j,\nu_j)} F_{i,j,\nu_j,\ell}\, \mathbb{E}_{\mathcal{A}_i^{(\ell)}}\left[\mathbf{1}_{\left\{(j,\nu_j)\in\mathcal{A}_i^{(\ell)}\right\}}\right] \tag{4.31}$$

$$= \sum_{(j,\nu_j)} F_{i,j,\nu_j,\ell}\, \mathbb{P}\left((j,\nu_j)\in\mathcal{A}_i^{(\ell)}\right) \tag{4.32}$$

$$\overset{(c)}{=} \sum_{j=1}^{m}\sum_{\nu_j=1}^{d_j} F_{i,j,\nu_j,\ell}\pi_{i,j,\nu_j}^{(\ell)} \tag{4.33}$$

where (a) follows from (4.21), (b) follows by upper bounding $\max_{(j,\nu_j)\in\mathcal{A}_i^{(\ell)}}$ by $\sum_{(j,\nu_j)\in\mathcal{A}_i^{(\ell)}}$, (c) follows by two-stage probabilistic scheduling where $\mathbb{P}\left((j,\nu_j)\in\mathcal{A}_i^{(\ell)}\right) = \pi_{i,j,\nu_j}^{(\ell)}$, and $F_{i,j,\nu_j,\ell} \triangleq \mathbb{E}\left[\max_{z} e^{t_i \mathcal{F}_{i,z,\nu_j,\ell}}\right]$. Recall that this choice is feasible as illustrated in [22,23]. We note that the only inequality here is for replacing the maximum by the sum. Since this term will be inside the logarithm for the mean stall latency, the gap between the term and its bound becomes additive rather than multiplicative.

To use the bound (A.74), $F_{i,j,\nu_j,\ell}$ needs to be bounded too. Thus, an upper bound on $F_{i,j,\nu_j,\ell}$ is calculated as follows.

$$\begin{aligned}
F_{i,j,\nu_j,\ell} &= \mathbb{E}\left[\max_z e^{t_i \mathcal{F}_{j,z,\nu_j,\ell}}\right] \\
&\overset{(d)}{\leq} \sum_z \mathbb{E}\left[e^{t_i \mathcal{F}_{j,z,\nu_j,\ell}}\right] \\
&\overset{(e)}{=} e^{t_i(d_s+(L_i-1)\tau)} + \sum_{z=2}^{L_i+1} \frac{e^{t_i(L_i-z+1)\tau}\left(1-\rho_{j,\nu_j}\right)t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{t_i \beta_{j,\nu_j}^{(\ell)}}}{\alpha_{j,\nu_j}^{(\ell)} - t_i}\right)^{z-1} \\
&\overset{(f)}{=} e^{t_i(d_s+(L_i-1)\tau)} + \sum_{v=1}^{L_i} \frac{e^{t_i(L_i-v)\tau}\left(1-\rho_{j,\nu_j}\right)t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{t_i \beta_{j,\nu_j}^{(\ell)}}}{\alpha_{j,\nu_j}^{(\ell)} - t_i}\right)^{v} \qquad (4.34)
\end{aligned}$$

where (d) follows by bounding the maximum by the sum, (e) follows from (4.22), and (f) follows by substituting $v = z - 1$.

Substituting (A.74) in (A.73), we have

$$\mathbb{E}\left[T_i^{(L_i,\ell)}\right] \leq \frac{1}{t_i}\log\left(\sum_{j=1}^{m}\sum_{\nu_j=1}^{d_j} \pi_{i,j,\nu_j}^{(\ell)} F_{i,j,\nu_j,\ell}\right). \qquad (4.35)$$

Further, substituting the bounds (A.3) and (4.35) in (A.72), the mean stall duration is bounded as follows.

$$\begin{aligned}
\mathbb{E}\left[\Gamma^{(i,\ell)}\right] &\leq \tfrac{1}{t_i}\log\left(\textstyle\sum_{j=1}^{m}\sum_{\nu_j=1}^{d_j} \pi_{i,j,\nu_j}^{(\ell)} \left(e^{t_i(d_s+(L_i-1)\tau)} + \sum_{v=1}^{L_i} e^{t_i(L_i-\nu)\tau}\right) Z_{D_{i,j,\nu_j}}^{(v,\ell)}(t_i)\right. \\
&\qquad \left. - \left(d_s + (L_i-1)\tau\right)\right) \\
&= \tfrac{1}{t_i}\log\left(\textstyle\sum_{j=1}^{m}\sum_{\nu_j=1}^{d_j} \pi_{i,j,\nu_j}^{(\ell)} \left(e^{t_i(d_s+(L_i-1)\tau)} + \sum_{v=1}^{L_i} e^{t_i(L_i-v)\tau} Z_{D_{i,j,\nu_j}}^{(v,\ell)}(t_i)\right)\right. \\
&\qquad \left. - \tfrac{1}{t_i}\log\left(e^{t_i(d_s+(L_i-1)\tau)}\right)\right. \\
&= \tfrac{1}{t_i}\log\left(\textstyle\sum_{j=1}^{m}\sum_{\nu_j=1}^{d_j} \pi_{i,j,\nu_j}^{(\ell)} \left(1 + \sum_{v=1}^{L_i} e^{-t_i(d_s+(v-1)\tau)} Z_{D_{i,j,\nu_j}}^{(v,\ell)}(t_i)\right)\right., \qquad (4.36)
\end{aligned}$$

where $Z_{D_{i,j,\nu_j}}^{(v,\ell)}(t_i) \triangleq \frac{\left(1-\rho_{j,\nu_j}\right)t_i B_{j,\nu_j}(t_i)}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{t_i \beta_{j,\nu_j}^{(\ell)}}}{\alpha_{j,\nu_j}^{(\ell)} - t_i}\right)^{v}$.

Let $H_{i,j,\nu_j,\ell} = \sum_{v=1}^{L_i} e^{-t_i(d_s+(v-1)\tau)} Z_{D_{i,j,\nu_j}}^{(v,\ell)}(t_i)$, which is the inner summation in (4.36). $H_{i,j,\nu_j,\ell}$ can be simplified using the geometric series formula to obtain

$$H_{i,j,\nu_j,\ell} = \sum_{v=1}^{L_i} \left(\frac{e^{-t_i(d_s+(v-1)\tau)} \left(1-\rho_{j,\nu_j}\right) t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{t_i \beta_{j,\nu_j}^{(\ell)}}}{\alpha_{j,\nu_j}^{(\ell)} - t_i} \right)^v \right) \tag{4.37}$$

$$= \frac{e^{-t_i d_s} \left(1-\rho_{j,\nu_j}\right) t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \times \sum_{v=1}^{L_i} \left(e^{-t_i(v-1)\tau} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{t_i \beta_{j,\nu_j}^{(\ell)}}}{\alpha_{j,\nu_j}^{(\ell)} - t_i} \right)^v \right) \tag{4.38}$$

$$= \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_{j,\nu_j}\right) t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \times \sum_{v=1}^{L_i} \left(e^{-v(t_i\tau)} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{t_i \beta_{j,\nu_j}^{(\ell)}}}{\alpha_{j,\nu_j}^{(\ell)} - t_i} \right)^v \right) \tag{4.39}$$

$$= \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_{j,\nu_j}\right) t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \times \sum_{v=1}^{L_i} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{t_i \beta_{j,\nu_j}^{(\ell)} - t_i\tau}}{\alpha_{j,\nu_j}^{(\ell)} - t_i} \right)^v \tag{4.40}$$

$$= \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_{j,\nu_j}\right) t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \times \left(M_{j,\nu_j}^{(\ell)}(t_{\mathrm{i}}) e^{-t_i\tau} \frac{1 - \left(M_{j,\nu_j}^{(\ell)}(t_i)\right)^{Li} e^{-t_i L_i \tau}}{1 - M_{j,\nu_j}^{(\ell)}(t_i) e^{-t_i\tau}} \right) \tag{4.41}$$

$$= \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_{j,\nu_j}\right) t_i}{t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i)-1\right)} \times \left[\frac{\widetilde{M}_{j,\nu_j}^{(\ell)}(t_i) \left(1 - \left(\widetilde{M}_{j,\nu_j}^{(\ell)}(t_i)\right)^{L_i}\right)}{1 - \widetilde{M}_{j,\nu_j}^{(\ell)}(t_i)} \right] \tag{4.42}$$

where

$$\widetilde{M}_{j,\nu_j}^{(\ell)}(t_i) = M_{j,\nu_j}^{(\ell)}(t_{\mathrm{i}}) e^{-t_i\tau}, \tag{4.43}$$

$M_{j,\nu_j}^{(\ell)}(t_i)$ is given in (4.5), and $B_{j,\nu_j}(t_i)$ is given in (4.12).

Theorem 4.5.1 *The mean stall duration time for file i streamed with quality ℓ is bounded by*

$$\mathbb{E}\left[\Gamma^{(i,\ell)}\right] \leq \frac{1}{t_i} log \left(\sum_{j=1}^{m} \sum_{\nu_j=1}^{d_j} \pi_{i,j,\nu_j}^{(\ell)} \left(1 + H_{i,j,\nu_j,\ell}\right) \right) \tag{4.44}$$

for any $t_i > 0$, $\rho_{j,\nu_j} = \sum_{i,\ell} \pi_{i,j,\nu_j}^{(\ell)} \lambda_i b_{i,\ell} L_i \left(\beta_{j,\nu_j}^{(\ell)} + \frac{1}{\alpha_{j,\nu_j}^{(\ell)}} \right)$, $\rho_{j,\nu_j} < 1$, and

$\sum_{f=1}^{r} \sum_{\ell=1}^{V} \pi_{f,j,\nu_j}^{(\ell)} \lambda_f b_{f,\ell} \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{-\beta_{j,\nu_j}^{(\ell)} t_i}}{\alpha_{j,\nu_j}^{(\ell)} - t_i} \right)^{L_f} - \left(\Lambda_{j,\nu_j} + t_i\right) < 0, \forall j, \nu_j.$

Note that Theorem above holds only in the range of t_i when $t_i - \Lambda_{j,\nu_j}\left(B_{j,\nu_j}(t_i) - 1\right) > 0$ which reduces to $\sum_{f=1}^{r}\sum_{\ell=1}^{V}\pi_{f,j,\nu_j}^{(\ell)}\lambda_f b_{f,\ell}\left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{-\beta_{j,\nu_j}^{(\ell)} t_i}}{\alpha_{j,\nu_j}^{(\ell)} - t_i}\right)^{L_f} - \left(\Lambda_{j,\nu_j} + t_i\right) < 0$, $\forall i, j, \nu_j$, and $\alpha_{j,\nu_j} - t_i > 0$. Further, the server utilization ρ_{j,ν_j} must be less than 1 for stability of the system.

4.6 Optimization Problem Formulation and Proposed Algorithm

We are now ready to formulate the joint stall-quality optimization problem and explain the proposed algorithm to efficiently solve it.

4.6.1 Problem Formulation

Let $\boldsymbol{q} = (q_{i,j}^{(\ell)}, \forall i = 1, \ldots, r, j = 1, \cdots, m, \ell = 1, \ldots, V)$, $\boldsymbol{b} = (b_{i,\ell}, \forall i = 1, \cdots, r, l = 1, \cdots, V)$, $\boldsymbol{w} = \left(w_{j,\nu_j}, \forall j = 1, \cdots, m, \nu_j = 1, \cdots, d_j\right)$, $\boldsymbol{p} = \left(p_{j,\nu_j}^{(\ell)}, \forall j = 1, \cdots, m,\ \nu_j = 1, \cdots d_j, \ell = 1, \cdots, V\right)$, and $\boldsymbol{t} = (t_1, t_2, \cdots, t_r)$. We wish to minimize the two proposed QoE metrics over the choice of two-stage probabilistic scheduling parameters, bandwidth allocation, probability of the quality of the streamed video and auxiliary variables. Since this is a multi-objective optimization, the objective can be modeled as a convex combination of the two QoE metrics.

Let $\overline{\lambda} = \sum_i \lambda_i$ be the total arrival rate of file i. Then, $\lambda_i/\overline{\lambda}$ is the ratio of video i requests. The first objective is the minimization of the mean stall duration, averaged over all the file requests, and is given as $\sum_{i,\ell}\frac{\lambda_i}{\overline{\lambda}}\mathbb{E}\left[\Gamma^{(i,\ell)}\right]$. The second objective is maximizing the streamed quality of all video requests, averaged over all the file requests, and is given as $\sum_{i,\ell}\frac{\lambda_i}{\overline{\lambda}}L_i b_{i,\ell} a_\ell$. Using the expressions for the mean stall duration in Section A.19 and the average streamed quality, optimization of a convex combination of the two QoE metrics can be formulated as follows.

$$\min \sum_{i=1}^{r} \frac{\lambda_i}{\overline{\lambda_i}} \left[\theta \left(\sum_{\ell=1}^{V} -b_{i,\ell} L_i a_\ell \right) + (1-\theta) \times \right.$$
$$\left. \sum_{\ell} \frac{b_{i,\ell}}{t_i} \log \left(\sum_{j=1}^{m} \sum_{\nu_j=1}^{d_j} q_{i,j}^{(\ell)} p_{j,\nu_j}^{(\ell)} (1 + H_{i,j,\nu_j,\ell}) \right) \right] \tag{4.45}$$

subject to: (4.46)

$$(4.42), (4.43), (4.5), (4.12), (4.13), (\text{A}.12), (4.6), \tag{4.47}$$

$$\rho_{j,\nu_j} < 1 \quad \forall j, \nu_j \tag{4.48}$$

$$\Lambda_{j,\nu_j} = \sum_{f=1}^{r} \sum_{\ell=1}^{V} \lambda_f b_{f,\ell} q_{i,j}^{(\ell)} p_{j,\nu_j}^{(\ell)} \quad \forall j, \nu_j \tag{4.49}$$

$$\sum_{j=1}^{m} q_{i,j}^{(\ell)} = k_i \,, \, \forall i, \, \ell \tag{4.50}$$

$$q_{i,j}^{(\ell)} = 0 \text{ if } j \notin S_i^{(\ell)} \,, q_{i,j}^{(\ell)} \in [0,1] \tag{4.51}$$

$$\sum_{\nu_j} p_{j,\nu_j}^{(\ell)} = 1, \; p_{j,\nu_j}^{(\ell)} \geq 0, \; \forall j, \nu_j, \ell, \tag{4.52}$$

$$\sum_{\ell} b_{i,\ell} = 1, \, b_{i,\ell} \geq 0, \; \forall i, \ell \tag{4.53}$$

$$0 \leq w_{j,\nu_j} \leq 1, \; \forall j, \nu_j \tag{4.54}$$

$$\sum_{\nu_j} w_{j,\nu_j} \leq 1, \; \forall j, \tag{4.55}$$

$$0 < t_i < \alpha_{j,\nu_j}^{(\ell)}, \, \forall i, j, \ell, \nu_j \tag{4.56}$$

$$\alpha_{j,\nu_j}^{(\ell)} \left(e^{(\beta_{j,\nu_j}^{(\ell)} - \tau) t_i} - 1 \right) + t_i < 0 \,, \forall i, j, \nu_j, \ell \tag{4.57}$$

$$\sum_{f=1}^{r} \sum_{\ell=1}^{V} q_{f,j}^{(\ell)} p_{j,\nu_j}^{(\ell)} b_{f,\ell} \lambda_f \left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{\beta_{j,\nu_j}^{(\ell)} t_i}}{\alpha_{j,\nu_j}^{(\ell)} - t_i} \right)^{L_f} -$$
$$\left(\Lambda_{j,\nu_j} + t_i \right) < 0, \, \forall i, j, \nu_j \tag{4.58}$$

$$\text{var.} \quad \boldsymbol{q}, \boldsymbol{t}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{p} \tag{4.59}$$

Here, $\theta \in [0,1]$ is a trade-off factor that determines the relative significance of the mean stall duration and the average streamed quality in the minimization problem. Varying $\theta = 0$ to $\theta = 1$, the solution for (4.45) spans the solutions that maximize

the video quality to those minimizing the mean stall duration. The equations (4.42), (4.43), (4.5), (4.12), (4.13), (A.12), and (4.6) give the terms in the objective function. The constraint (4.48) indicates that the load intensity of server j is less than 1. Equation (4.49) gives the aggregate arrival rate Λ_j for each node. Constraints (4.51), (4.51), and (4.52) guarantee that the two-stage scheduling probabilities are feasible. Constraint (4.53) guarantees that the quality assignment probabilities are feasible and (4.55) is for bandwidth splitting among different streams. Constraints (4.56), (4.57), and (4.58) ensure that $\widetilde{M}_j(t)$ and the moment generating function given in (4.23) exist. In the next subsection, we will describe the proposed algorithm for this optimization problem.

4.6.2 Proposed Algorithm

The mean stall duration optimization problem given in (4.45)-(4.59) is optimized over five set of variables: server scheduling probabilities $\boldsymbol{q}$, PS selection probabilities $\boldsymbol{p}$, auxiliary parameters $\boldsymbol{t}$, video quality parameters $\boldsymbol{b}$, and bandwidth allocation weights $\boldsymbol{w}$. We first note that the problem is non-convex in all the parameters jointly, which can be easily seen in the terms which are product of the different variables. Since the problem is non-convex, we propose an iterative algorithm to solve the problem. The proposed algorithm divides the problem into five sub-problems (i.e., convex problems and thus easy to solve with low complexity) that optimize one variable while fixing the remaining four. The five sub-problems are labeled as (i) Server Access Optimization: optimizes $\boldsymbol{q}$, for given $\boldsymbol{p}$, $\boldsymbol{t}$, $\boldsymbol{b}$ and $\boldsymbol{w}$, (ii) PS Selection Optimization: optimizes $\boldsymbol{p}$, for given $\boldsymbol{q}$, $\boldsymbol{t}$, $\boldsymbol{b}$ and $\boldsymbol{w}$, (iii) Auxiliary Variables Optimization: optimizes $\boldsymbol{t}$ for given $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{b}$ and $\boldsymbol{w}$, and (iv) Video Quality Optimization: optimizes $\boldsymbol{b}$ for given $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{t}$, and $\boldsymbol{w}$, and (v) Bandwidth Allocation Optimization: optimizes $\boldsymbol{w}$ for given $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{t}$, and $\boldsymbol{b}$. The algorithm is summarized as follows.

1. **Initialization**: Initialize $\boldsymbol{t}, \boldsymbol{b}$, $\boldsymbol{w}$, $\boldsymbol{p}$, and $\boldsymbol{q}$ in the feasible set.

2. **While Objective Converges**

(a) Run Server Access Optimization using current values of $\boldsymbol{p}$, $\boldsymbol{t}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ to get new values of $\boldsymbol{q}$

(b) Run PS Selection Optimization using current values of $\boldsymbol{q}$, $\boldsymbol{t}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ to get new values of $\boldsymbol{p}$

(c) Run Auxiliary Variables Optimization using current values of $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ to get new values of $\boldsymbol{t}$

(d) Run Streamed Quality Optimization using current values of $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{t}$, and $\boldsymbol{w}$ to get new values of $\boldsymbol{b}$.

(e) Run Bandwidth Allocation Optimization using current values of $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{t}$, and $\boldsymbol{b}$ to get new values of $\boldsymbol{w}$.

We next describe the five sub-problems along with the proposed solutions for the sub-problems.

Server Access Optimization

Given the probability distribution of the streamed video quality, the bandwidth allocation weights, the PS selection probabilities, and the auxiliary variables, this subproblem can be written as follows.

Input: $\boldsymbol{t}$, $\boldsymbol{b}$, $\boldsymbol{p}$, and $\boldsymbol{w}$

Objective: min (4.45)
s.t. (4.48), (4.49), (4.51), (4.51), (4.58)
var. $\boldsymbol{q}$

In order to solve this problem, we have used iNner cOnVex Approximation (NOVA) algorithm proposed in [57] to solve this sub-problem. The key idea for this algorithm is that the non-convex objective function is replaced by suitable convex approximations at which convergence to a stationary solution of the original non-convex optimization is established. NOVA solves the approximated function efficiently and maintains feasibility in each iteration. The objective function can be approximated by a convex

one (e.g., proximal gradient-like approximation) such that the first order properties are preserved [57], and this convex approximation can be used in NOVA algorithm.

Let $\widetilde{U_q}\left(\boldsymbol{q};\boldsymbol{q^\nu}\right)$ be the convex approximation at iterate $\boldsymbol{q^\nu}$ to the original non-convex problem $U\left(\boldsymbol{q}\right)$, where $U\left(\boldsymbol{q}\right)$ is given by (4.45). Then, a valid choice of $\widetilde{U_q}\left(\boldsymbol{q};\boldsymbol{q^\nu}\right)$ is the first order approximation of $U\left(\boldsymbol{q}\right)$, e.g., (proximal) gradient-like approximation, i.e.,

$$\widetilde{U_q}\left(\boldsymbol{q},\boldsymbol{q^\nu}\right)=\nabla_{\boldsymbol{q}}U\left(\boldsymbol{q^\nu}\right)^T\left(\boldsymbol{q}-\boldsymbol{q^\nu}\right)+\frac{\tau_u}{2}\left\|\boldsymbol{q}-\boldsymbol{q^\nu}\right\|^2, \tag{4.60}$$

where τ_u is a regularization parameter. Note that all the constraints (4.48), (4.49), (4.51), (4.51), and (4.58) are linear in $\boldsymbol{q_{i,j}}$. The NOVA Algorithm for optimizing $\boldsymbol{q}$ is described in Algorithm 3 (given in Appendix A.6). Using the convex approximation $\widetilde{U_q}\left(\boldsymbol{q};\boldsymbol{q^\nu}\right)$, the minimization steps in Algorithm 3 are convex, with linear constraints and thus can be solved using a projected gradient descent algorithm with low complexity and small timing overhead. A step-size (γ) is also used in the update of the iterate $\boldsymbol{q}^\nu$. Note that the iterates $\left\{\boldsymbol{q}^{(\nu)}\right\}$ generated by the algorithm are all feasible for the original problem and, further, convergence is guaranteed, as shown in [57] and described in lemma 21.

In order to use NOVA, there are some assumptions (given in [57]) that have to be satisfied in both original function and its approximation. These assumptions can be classified into two categories. The first category is the set of conditions that ensure that the original problem and its constraints are continuously differentiable on the domain of the function, which are satisfied in our problem. The second category is the set of conditions that ensures that the approximation of the original problem is uniformly strongly convex on the domain of the function. The latter set of conditions are also satisfied as the chosen function is strongly convex and its domain is also convex. To see this, we need to show that the constraints (4.48), (4.49), (4.51), (4.51), (4.58) form a convex set in $\mathbf{q}$ which is easy to see from the linearity of the constraints in $\mathbf{q}$. Further details on the assumptions and function approximation can be found in [57]. Therefore, the following result holds.

Lemma 6 *For fixed* $\boldsymbol{b}$, $\boldsymbol{p}$, $\boldsymbol{w}$, *and* $\boldsymbol{t}$, *the optimization of our problem over* $\boldsymbol{q}$ *generates a sequence of decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Auxiliary Variables Optimization

Given the probability distribution of the streamed video quality, the bandwidth allocation weights, the PS selection probabilitites and the server scheduling probabilities, this subproblem can be written as follows.

Input: $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{b}$, **and** $\boldsymbol{w}$

Objective: min (4.45)

s.t. (4.56), (4.57), (4.58)

var. $\boldsymbol{t}$

Similar to Access Optimization, this optimization can be solved using NOVA algorithm. The constraint (4.56) is linear in $\boldsymbol{t}$. Further, the next two Lemmas show that the constraints (4.57) and (4.58) are convex in $\boldsymbol{t}$, respectively.

Lemma 7 *The constraint* (4.57) *is convex with respect to* $\boldsymbol{t}$.

Proof The constraint (4.57) is separable for each t_i and thus it is enough to prove convexity of $C(t) = \alpha_{j,\nu_j}\left(e^{\left(\beta_{j,\nu_j}-\tau\right)t}-1\right)+t$. Thus, it is enough to prove that $C''(t) \geq 0$.

The first derivative of $C(t)$ is given as

$$C'(t) = \alpha_{j,\nu_j}\left(\left(\beta_{j,\nu_j}-\tau\right)e^{\left(\beta_{j,\nu_j}-\tau\right)t}\right)+1 \tag{4.61}$$

Differentiating it again, we get the second derivative as follows.

$$C''(t) = \alpha_{j,\nu_j}\left(\beta_{j,\nu_j}-\tau\right)^2 e^{\left(\beta_{j,\nu_j}-\tau\right)t} \tag{4.62}$$

Since $\alpha_{j,\nu_j} \geq 0$, $C''(t)$ given in (A.9) is non-negative, which proves the Lemma. ■

Lemma 8 *The constraint* (4.58) *is convex with respect to* $\boldsymbol{t}$.

Proof The constraint (4.58) is separable for each t_i, and thus it is enough to prove convexity of $E(t) = \sum_{f=1}^{r} \pi_{f,j,\nu_j} \lambda_f b_{f,\ell} a_\ell \left(\frac{\alpha_{j,\nu_j} e^{\beta_{j,\nu_j} t}}{\alpha_{j,\nu_j} - t} \right)^{L_f} - \left(\Lambda_{j,\nu_j} + t \right)$ for $t < \alpha_{j,\nu_j}$. Thus, it is enough to prove that $E''(t) \geq 0$ for $t < \alpha_{j,\nu_j}$. We further note that it is enough to prove that $D''(t) \geq 0$, where $D(t) = \frac{e^{L_f \beta_{j,\nu_j} t}}{(\alpha_{j,\nu_j} - t)^{L_f}}$. This follows since

$$\begin{aligned} D^{'}(t) &= \frac{L_f e^{L_f \beta_{j,\nu_j} t} \left[\beta_{j,\nu_j} + \left(\alpha_{j,\nu_j} - t \right)^{-1} \right]}{\left(\alpha_{j,\nu_j} - t \right)^{L_f}} \geq 0 \\ D^{''}(t) &= \frac{L_f \beta_{j,\nu_j} e^{L_f \beta_{j,\nu_j} t} \left[\beta_{j,\nu_j} + \frac{1+L_f}{\alpha_{j,\nu_j} - t} \left(1 + \frac{1/\beta_{j,\nu_j}}{\alpha_{j,\nu_j} - t} \right) \right]}{\left(\alpha_{j,\nu_j} - t \right)^{L_f + 2}} \geq 0 \end{aligned}$$

■

Algorithm 4 (given in Appendix A.6) shows the used procedure to solve for $\boldsymbol{t}$. Let $\overline{U}\left(\boldsymbol{t}; \boldsymbol{t^\nu}\right)$ be the convex approximation at iterate $\boldsymbol{t^\nu}$ to the original non-convex problem $U\left(\boldsymbol{t}\right)$, where $U\left(\boldsymbol{t}\right)$ is given by (4.45), assuming other parameters constant. Then, a valid choice of $\overline{U}\left(\boldsymbol{t}; \boldsymbol{t^\nu}\right)$ is the first order approximation of $U\left(\boldsymbol{t}\right)$, i.e.,

$$\overline{U}\left(\boldsymbol{t}, \boldsymbol{t^\nu}\right) = \nabla_{\boldsymbol{t}} U\left(\boldsymbol{t^\nu}\right)^T \left(\boldsymbol{t} - \boldsymbol{t^\nu}\right) + \frac{\tau_t}{2} \left\| \boldsymbol{t} - \boldsymbol{t^\nu} \right\|^2. \tag{4.63}$$

where τ_t is a regularization parameter. The detailed steps can be seen in Algorithm 4. Since all the constraints (4.56), (4.57), and (4.58) have been shown to be convex in $\boldsymbol{t}$, the optimization problem in Step 1 of Algorithm 4 can be solved by the standard projected gradient descent algorithm.

Lemma 9 *For fixed $\boldsymbol{q}$, $\boldsymbol{b}$, $\boldsymbol{w}$, and $\boldsymbol{p}$, the optimization of our problem over $\boldsymbol{t}$ generates a sequence of monotonically decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Streamed Video Quality Optimization

Given the auxiliary variables, the bandwidth allocation weights, the PS selection probabilities, and the scheduling probabilities, this subproblem can be written as follows.

Input: $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{t}$, **and** $\boldsymbol{w}$

Objective: min (4.45)

s.t. (4.48), (4.49), (4.53), (4.58)

var. $\boldsymbol{b}$

Similar to the aforementioned two Optimization problems, this optimization can be solved using NOVA algorithm. The constraints (4.48), (4.49), (4.53), and (4.58) are linear in $\boldsymbol{b}$, and hence, form a convex domain.

Algorithm 5 (given in Appendix A.6) shows the used procedure to solve for $\boldsymbol{b}$. Let $U_b\left(\boldsymbol{b};\boldsymbol{b^\nu}\right)$ be the convex approximation at iterate $\boldsymbol{b^\nu}$ to the original non-convex problem $U\left(\boldsymbol{b}\right)$, where $U\left(\boldsymbol{b}\right)$ is given by (4.45), assuming other parameters constant. Then, a valid choice of $U_b\left(\boldsymbol{b};\boldsymbol{b^\nu}\right)$ is the first order approximation of $U\left(\boldsymbol{b}\right)$, i.e.,

$$U_b\left(\boldsymbol{b},\boldsymbol{b^\nu}\right) = \nabla_{\boldsymbol{b}} U\left(\boldsymbol{b^\nu}\right)^T\left(\boldsymbol{b}-\boldsymbol{b^\nu}\right) + \frac{\tau_b}{2}\left\|\boldsymbol{b}-\boldsymbol{b^\nu}\right\|^2. \tag{4.64}$$

where τ_t is a regularization parameter. The detailed steps can be seen in Algorithm 5. Since all the constraints have been shown to be convex in $\boldsymbol{b}$, the optimization problem in Step 1 of Algorithm 5 can be solved by the standard projected gradient descent algorithm.

Lemma 10 *For fixed* $\boldsymbol{t}$, $\boldsymbol{w}$, $\boldsymbol{p}$, *and* $\boldsymbol{q}$, *the optimization of our problem over* $\boldsymbol{b}$ *generates a sequence of monotonically decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Bandwidth Allocation Weights Optimization

Given the auxiliary variables, the streamed video quality probabilities, the PS selection probabilities, and the scheduling probabilities, this subproblem can be written as follows.

Input: $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{t}$, **and** $\boldsymbol{b}$

Objective: min (4.45)

s.t. (4.48), (4.54), (4.55), (4.58)

$$\text{var. } \boldsymbol{w}$$

This optimization problem can be solved using NOVA algorithm. It is easy to notice that the constraints (4.54) and (4.55) are linear and thus convex with respect to $\boldsymbol{b}$. Further, the next two Lemmas show that the constraints (4.48) and (4.58) are convex in $\boldsymbol{w}$, respectively.

Lemma 11 *The constraint* (4.48) *is convex with respect to* $\boldsymbol{w}$.

Proof Since there is no coupling between the subscripts j, ℓ, and ν_j in (4.48), we remove the subscripts in the rest of the proof. Moreover, since α is linear in w, it is enough to prove the convexity with respect to α. Also, the constraint (4.48) is separable for each α and thus it is enough to prove convexity of $C_1(\alpha) = 1/\alpha$. It is easy to show that the second derivative of $C_1(\alpha)$ with respect to α is given by

$$C_1^{''}(\alpha) = \frac{2}{\alpha^3} \tag{4.65}$$

Since $\alpha \geq 0$, $C_1^{''}(\alpha)$ given in (4.65) is non-negative, which proves the Lemma. ■

Lemma 12 *The constraint* (4.58) *is convex with respect to* $\boldsymbol{w}$.

Proof The constraint (4.58) is separable for each α_{j,ν_j}^{ℓ}, and thus it is enough to prove convexity of
$E_1(\alpha_{j,\nu_j}^{(\ell)}) = \sum_{f=1}^{r}\sum_{\ell=1}^{V}\pi_{f,j,\nu_j}^{(\ell)}\lambda_f b_{f,\ell}\left(\frac{\alpha_{j,\nu_j}^{(\ell)} e^{\beta_{j,\nu_j}^{(\ell)} t}}{\alpha_{j,\nu_j}^{(\ell)} - t}\right)^{L_f} - \left(\Lambda_{j,\nu_j} + t\right)$ for $t < \alpha_{j,\nu_j}^{(\ell)}$. Since there is only a single index j, ν_j, and ℓ here, we ignore the subscripts and superscripts for the rest of this proof. Thus, it is enough to prove that $E_1^{''}(\alpha) \geq 0$ for $t < \alpha$. We further note that it is enough to prove that $D_1^{''}(\alpha) \geq 0$, where $D_1(\alpha) = \left(1 - \frac{t}{\alpha}\right)^{-L_i}$. This holds since,

$$D_1^{'}(\alpha) = \frac{-L_i \times t}{\alpha^2}\left(\frac{\alpha}{\alpha - t}\right)^{L_i+1} \tag{4.66}$$

$$D_1^{''}(\alpha) = \frac{L_i \times t}{\alpha^3}\left(\frac{\alpha}{\alpha - t}\right)^{L_i+1}\left[2 + \frac{\alpha\left(L_i + 1\right)}{\alpha_j - t}\right] \geq 0 \tag{4.67}$$

■

Algorithm 5 (given in Appendix A.17) shows the used procedure to solve for $\boldsymbol{w}$. Let $U_w(\boldsymbol{w};\boldsymbol{w}^{\boldsymbol{\nu}})$ be the convex approximation at iterate $\boldsymbol{w}^{\boldsymbol{\nu}}$ to the original non-convex problem $U(\boldsymbol{w})$, where $U(\boldsymbol{w})$ is given by (4.45), assuming other parameters constant. Then, a valid choice of $U_w(\boldsymbol{w};\boldsymbol{w}^{\boldsymbol{\nu}})$ is the first order approximation of $U(\boldsymbol{w})$, i.e.,

$$U_w(\boldsymbol{w},\boldsymbol{w}^{\boldsymbol{\nu}}) = \nabla_{\boldsymbol{w}} U(\boldsymbol{w}^{\boldsymbol{\nu}})^T(\boldsymbol{w}-\boldsymbol{w}^{\boldsymbol{\nu}}) + \frac{\tau_w}{2}\|\boldsymbol{w}-\boldsymbol{w}^{\boldsymbol{\nu}}\|^2. \tag{4.68}$$

where τ_t is a regularization parameter. The detailed steps can be seen in Algorithm 5. Since all the constraints have been shown to be convex, the optimization problem in Step 1 of Algorithm 5 can be solved by the standard projected gradient descent algorithm.

Lemma 13 *For fixed $\boldsymbol{q}$, $\boldsymbol{p}$, $\boldsymbol{t}$, and $\boldsymbol{b}$, the optimization of our problem over $\boldsymbol{w}$ generates a sequence of decreasing objective values and therefore is guaranteed to converge to a stationary point.*

PS Selection Probabilities

Given the auxiliary variables, the bandwidth allocation weights, the streamed video quality probabilities, and the scheduling probabilities, this subproblem can be written as follows.

Input: $\boldsymbol{q}$, $\boldsymbol{b}$, $\boldsymbol{t}$, and $\boldsymbol{w}$

Objective: min (4.45)
s.t. (4.48), (4.49), (4.52), (4.58),
var. $\boldsymbol{p}$

This optimization can be solved using NOVA algorithm. The constraints (4.48), (4.49), (4.52), and (4.58) are linear in $\boldsymbol{p}$, and hence, the domain is convex.

Algorithm 6 (given in Appendix A.17) shows the used procedure to solve for $\boldsymbol{p}$. Let $U_p(\boldsymbol{p};\boldsymbol{p}^{\boldsymbol{\nu}})$ be the convex approximation at iterate $\boldsymbol{p}^{\boldsymbol{\nu}}$ to the original non-convex

problem $U(\boldsymbol{p})$, where $U(\boldsymbol{p})$ is given by (4.45), assuming other parameters constant. Then, a valid choice of $U_p(\boldsymbol{p};\boldsymbol{p^\nu})$ is the first order approximation of $U(\boldsymbol{p})$, i.e.,

$$U_p(\boldsymbol{p},\boldsymbol{p^\nu}) = \nabla_{\boldsymbol{p}} U(\boldsymbol{p^\nu})^T(\boldsymbol{p}-\boldsymbol{p^\nu}) + \frac{\tau_p}{2}\|\boldsymbol{p}-\boldsymbol{b^\nu}\|^2. \tag{4.69}$$

where τ_p is a regularization parameter. The detailed steps can be seen in Algorithm 6. Since all the constraints have been shown to be convex in $\boldsymbol{p}$, the optimization problem in Step 1 of Algorithm 6 can be solved by the standard projected gradient descent algorithm.

Lemma 14 *For fixed $\boldsymbol{t}$, $\boldsymbol{w}$, $\boldsymbol{b}$, and $\boldsymbol{q}$, the optimization of our problem over $\boldsymbol{p}$ generates a sequence of monotonically decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Proposed Algorithm Convergence

We first initialize $q_{i,j}^{(\ell)}$, $p_{j,\nu_j}^{(\ell)}$, w_{j,ν_j}, t_i and $b_{i,\ell}$, $\forall\, i,j,\nu_j,\ell$ such that the choice is feasible for the problem. Then, we do alternating minimization over the five sub-problems defined above. Since each sub-problem converges (decreasing) and the overall problem is bounded from below, we have the following result.

Theorem 4.6.1 *The proposed algorithm converges to a stationary solution.*

4.6.3 Online Algorithm

While our proposed algorithm is optimized for an offline scenario, an online version of this algorithm can be derived according to the stationary scheduling probabilities and optimized probabilities. The arrival rates λ_i can be estimated based on a window based method. In this setting, a window size of ΔW is chosen, and the decisions in a window are based on the estimated arrival rates from the preceding window. Using these estimated arrival rates, the solution for the optimization problem in (4.45) gives

the optimal offline parameters (i.e., two-stage scheduling probabilities and quality decisions). According to these stationary scheduling probabilities, a randomized online policy can be obtained.

4.6.4 Time Complexity of the Offline Algorithm

In this section, we explain the time complexity of both online and offline algorithms. In the offline version, we solve in an alternative manner five sub-problems. Each problem is a approximated by a convex one. We use NOVA [57] to solve every sub-problem, which has been shown to converge in a few iterations. In addition, the objective function is separable with respect to the index i (i.e., video file index) and thus every request for video file i can be solved in parallel with other requests. Hence, solving the objective function can be parallelized which significantly results in reducing the overall complexity.

It can be seen that while providing an approximate solution on the original problem, our algorithm needs only a few iterations to converge to a stationary point which translates into a good scalability (see Figure 5.7 for further details). We note that an overhead is incurred every time we solve the offline problem at the central controller. However, this optimization can be performed in the off-peak hours. The time and optimization overhead depend on the service provider capabilities and how much overhead the system can handle. For the online algorithm, the optimization is performed offline and based on the optimized parameters, a stationary online policy is obtained. Thus, there is $O(1)$ complexity for the online problem, given the solution to the offline problem.

4.7 Numerical Results

In this section, we evaluate our proposed algorithms for joint optimization of the mean stall duration and the average streamed video quality.

4.7.1 Parameter Setup

We simulate our algorithm in a distributed storage system of $m = 12$ distributed nodes, where each video file uses an $(7, 4)$ erasure code. However, our model can be used for any given number of storage servers and for any erasure coding setting. We assume $d_j = 20$ (unless otherwise explicitly stated) and $r = 1000$ files, whose sizes are generated based on Pareto distribution [62] (as it is a commonly used distribution for file sizes [63]) with shape factor of 2 and scale of 300, respectively. Since we assume that the video file sizes are not heavy-tailed, the first 1000 file-sizes that are less than 120 minutes are chosen. We also assume that the chunk service time follows a shifted-exponential distribution with rate $\alpha_j^{(\ell)}$ and shift $\beta_j^{(\ell)}$, given as (4.6). The value of $\beta_j a_1$ is chosen to be 10 ms, while the value of α_j/a_1 is chosen as in Table 5.1 (the parameters of α_j/a_1 were chosen using a distribution, and kept fixed for the experiments). Unless explicitly stated, the arrival rate for the first 500 files is $0.002s^{-1}$ while for the next 500 files is set to be $0.003s^{-1}$. Chunk size τ is set to be equal to 4 seconds (s). When generating video files, the size of each video file is rounded up to the multiple of 4 seconds. The values of a_ℓ for the 4 second chunk are given in Table 4.2, where the numbers have been taken from the dataset in [68]. We use a random placement of each file on 7 out of the 12 servers. In order to initialize our algorithm, we assume uniform scheduling, $q_{i,j}^{(\ell)} = k/n$ on the placed servers and $p_{j,\nu_j}^{(\ell)} = 1/d_j$. Further, we choose $t_i = 0.01$, $b_{i,\ell} = 1/V$, and $w_{j,\nu_j} = 1/d_j$. However, these choices of the initial parameters may not be feasible. Thus, we modify the parameter initialization to be closest norm feasible solutions.

Table 4.1.: The value of α_j/a_1 used in the Numerical Results, where the units are 1/s.

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
18.238	24.062	11.950	17.053	26.191	23.906

Node 7	Node 8	Node 9	Node 10	Node 11	Node 12
27.006	21.381	9.910	24.959	26.529	23.807

Table 4.2.: Data Size (per Mb) of the different quality levels.

ℓ	1	2	3	4	5	6
a_ℓ	6	11	19.2	31.2	41	56.2

4.7.2 Baselines

We compare our proposed approach with six strategies, which are described as follows.

1. *Projected Equal Access, Optimized Quality Probabilities, Auxiliary variables and Bandwidth Wights (PEA-QTB):* Starting with the initial solution mentioned above, the problem in (4.45) is optimized over the choice of $\boldsymbol{t}$, $\boldsymbol{b}$, $\boldsymbol{w}$, and $\boldsymbol{p}$ using alternating minimization. Thus, the value of $q_{i,j}^{(\ell)}$ will be approximately close to k/n for the servers on which the content is placed, indicating equal access of the k-out-of-n servers.

2. *Projected Equal Bandwidth, Optimized Quality Probabilities, Auxiliary variables and Server Access (PEB-QTA):* Starting with the initial solution mentioned above, the problem in (4.45) is optimized over the choice of $\boldsymbol{q}$, $\boldsymbol{t}$, $\boldsymbol{b}$, and $\boldsymbol{p}$ (using Algorithms 3, 4, 5, and 6, respectively) using alternating minimization. Thus, the bandwidth split w_{j,ν_j} will be approximately $1/d_j$.

3. *Projected Equal Quality, Optimized Bandwidth Wights, Auxiliary variables and Server Access (PEQ-BTA):* Starting with the initial solution mentioned above, the problem in (4.45) is optimized over the choice of $\boldsymbol{q}$, $\boldsymbol{t}$, $\boldsymbol{w}$, and $\boldsymbol{p}$ using alternating minimization. Thus, the quality assignment, $b_{i,\ell}$ will be approximately $1/V$.

4. *Projected Proportional Service-Rate, Optimized Quality, Auxiliary variables and Bandwidth Wights (PSP-QTB):* In the initialization, the access probabilities among the servers on which file i is placed, is given as $q_{i,j}^{(\ell)} = k_i \frac{\mu_j^{(\ell)}}{\sum_j \mu_j^{(\ell)}}$, $\forall i, j, \ell$.

This policy assigns servers proportional to their service rates. The choice of all parameters are then modified to the closest norm feasible solution. Using this initialization, the problem in (4.45) is optimized over the choice of $\boldsymbol{t}$, $\boldsymbol{b}$, $\boldsymbol{w}$, and $\boldsymbol{p}$ using alternating minimization.

5. *Projected Lowest Quality, Optimized Bandwidth Wights, Auxiliary variables and Server Access (PLQ-BTA):* In this strategy, we set $b_{i,1} = 1$ and $b_{i,\ell} = 0$, $\forall \ell \neq 1$ in the initialization thus choosing the lowest quality for all videos. Then, this choice is projected to the closest norm feasible solution. Using this initialization, the problem in (4.45) is optimized over the choice of $\boldsymbol{q}$, $\boldsymbol{t}$, $\boldsymbol{w}$, and $\boldsymbol{p}$ using alternating minimization.

6. *Projected Highest Quality, Optimized Bandwidth Wights, Auxiliary variables and Server Access (PHQ-BTA):* In this strategy, we set $b_{i,6} = 1$ and $b_{i,\ell} = 0$, $\forall \ell \neq 6$ in the initialization thus choosing the highest quality for all videos. Then, this choice is projected to the closest norm feasible solution. Using this initialization, the problem in (4.45) is optimized over the choice of $\boldsymbol{q}$, $\boldsymbol{t}$, $\boldsymbol{w}$, and $\boldsymbol{p}$ (using Algorithms 3, 4, 5, and 6, respectively) using alternating minimization.

Regarding the online mode, we compare our algorithm with three policies as described below:

1. *Join Shortest Queue (JSQ) Policy [46]:* In this policy, the video requests are assigned to the servers/PSs that have the lowest queue(s). All other parameters are optimized in the same manner as of our proposed policy. For detailed treatment of this policy, interested reader can refer to [46].

2. *Least Load-d LL(d) Policy [47]:* This policy works akin to a water-filling approach, where a set of d servers are chosen uniformly at random and then requests are assigned to the $k < d$ servers that have the lowest (remaining) loads (or processing times) among those selected servers. Interested reader can refer to [47] for further details. Note that also all other parameters are optimized in the same manner as of our proposed policy.

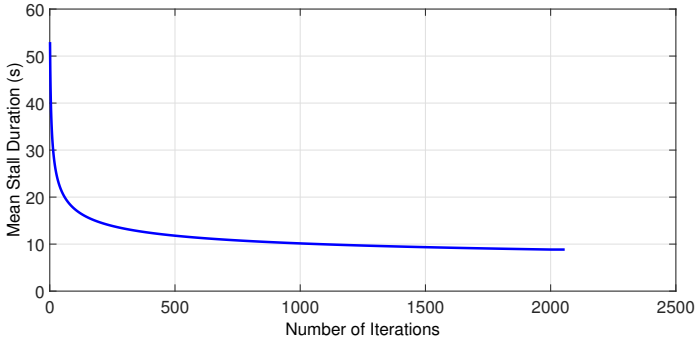

Fig. 4.3.: Convergence of mean stall duration.

3. *Power-of-d Pow(d) Policy [48]:* In this policy, a set of d servers are chosen uniformly at random and then video file requests are assigned to the server (or servers) that has the lowest queue/queues among those selected servers. In-detail description for this strategy can be found in [48]. Similar to the previous two policies, the other four parameters are optimized as of our proposed policy.

4.7.3 Results

In this subsection, we set $\theta = 10^{-4}$, *i.e.*, prioritizing stall minimization over quality enhancement. We note that the average quality numbers are orders of magnitude higher (since the quality term in (4.45) is proportional to the video length) than the mean stall duration and thus to bring the two to a comparable scale, the choice of $\theta = 10^{-4}$ is small. This choice of θ is motivated since users prefer not seeing interruptions more than seeing better quality. In this section, we will consider the average quality definition as Average Quality $= \sum_{i,\ell} \frac{\lambda_i}{\bar{\lambda}} \frac{L_i}{\sum_{k=1}^{r} L_k} b_{i,\ell} a_\ell$. We note that the maximum average quality is bounded by $a_6 = 56.2$. The division by the sum of lengths is used as a normalization so that the numbers in the figures can be interpreted better.

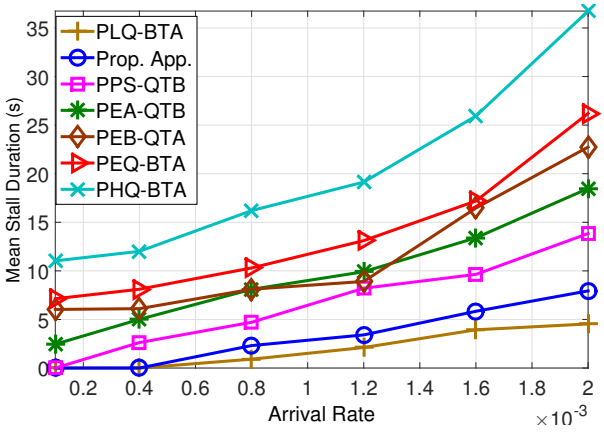

Fig. 4.4.: Mean stall duration for different video arrival rates.

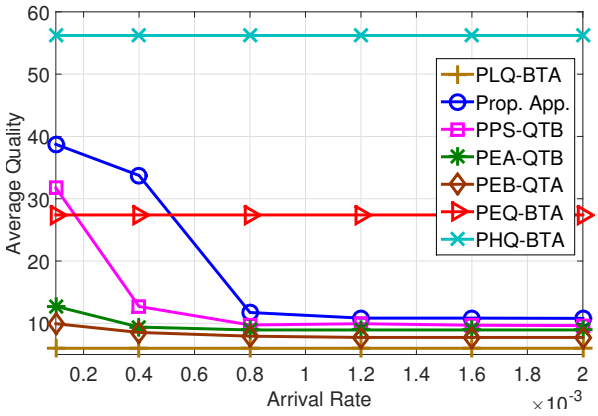

Fig. 4.5.: Average quality for different video arrival rates.

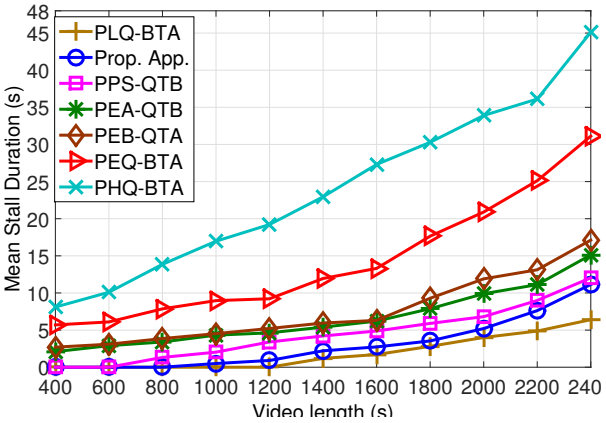

Fig. 4.6.: Mean stall duration for different video lengths.

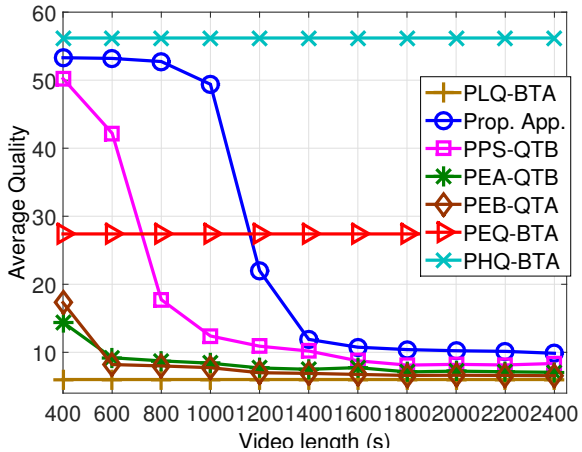

Fig. 4.7.: Average quality for different video lengths.

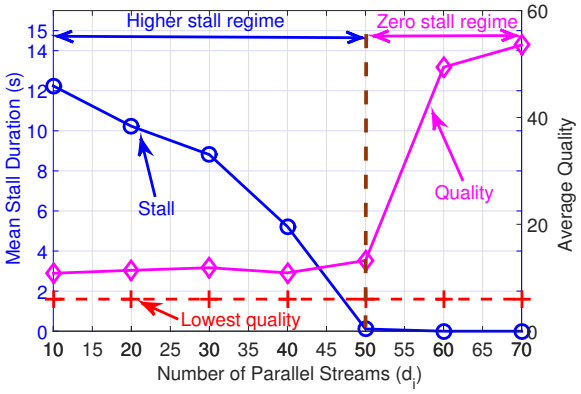

Fig. 4.8.: Average video quality and mean stall duration for different number of parallel streams d_j.

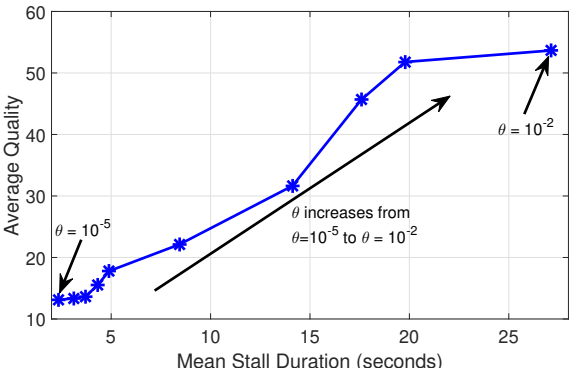

Fig. 4.9.: Tradeoff between mean stall duration and average streamed video quality obtained by varying θ.

Convergence of the Proposed Algorithm

Figure 4.3 shows the convergence of our proposed algorithm, where we see the convergence of mean stall duration in about 2000 iterations.

Effect of Arrival Rate

We assume the arrival rate of all the files the same, and vary the arrival rates as depicted in Figures 4.4 and 4.5. These figures show the effect of different video arrival rates on the mean stall duration and averaged quality, respectively. We note that PLQ-BTA achieves lowest stalls and lowest quality, since it fetches all videos at the lowest qualities. Similarly, PHQ-BTA has highest stalls, and highest video quality since it fetches all videos in the highest possible rate. The proposed algorithm has mean stall duration less than all the algorithms other than PLQ-BTA, and is very close to PLQ-BTA. Further, the proposed algorithm has the highest video quality among all algorithms except PHQ-BTA and PEQ-BTA. Thus, the proposed algorithm helps optimize both the QoEs simultaneously achieving close to the best possible stall durations and achieving better average video quality than the baselines. With the choice of low θ, the stall duration can be made very close to the stall duration achieved with the lowest quality while the proposed algorithm will still opportunistically increase quality of certain videos to obtain better average quality.

Effect of Video Length

The effect of having different video lengths on the mean stall duration and average quality is also captured in Figures 4.6 and 4.7, respectively, where we assume that all the videos are of the same length. Apparently, the mean stall duration increases with the video length while the average quality decreases with the video length. The qualitative comparison of the different algorithms is the same as described in the case of varying arrival rates. Thus, at $\theta = 10^{-4}$, the proposed algorithm achieves

the mean stall duration close to that of PLQ-BTA while achieving significantly better quality. For algorithms other than PLQ-BTA, PEQ-BTA, and PHQ-BTA, the proposed algorithms outperforms all other baselines in both the metrics.

Effect of the Number of the Parallel Streams (d_j)

Figure 4.8 plots the average streamed video quality and mean stall duration for varying number of parallel streams, d_j, for our proposed algorithm. We vary the number of PSs from 10 to 70 with increment step of 10 with $\theta = 10^{-4}$. Increasing d_j can only improve performance since some of the bandwidth splits can be zero thus giving the lower d_j solution as one of the possible feasible solution. Increasing d_j thus decreases stall durations by having more parallel streams, while increasing average quality. We note that for $d_j < 50$, mean stall duration is non-zero and the stall duration decreases significantly while the average quality increases only slightly. For $d_j > 50$, the stall duration remains zero and the average video quality increases significantly with increase in d_j. Even though larger d_j gives better results, the server may only be able to handle a limited parallel connections thus limiting the value of d_j in the real systems due to physical limitations.

Tradeoff between mean stall duration and average video quality

The preceding results show a trade off between the mean stall duration and the average quality of the streamed video. In order to investigate such tradeoff, Figure 4.9 plots the average video quality versus the mean stall duration for different values of θ ranging from $\theta = 10^{-5}$ to $\theta = 10^{-2}$. This figure implies that a compromise between the two QoE metrics can be achieved by our proposed streaming algorithm by setting θ to an appropriate value. As expected, increasing θ will increase the mean stall duration as there is more priority to maximizing the average video quality. Thus, an efficient tradeoff point between the QoE metrics can be chosen based on the service quality level desired by the service provider.

Appendix A.9 further provides evaluation results for varying the number of servers, encoding parameters, the time complexity of the algorithm, and the performance of the online algorithm.

4.8 Chapter Conclusion

In this chapters, a video streaming over cloud is considered where the content is erasure-coded on the distributed servers. We consider two quality of experience metrics to optimize: mean stall duration and average quality of the streamed video. A two-stage probabilistic scheduling is proposed for the choice of servers and the parallel streams between the server and the edge router. Using the two-stage probabilistic scheduling and probabilistic quality assignment for the videos, an upper bound on the mean stall duration is derived. An optimization problem that minimizes a convex combination of the two QoE metrics is formulated, over the choice of two-stage probabilistic scheduling, probabilistic quality assignment, bandwidth allocation, and auxiliary variables. Efficient algorithm is proposed to solve the optimization problem and the evaluation results depict the improved performance of the algorithm as compared to the considered baselines.

5. MULTI-TIER CACHING ANALYSIS IN CDN-BASED OTT VIDEO STREAMING SYSTEMS

5.1 Introduction

Over-the-top video streaming, e.g., Netflix and YouTube, has been dominating the global IP traffic in recent years. The traffic will continue to grow due to the introduction of even higher resolution video formats such as 4K on the horizon. As end-users consume video in massive amounts and in an increasing number of ways, service providers need flexible solutions in place to ensure that they can deliver content quickly and easily regardless of their customer's device or location. More than 50% of over-the-top video traffic are now delivered through content distribution networks (CDNs) [69]. Even though multiple solutions have been proposed for improving congestion in the CDN system, managing the ever-increasing traffic requires a fundamental understanding of the system and the different design flexibilities (control knobs) to make the best use of the limited hardware resources. This is the focus of this work.

The service providers typically use two-tier caching approach to improve the quality of streaming services [70–72]. In addition to the distributed cache servers provided by the CDN, the edge router can also have a cache so that some videos could be stored in this cache and gets the advantage of the proximity to end-users. However, there are many edge routers which imply that the hot content could be stored at multiple edge routers. There is an additional cache at the distributed cache servers (in CDN) from which data can be obtained if not already at the edge router. Such multi-tier caching is related to fog computing where the caching could be distributed at multiple locations in the network [71]. We also assume that the edge cache can help provide advantages similar to multicasting. If another user on the edge router is already con-

suming the file, the part of the video already downloaded is sent directly to the new user and the later part is sent to multiple users who requested the content on the same edge router. This work aims to analyze two-tiered caching in video streaming systems.

In this work, we consider an architecture of streaming system with a Virtualized Content Distribution Network (vCDN) [73,74]. The main role of this CDN infrastructure is not only to provide users with lower response time and higher bandwidth, but also to distribute the load (especially during peak time) across many edge locations. The infrastructure consists of a remote datacenter that stores complete original video data and multiple CDN sites (i.e., distributed cache servers) that only have part of those data and are equipped with solid state drives (SSDs) for high throughput. In addition, we assume that a second caching tier is located at the edge routers. A user request for video content not served from the edge cache is directed to a distributed cache. If it is still not completely served, the remaining part of the request is directed to the remote datacenter (as shown in Fig. 5.1). Multiple parallel connections are established between the distributed cache server and the edge router, as well as between the distributed cache servers and the origin server, to support multiple video streams simultaneously. Our goal is to develop an optimization framework and QoE metrics that service providers (or infrastructure) could use to answer the following questions: How to quantify the impact of multi-tier video caching on end user experience? What is the best video multi-tier caching strategy for CDN? How to optimize QoE metrics over various "control knobs"? Are there enough benefits to justify the adoption of proposed solutions in practice?

It has been shown that, in modern cloud applications such as Facebook, Bing, and Amazon's retail platform, the long tail of latency is of a major concern, with 99.9th percentile response times that are, orders of magnitude worse than the mean [11,56]. Thus, this work considers a QoE metric, called the stall duration tail probability (SDTP), which measures the likelihood of end users suffering a worse-than-expected stall duration, and develop a holistic optimization framework for minimizing the over-

all SDTP over joint caching content placement, network resource optimization and user request scheduling. SDTP, denoted by $\Pr(\Gamma^{(i)} > \sigma)$, measures the probability that the stall duration $\Gamma^{(i)}$ of video i is greater than a pre-defined threshold σ. Despite resource and load-balancing mechanisms, large scale storage systems evaluations show that there is a high degree of randomness in delay performance [75]. In contrast to web object caching and delivery, the video chunks in the latter part of a video do not have to be downloaded much earlier than their actual play time to maintain the desired QoE, making SDTP highly dependent on the joint optimization with resource management and request scheduling in CDN-based video streaming.

Quantifying SDTP with multi-tier cache/storage is an open problem. Even for single-chunk video files, the problem is equivalent to minimizing the download tail latency, which is still an open problem [26]. The key challenge arises from the difficulty of constructing and analyzing a scheduling policy that (optimally) redirects each request based on *dependent system and queueing dynamics* (including cache content, network conditions, request queue status) on the fly. To overcome these challenges, we propose a novel two-stage, probabilistic scheduling approach, where each request of video i is (i) processed by cache server j with probability $\pi_{i,j}$ and (ii) assigned to video stream v with probability $p_{i,j,v}$. The two-stage, probability scheduling allows us to model each cache server and video stream as separate queues, and thus, to characterize the distributions of different video chunks' download time and playback time. Further, the edge caching policy plays a key role in the system design. This work proposes an adaption of least-recently-used (LRU) caching mechanism [12,76], where each file is removed from the edge cache if it has not been requested again for a time that depends on the edge router and the file index. By optimizing these probabilities and the edge-cache parameters, we quantify SDTP through a closed-form, tight upper bound for CDN-based video streaming with arbitrary cache content placement and network resource allocation. We note that the analysis in this work is fundamentally different from those for distributed file storage, e.g., [22,23], because the stall duration of a video relies on the download times of all its chunks, rather than simply the time

to download the last chunk of a file. Further, since video chunks are downloaded and played sequentially, the download times and playback times of different video chunks are highly correlated and thus jointly determine the SDTP metric.

This work proposes a holistic optimization framework for minimizing overall SDTP in CDN-based video streaming. To the best of our knowledge, this is the first framework for multi-tier caching to jointly consider all key design degrees of freedom, including bandwidth allocation among different parallel streams, multi-tier cache content placement and update, request scheduling, and the modeling variables associated with the SDTP bound. An efficient algorithm is then proposed to solve this non-convex optimization problem. In particular, the proposed algorithm performs an alternating optimization over the different dimensions, such that each sub-problem is shown to have convex constraints and thus can be efficiently solved using the iNner cOnVex Approximation (NOVA) algorithm proposed in [57]. The proposed algorithm is implemented in a virtualized cloud system managed by Openstack [77]. The experimental results demonstrate significant improvement of QoE metrics as compared to the considered baselines.

The main contributions of this chapter can be summarized as follows:

- We propose a novel framework for analyzing CDN-based over-the top video streaming systems with the use of multiple caching tiers and multiple parallel streams between nodes. A novel two-stage probabilistic scheduling policy is proposed to assign each user request to different cache servers and parallel video streams. Further, the edge router uses an adaptation of LRU, and the distributed cache servers cache partial files.

- The distribution of (random) download time of different video chunks are analyzed. Then, using ordered statistics, we quantify the playback time of each video segment.

- Multiple recursive relations are set up to compute the stall duration tail probability. We first relate the compute of download time of each chunk to the play

time of each chunk, Since the play time depends not only on download of the current chunk, but also on the previous chunks. Second, stall duration must account for whether the file has been requested by anyone within a window time of a certain size to get advantage of edge cache. If it had been requested, the stall duration is a function of the time of the last request and the stall duration at that time. In the steady state analysis, this will lead to a recursion. This analysis has been used to derive an analytical upper bound on SDTP for arbitrary distributed cache content placement, parameters of edge cache, and the parameters of the two-stage probabilistic scheduling (Appendix A.13).

- A holistic optimization framework is developed to optimize a weighted sum of SDTP of all video files over the request scheduling probabilities, distributed cache content placement, the bandwidth allocation among different streams, edge cache parameters, and the modeling parameters in SDTP bound. An efficient algorithm is provided to decouple and solve this non-convex optimization (Section 5.5).

- To better understand the SDTP and how it relates to the QoE of users, we correlate this metric to a well-known QoE metric (called mean stall duration). Since the optimal point for the mean stall duration is not the same as that of the SDTP, we optimize a convex combination of the two metrics and show how the two QoE metrics can be compromised based on the point on the curve that is appropriate for the clients (Appendix A.19).

- The algorithm is implemented on a virtualized cloud system managed by Openstack. The simulation and trace-based results validate our theoretical analysis with the implementation and analytical results being close, thus demonstrating the efficacy of our proposed algorithm. The QoE metric is shown to have significant improvement as compared to competitive strategies (Appendix 5.6).

5.2 Chapter Organization

The rest of this chapter is organized as follows. In Section 5.3, we describe the system model used in the work with a description of CDN-based Over-the-top video streaming systems. Section 5.4 provides an upper bound on the mean stall duration. Section 5.5 formulates the QoE optimization problem as a weighted sum of all SDTP of all files and proposes the iterative algorithmic solution of this problem. Experimental results are provided in Section 5.6. Section 5.7 concludes this chapter.

5.3 System Model

5.3.1 System Description

We consider a content delivery network as shown in Fig. 5.1, consisting of a single datacenter that has an origin server, m geographically-distributed cache servers denoted by $j = 1, \ldots, m$, and edge-cache storage nodes associated with the edge routers $\ell \in \{1, 2, \cdots, R\}$, where R is the total number of edge-routers, as depicted in Figure 5.2. The compute cache servers (also called storage nodes) are located close to the edge of the network and thus provide lower access latency for end users. We also assume that each cache server is connected to one edge router. Further, the connection from the edge router to the users is not considered as a bottleneck. Thus, the edge router is considered as a combination of users and is the last hop for our analysis. We also note that the link from the edge router to the end users is not controlled by the service provider and thus cannot be considered for optimized resource allocation from the network. The service provider wishes to optimize the links it controls for efficient quality of experience to the end user.

A set of r video files (denoted by $i = 1, \ldots, r$) are stored in the datacenter, where video file i is divided into L_i equal-size segments each of length τ seconds. We assume that the first $L_{j,i}$ chunks of video i are stored on cache server j. Even though

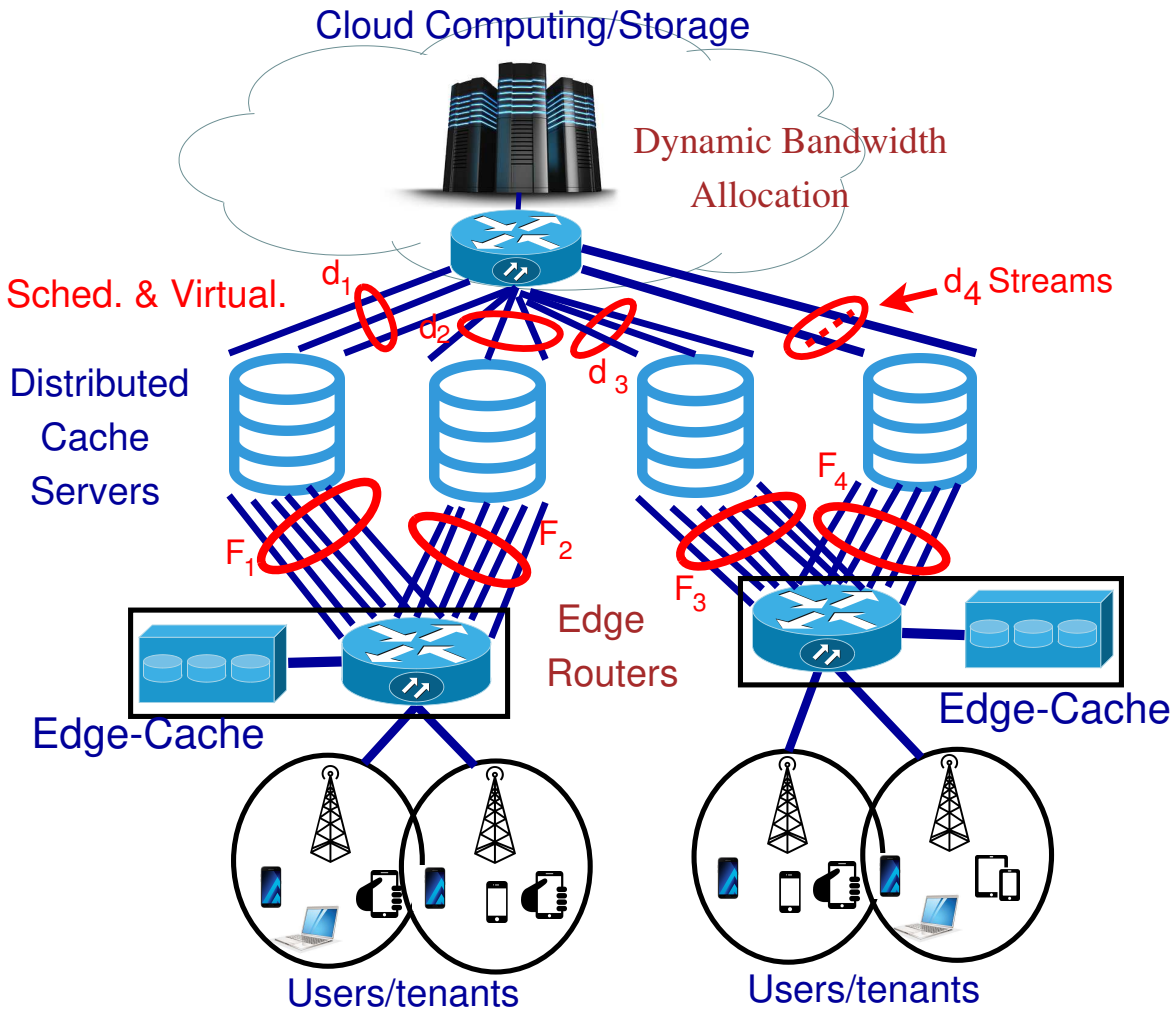

Fig. 5.1.: An illustration of our system model for video content delivery, consisting of a datacenter, four cache servers ($m = 4$), and 2 edge routers. d_j and F_j parallel connections are assumed between datacenter and cache server j, and datacenter and edge router, respectively.

we consider a fixed cache placement, we note that $L_{j,i}$ are optimization variables and can be updated when sufficient arrival rate change is detected.

We assume that the bandwidth between the data center and the cache server j (service edge router ℓ) is split into d_j parallel streams, where the streams are denoted

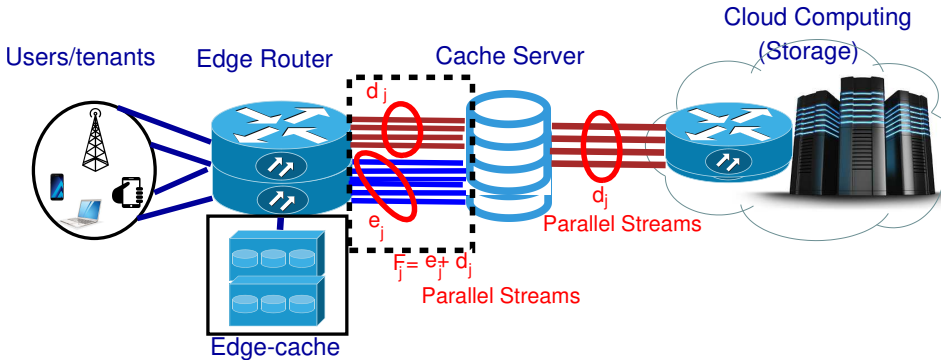

Fig. 5.2.: A schematic illustrates the parallel streams setup between the different system model components.

as $PS^{(d,j)}_{\beta_j,\ell}$ for $\beta_j = 1, \cdots, d_j$. Further, the bandwidth between the cache server j and the edge router ℓ is divided into $f^{(\ell)}_j$ parallel streams, denoted as $PS^{(f,j)}_{\zeta_j,\ell}$ for $\zeta^{(\ell)}_j = 1, \cdots, f^{(\ell)}_j$, and $\ell = 1, 2, \cdots, R$. Multiple parallel streams are assumed for video streaming since multiple video downloads can happen simultaneously. Since we care about stall duration, obtaining multiple videos simultaneously is helpful as the stall durations of multiple videos can be improved. We further assume that $f^{(\ell)}_j$ PSs are divided into two set of streams d_j and $e^{(\ell)}_j$. This setup is captured in Figure 5.2. The first d_j parallel streams are denoted as $PS^{(\bar{d},j)}_{\beta_j,\ell}$ for $\beta_j = 1, \cdots, d_j$, and for all ℓ while the remaining $e^{(\ell)}_j$ streams are denoted as $PS^{(e,j)}_{\nu_j,\ell}$ for $\nu_j = 1, \cdots, e^{(\ell)}_j$. In order to consider the splits, $PS^{(d,j)}_{\beta_j,\ell}$ gets $\{w^{(d)}_{j,\beta_j,\ell}, \beta_j = 1, \ldots, d_j\}$ fraction of the bandwidth from the data center to the cache server j. Similarly, $PS^{(\bar{d},j)}_{\beta_j,\ell}$ gets $\{w^{(\bar{d})}_{j,\beta_j,\ell}, \beta_j = 1, \ldots, d_j\}$ fraction of bandwidth from cache server j to the edge router ℓ and $PS^{(e,j)}_{\nu_j,\ell}$ gets $\{w^{(e)}_{j,\nu_j,\ell}, \nu_j = 1, \ldots, e^{(\ell)}_j\}$ fraction of bandwidth from cache server j to the edge router ℓ. Thus, we have

$$\sum_{\beta_j=1}^{d_j} w^{(d)}_{j,\beta_j,\ell} \leq 1, \qquad \sum_{\beta_j=1}^{d_j} w^{(\bar{d})}_{j,\beta_j,\ell} + \sum_{\nu_j=1}^{e^{(\ell)}_j} w^{(e)}_{j,\nu_j,\ell} \leq 1, \tag{5.1}$$

for all $j = 1, \cdots, m$ and $\ell = 1, \cdots, R$. We note that the sum of weights may be less than 1 and some amount of the bandwidth may be wasted. While the optimal solution will satisfy this with equality since for better utilization, we do not need to explicitly enforce the equality constraint. We note that if the cache server serves

multiple edge routers, the parallel streams between cloud storage and cache server will be the sum of d_j to each edge router thus making the problem separated for each edge router. For ease, we will sometime omit ℓ to focus on links to one edge router only and the same procedure can be used for each edge router.

We assume that the service time of a segment for data transfer from the data center to the cache server j is shifted-exponential with rate $\alpha_{j,\ell}^{(d)}$ and a shift of $\eta_{j,\ell}^{(d)}$ while that between the cache server j and the edge router ℓ is also shifted-exponential with rate $\alpha_{j,\ell}^{(f_j)}$ and a shift of $\eta_{j,\ell}^{(f_j)}$. The shifted exponential distribution can be seen as an approximation of the realistic service time distribution in the prior works, e.g., [78], and references therein. We also note that the rate of a parallel stream is proportional to the bandwidth split. Thus, the service time distribution of $PS_{\beta_j,\ell}^{(d,j)}$, $PS_{\beta_j,\ell}^{(\bar{d},j)}$, and $PS_{\nu_j,\ell}^{(e,j)}$, denoted as $\alpha_{j,\beta_j,\ell}^{(d)}$, $\alpha_{j,\beta_j,\ell}^{(\bar{d})}$, and $\alpha_{j,\nu_j,\ell}^{(\bar{d})}$, respectively, and are given as follows.

$$\alpha_{j,\beta_j,\ell}^{(d)} = w_{j,\beta_j,\ell}^{(d)} \alpha_j^{(d)}, \tag{5.2}$$

$$\alpha_{j,\beta_j,\ell}^{(\bar{d})} = w_{j,\beta_j,\ell}^{(\bar{d})} \alpha_{j,\ell}^{(f_j)}, \quad \alpha_{j,\nu_j,\ell}^{(e)} = w_{j,\nu_j,\ell}^{(e)} \alpha_j^{(f_j)}, \tag{5.3}$$

for all β_j, ν_j, and ℓ. We further define the moment generating functions of the service times of $PS_{\beta_j,\ell}^{(d,j)}$, $PS_{\beta_j,\ell}^{(\bar{d},j)}$, and $PS_{\nu_j,\ell}^{(e,j)}$ as $M_{j,\beta_j,\ell}^{(d)}$, $M_{j,\beta_j,\ell}^{(\bar{d})}$, and $M_{j,\nu_j,\ell}^{(\bar{d})}$, which are defined as follows.

$$M_{j,\beta_j,\ell}^{(d)} = \frac{\alpha_{j,\beta_j,\ell}^{(d)} e^{\eta_{j,\beta_j,\ell}^{(d)} t}}{\alpha_{j,\beta_j,\ell}^{(d)} - t}, \tag{5.4}$$

$$M_{j,\beta_j,\ell}^{(\bar{d})} = \frac{\alpha_{j,\beta_j,\ell}^{(\bar{d})} e^{\eta_{j,\beta_j,\ell}^{(\bar{d})} t}}{\alpha_{j,\beta_j,\ell}^{(\bar{d})} - t}, \tag{5.5}$$

$$M_{j,\nu_j,\ell}^{(e)} = \frac{\alpha_{j,\nu_j,\ell}^{(e)} e^{\eta_{j,\nu_j,\ell}^{(e)} t}}{\alpha_{j,\nu_j,\ell}^{(e)} - t} \tag{5.6}$$

We also assume that there is a start-up delay of d_s (in seconds) for the video which is the duration in which the content can be buffered but not played.

5.3.2 Edge-cache Model

Edge cache $\ell \in \{1, 2, \cdots, R\}$, where R is the total number of edge-routers, stores the video content closer to end users. This improves the QoE to end users. We assume a limited cache size at the edge-router (edge-cache) of a maximum capacity of $C_{\ell,e}$ seconds, at edge-router ℓ. When a file is requested by the user, the edge cache is first checked to see if the file is there completely or partly (in case some other user is watching that content). If the file is not in the edge cache, the space for this video file is created in the edge cache, and a file or some other video files have to be evicted so as not to violate the space constraint.

We consider edge cache policy as follows. The file i is removed from edge cache if it has not been accessed in time $\omega_{i,\ell}$ after its last request time from edge-router ℓ. The parameter $\omega_{i,\ell}$ is a variable that can be optimized based on the file preference and its placement in the CDN cache. This caching policy is motivated by LRU since the file is evicted if it has not been used in some time in the past. The key advantages of this approach is that (i) It is tunable, in the sense that the parameters $\omega_{i,\ell}$ can be optimized, and (ii) the performance of the policy is easier to optimize as compared to LRU. When a file i is requested, and someone has already requested from edge router ℓ in the last $\omega_{i,\ell}$ time units, the file is obtained from the edge router. Even if the file may not be completely in the edge-router yet (not yet finished downloading), the downloaded part is given directly to the new user and the remaining content is delivered as it becomes available to the edge cache. This is akin to multicasting the remaining part of the video to multiple users [79].

An illustration of the evolution of caching policy is illustrated in Figure 5.3, where the index ℓ is omitted since we consider the procedure at a single edge router. Video file i is requested at three times t_1, t_2, and t_3. At t_1, the file enters the edge cache. Since it had not been requested in ω_i time units, it is evicted. When the file is again requested at t_2, the space for the file is reserved in the edge cache. The file, when requested at t_3 is within the ω_i duration from t_2 and thus will be served from the

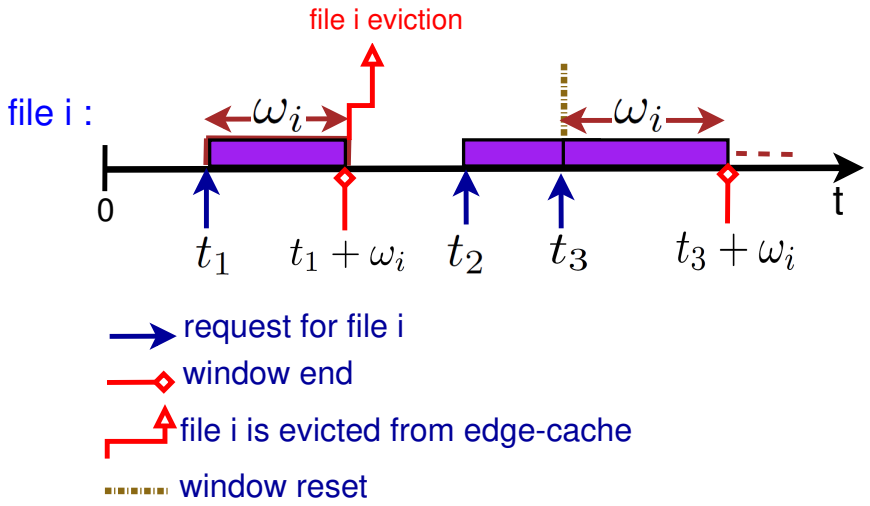

Fig. 5.3.: A schematic showing an example of how a time-line of a file i can change according to the request pattern of a video file i.

edge cache. If the file is still not in the edge cache completely, it will obtain the part already there directly while the remaining part will be streamed as it becomes available. Since the file i was not requested for time ω_i after t_3, the file is again evicted from the edge cache.

We note that the arrival rate of the files is random. Thus, this file eviction policy may not satisfy the maximum edge cache constraint at all times. In order to handle this, we will first assume for the analytical optimization that the probability that the cache capacity of edge-cache ℓ is violated is bounded by ϵ_ℓ which is small. That could lead us to obtain a rough estimate on the different parameters in the system. The hard constraint on the capacity can be made in run-time, by evicting the files that are closest to be going out based on when they were requested and the corresponding $\omega_{i,\ell}$. This online adaptation will be explained in Appendix A.20.

5.3.3 Queueing Model and Two-stage probabilistic scheduling

If cache server j is chosen for accessing video file i on edge router ℓ, the first $L_{j,i}$ chunks are obtained from one of the $e_j^{(\ell)}$ parallel streams $PS_{\nu_j,\ell}^{(e,j)}$. Further, the remaining $L_i - L_{j,i}$ chunks are obtained from the data center where a choice of β_j

is made from $1, \cdots, d_j$ and the chunks are obtained from the stream $PS_{\beta_j}^{(d,j)}$ which after being served from this queue is enqueued in the queue for the stream $PS_{\beta_j,\ell}^{(\bar{d},j)}$. However, if video file i is already requested at time t_i within a window of size ω_i, the request will be served from the edge-cache and will not be sent to a higher level in the hierarchy, e.g., cache server.

We assume that the arrival of requests at the edge router ℓ for each video i form an independent Poisson process with a known rate $\lambda_{i,\ell}$. In order to serve the request for file i, we need to choose three things - (i) Selection of Cache server j, (ii) Selection of ν_j to determine one of $PS_{\nu_j,\ell}^{(e,j)}$ streams to deliver cached content, (iii) Selection of β_j to determine one of $PS_{\beta_j}^{(d,j)}$ streams from the data-center which automatically selects the stream $PS_{\beta_j,\ell}^{(\bar{d},j)}$ from the cache server, to obtain the non-cached content from the datacenter. Thus, we will use a two-stage probabilistic scheduling to select the cache server and the parallel streams. For a file request at edge-router ℓ, we choose server j with probability $\pi_{i,j,\ell}$ for file i randomly. Further, having chosen the cache server, one of the e_j streams is chosen with probability $p_{i,j,\nu_j,\ell}$. Similarly, one of the d_j streams is chosen with probability $q_{i,j,\beta_j,\ell}$. We note that these probabilities only have to satisfy

$$\sum_{j=1}^{m} \pi_{i,j,\ell} = 1 \forall i, \ell; \tag{5.7}$$

$$\sum_{\nu_j=1}^{e_j^{(\ell)}} p_{i,j,\nu_j,\ell} = 1 \forall i, j, \ell; \tag{5.8}$$

$$\sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j,\ell} = 1 \forall i, j, \ell, \tag{5.9}$$

$$\pi_{i,j,\ell}, p_{i,j,\nu_j,\ell}, q_{i,j,\beta_j,\ell} \geq 0 \forall i, j, \beta_j, \nu_j, \ell \tag{5.10}$$

We note that since file i is removed from the edge cache after time ω_i, the requests at the cache server are no longer Poisson. We note that this could be alleviated by assuming that every time the file is requested, the time ω_i is chosen using an exponential distribution. This change of distribution will make the distribution of requests at the cache server Poisson thus alleviating the issue. This is because if ω_i follows an

exponential distribution (i.e., not fixed) with parameter ν_i, the probability that the request of video file i is directed to the distributed cache servers and/or central server is given by $P(\widetilde{t}_i > \omega_i) = \frac{\nu_i}{\nu_i+\lambda_i}$. This result follows since $\widetilde{t}_i$ and ω_i are exponentially distributed with paramters λ_i and ν_i, respectively. However, in the following, we will assume constant ω_i, while still approximate the request pattern at cache servers as Poisson which holds when the times for which file remains in the edge cache is chosen using an exponential distribution. This approximation turns out to be quite accurate as will be shown in the evaluation results. Further, such approximations of Poisson arrivals are widely used in the literature in similar fashions. In particular, it is used to characterize the coherence time of an LRU-based caching, e.g., see [52] and references therein. Further, in [80] (Ch.9, page 470) authors approximate the arrivals of new and retransmitted packets in CSMA protocol as Poisson even though they are not due to the dependencies between them.

Since sampling of Poisson process is Poisson, and superposition of independent Poisson processes is also Poisson, we get the aggregate arrival rate at $PS^{(d,j)}_{\beta_j,\ell}$, $PS^{(\bar{d},j)}_{\beta_j,\ell}$, and $PS^{(e,j)}_{\nu_j,\ell}$, denoted as $\Lambda^{(d)}_{j,\beta_j,\ell}$, $\Lambda^{(\bar{d})}_{j,\beta_j,\ell}$, and $\Lambda^{(e)}_{j,\nu_j,\ell}$, respectively are given as follows.

$$\Lambda^{(d)}_{j,\beta_j,\ell} = \sum_{i=1}^{r} \lambda_{i,\ell}\pi_{i,j,\ell}q_{i,j,\beta_j,\ell}e^{-\lambda_{i,\ell}\omega_{i,\ell}}; \quad \Lambda^{(\bar{d})}_{j,\beta_j,\ell} = \Lambda^{(d)}_{j,\beta_j,\ell}, \tag{5.11}$$

$$\Lambda^{(e)}_{j,\nu_j,\ell} = \sum_{i=1}^{r} \lambda_{i,\ell}\pi_{i,j,\ell}p_{i,j,\nu_j,\ell}e^{-\lambda_{i,\ell}\omega_i} \tag{5.12}$$

Remark 1 *We note that since shift $\boldsymbol{\eta}$ is not zero, the requests at the cache servers (i.e., the output of the first queues) are no longer Poisson. However, we note that this could be alleviated by assuming that this shift is relatively very small compared to the exponential rate $\boldsymbol{\alpha}$. This assumption will make the requests pattern at the storage servers Poisson, thus alleviating this issue. In the following, we will assume a Poisson approximation for simplicity, while still approximate the request pattern at storage servers as Poisson which holds when the shift approaches zero. This approximation turns out to be quite accurate as will be shown in the evaluation results. Further,*

such approximations of Poisson arrivals are widely used in the literature in similar fashions, e.g., see [50] and references therein for further details.

Assumption 1 *When the service time distribution of datacenter server (first queue) is given by shifted exponential distribution, the arrivals at the cache servers (second queue) are Poisson.*

5.3.4 Distribution of Edge Cache Utilization

We will now investigate the distribution of the edge-cache utilization at any time. This will help us in bounding the probability that the edge cache is more than the capacity of the cache. In the analytic part, we will bound this probability. However, the online adaptations in Appendix A.20 will provide an adaptation to maintain the maximum edge cache capacity constraint at all times.

Let $X_{i,\ell}$ be the random variable corresponding to amount of space in the edge-cache ℓ for video file i. Since the file arrival rate is Poisson, and the file is in the edge-cache ℓ if it has been requested in the last ω_i seconds. Then, $X_{i,\ell}$ is given as

$$X_{i,\ell} = \begin{cases} \tau\, L_i & \text{with prob. } 1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}} \\ 0 & \text{with prob. } e^{-\lambda_{i,\ell}\omega_{i,\ell}} \end{cases} \tag{5.13}$$

where $1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}}$ is the probability that file i is requested within a window-size of $\omega_{i,\ell}$ time units. The total utilization of the edge-cache j is given as

$$X_\ell = \sum_{i=1}^{r} X_{i,\ell} \tag{5.14}$$

The mean and variance of X_ℓ can be found to be $\sum_i^r \tau L_i(1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}}$ and $\sum_i^r (\tau L_i)^2 e^{-\lambda_{i,\ell}\omega_{i,\ell}}(1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}})$, respectively. Since r is large, we will approximate the distribution of X_j by a Gaussian distribution with mean $\sum_i^r \tau L_i(1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}}$ and variance $\sum_i^r (\tau L_i)^2 e^{-\lambda_{i,\ell}\omega_{i,\ell}}(1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}})$. This distribution is then used as a constraint in the design of $\omega_{i,\ell}$, where the constraint bounds the probability that the edge cache utilization is higher than the maximum capacity of the edge cache. Since X_ℓ can

be well approximated by a Gaussian distribution, the edge cache utilization, can be probabilistically bounded as follows,

$$\int_{C_{\ell,e}}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \leq \epsilon_\ell \tag{5.15}$$

where $\mu = \sum_i^r \tau L_i(1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}})$, $\sigma^2 = \sum_i^r (\tau L_i)^2 e^{-\lambda_{i,\ell}\omega_{i,\ell}}(1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}})$ are the mean and variance, respectively.

5.4 Stall Duration Tail Probability

This section will characterize the stall duration tail probability using the two-stage probabilistic scheduling and allocation of bandwidth weights. We note that the arrival rates are given in terms of the video files, and the service rate above is provided in terms of segment at each server. The analysis would require detailed consideration of the different segments in a video. In this section, we will assume that the edge-router for the request ℓ is known, and thus we omit the subscript/superscript ℓ to simplify notations.

In order to find the stall durations, we first consider the case where file i is not in the edge cache and has to be requested from the CDN. We also assume that the cache server j is used, with the streams β_j and ν_j known. We will later consider the distribution of these choices to compute the overall metric. In order to compute stall durations, we would first calculate the download time of each of the video segment, which accounts for the first $L_{j,i}$ segments at the cache j and the later $L_i - L_{j,i}$ segments at the server. After the download times are found, the play times of the different contents are found. The detailed calculations are shown in Appendix A.13, where the distribution of $T^{(g)}_{i,j,\beta_j,\nu_j}$, the time that segment g begins to play at the client i given that it is downloaded from β_j and ν_j queues, is found. The stall duration for the request of file i from β_j queue, ν_j queue and server j, if not in the edge-cache, i.e., $\Gamma_U^{(i,j,\beta_j,\nu_j)}$ is given as

$$\Gamma_U^{(i,j,\beta_j,\nu_j)} = T^{(L_i)}_{i,j,\beta_j,\nu_j} - d_s - (L_i - 1)\,\tau, \tag{5.16}$$

as explained in Appendix A.13. We use this expression to derive a tight bound on the SDTP.

The stall duration tail probability of a video file i is defined as the probability that the stall duration is greater than a pre-defined threshold σ. Since exact evaluation of stall duration is hard [25,81], we cannot evaluate $\Pr\left(\Gamma_{tot}^{(i)} \geq \sigma\right)$ in closed-form, where $\Gamma_{tot}^{(i)}$ is random variable indicating the overall stall duration for file i. In this section, we derive a tight upper bound on the SDTP through the two-stage Probabilistic Scheduling as follows.

We first note that the expression in equation (A.50) (Appendix A.13) accounts only for the stalls that would be incurred if the video segments are not accessed from the edge-cache (including stored, or multicasted). However, the user would experience lower stalls if the requested content is accessed from the edge-cache. Thus, we need an expectation over the choice of whether the file is accessed from the edge server, and the choice of (j, β_j, ν_j) in addition to the queue statistics. For a video file i requested at time $\widetilde{t}_i$ after the last request for file i, the stall duration for the request of file i can be expressed as follows:

$$\Gamma_{tot}^{(i)} \stackrel{d}{=} \begin{cases} \left(\Gamma_{tot}^{(i)} - \widetilde{t}_i\right)^+ & 0 \leq \widetilde{t}_i \leq \omega_i \\ \Gamma_U^{(i,j,\beta_j,\nu_j)} & \widetilde{t}_i > \omega_i \end{cases} \tag{5.17}$$

where $\stackrel{d}{=}$ means equal in distribution. This is because if file i is requested again within ω_i time, then the multicast or stored file can lead to the reduced stall duration based on how much time has passed since the last request. Further, if the file has not been requested in the last ω_i time units, then the file has to be obtained from the CDN, and thus the expression of random variable $\Gamma_{tot}^{(i)}$ also includes randomness over the choice of (j, β_j, ν_j) in this case. From (5.17), we can obtain the following result.

Lemma 15 *For a given choice of* (j, β_j, ν_j), $\mathbb{E}\left[e^{h_i \Gamma_{tot}^{(i)}}\right]$ *is bounded as*

$$\mathbb{E}\left[e^{h_i \Gamma_{tot}^{(i)}}\right] \leq \widetilde{c} + \widetilde{a}\,\mathbb{E}\left[e^{h_i \Gamma_U^{(i,j,\beta_j,\nu_j)}}\right] \tag{5.18}$$

where $c = 1 - e^{-\lambda_i \omega_i}$, $a = e^{-\lambda_i \omega_i}$, $b = \left[1 - \frac{\lambda_i}{\lambda_i + h_i}\left(1 - e^{-(\lambda_i + h_i)\omega_i}\right)\right]$, $\widetilde{c} = c/b$ *and* $\widetilde{a} = a/b$.

Proof The proof is provided in Appendix A.14. ■

We next derive $\mathbb{E}\left[e^{h_i D^{(v)}_{i,j,\beta_j,\nu_j}} | (j, \beta_j, \nu_j)\right]$ using the following two lemmas, which will be used in the main result. The key idea is that we characterize the download and play times of each segments and use them in determining the SDTP of each video file request.

Lemma 16 *For* $v \leq L_{j,i}$, $\mathbb{E}\left[e^{h_i D^{(v)}_{i,j,\beta_j,\nu_j}} | (j, \beta_j, \nu_j)\right]$ *is given by*

$$\begin{aligned} & \mathbb{E}\left[e^{h_i D^{(v)}_{i,j,\beta_j,\nu_j}} | (j, \beta_j, \nu_j)\right] \\ = \; & \frac{(1 - \rho^{(e)}_{j,\nu_j}) t_i}{t_i - \Lambda^{(e)}_{j,\nu_j}(B^{(e)}_{j,\nu_j}(t_i) - 1)} \left(\frac{\alpha^{(e)}_{j,\nu_j} e^{\eta^{(e)}_{j,\nu_j} t}}{\alpha^{(e)}_{j,\nu_j} - t_i}\right)^v \end{aligned} \tag{5.19}$$

Proof The proof follows from (A.27) in Appendix A.13 by replacing g by v and rearranging the terms in the result. ■

Lemma 17 *For* $v \geq (L_{j,i} + 1)$, $\mathbb{E}\left[e^{h_i D^{(v)}_r} | (j, \beta_j, \nu_j)\right]$ *is given by*

$$\begin{aligned} & \mathbb{E}\left[e^{h_i D^{(v)}_r} | (j, \beta_j, \nu_j)\right] \\ \leq \; & \mathbb{E}\left[e^{h_i U_{i,j,\beta_j,v,L_{j,i}}} | (j, \beta_j, \nu_j)\right] + \\ & \sum_{w=L_{j,i}+1}^{v} \frac{(1 - \rho^{(d)}_{j,\beta_j}) t_i}{h_i - \Lambda^{(d)}_{j,\beta_j}(B^{(d)}_{j,\beta_j}(h_i) - 1)} \times \\ & \left(\frac{\alpha^{(d)}_{j,\beta_j} e^{\eta^{(d)}_{j,\beta_j} h_i}}{\alpha^{(d)}_{j,\beta_j} - h_i}\right)^{w - L_{j,i} - 1} \left(\frac{\alpha^{(\bar{d})}_{j,\beta_j} e^{\eta^{(\bar{d})}_{j,\beta_j} h_i}}{\alpha^{(\bar{d})}_{j,\beta_j} - h_i}\right)^{v - w + 1}, \end{aligned} \tag{5.20}$$

where $\mathbb{E}\left[e^{h_i U_{i,j,\beta_j,v,L_{j,i}}} | (j, \beta_j, \nu_j)\right]$ *and* $B^{(d)}_{j,\beta_j}$ *are given in Appendix A.13, equations* (A.39), *and* (A.41), *respectively.*

Proof The proof is provided in Appendix A.15. ■

Corollary 3 *The (expected) time to the first chunk (TTFC) can be obtained from equation* (A.75) *(Lemma A.19.1) by setting* $d_s = 0$ *and* $g_i = 1$.

Using these expressions, the following theorem summarizes the stall duration tail probability for file i. We include the edge router index ℓ in all the expressions in the result for the ease of using it in the following section.

Theorem 5.4.1 *The stall distribution tail probability for video file i requested through edge router ℓ is bounded by*

$$Pr\left(\Gamma_{tot}^{(i,\ell)} \geq \sigma\right) \leq$$
$$\sum_{j=1}^{m} \pi_{i,j,\ell} \times \left[\overline{c}_\ell + \widetilde{a}_\ell\, e^{-h_i\sigma} + \overline{a}_\ell \sum_{\nu_j=1}^{e_j^{(\ell)}} p_{i,j,\nu_j,\ell} \times \right.$$
$$\left. \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j,\ell} e^{h_i L_i \tau} \times \left(\delta^{(e,\ell)} + \delta^{(\overline{d},\ell)} + \delta^{(d,\overline{d},\ell)}\right)\right. \tag{5.21}$$

for $\rho_{j,\beta_j}^{(d)} < 1$, $\rho_{j,\beta_j,\ell}^{(\overline{d})} < 1$, $\rho_{j,\nu_j,\ell}^{(e)} < 1$, *where the auxiliary variables in the statement of the Theorem are defined as*

$$\delta^{(e,\ell)} = \frac{\widetilde{M}_{j,\nu_j}^{(e,\ell)}(h_i)(1-\rho_{j,\beta_j,\ell}^{(e)})t_i((\widetilde{M}_{j,\nu_j}^{(e,\ell)}(h_i))^{L_{j,i}}-1)}{(h_i - \Lambda_{j,\beta_j,\ell}^{(e)}(B_{j,\beta_j}^{(e,\ell)}(h_i)-1))(\widetilde{M}_{j,\nu_j}^{(e,\ell)}(h_i))-1)} \tag{5.22}$$

$$\delta^{(\overline{d},\ell)} = \frac{(1-\rho_{j,\beta_j,\ell}^{(\overline{d})})t_i(\widetilde{M}_{j,\nu_j}^{(\overline{d},\ell)}(h_i))^{L_{j,i}-L_i}}{h_i - \Lambda_{j,\beta_j,\ell}^{(\overline{d})}(B_{j,\beta_j}^{(\overline{d},\ell)}(h_i)-1)} \tag{5.23}$$

$$\delta^{(d,\overline{d},\ell)} = \gamma^{(d,\ell)}\left(\frac{(\widetilde{M}_{j,\beta_j}^{(d,\overline{d},\ell)}(h_i))^{L_i-L_{j,i}} - (L_i - L_{j,i})}{(\widetilde{M}_{j,\beta_j}^{(d,\overline{d},\ell)}(h_i))-1} + \xi_{i,j,\beta_j,}^{(d,\overline{d},\ell)}\right) \tag{5.24}$$

$$\xi_{i,j,\beta_j,}^{(d,\overline{d},\ell)} = \frac{\widetilde{M}_{j,\beta_j}^{(d,\overline{d},\ell)}(h_i)\left((\widetilde{M}_{j,\beta_j}^{(d,\overline{d},\ell)}(h_i))^{L_i-L_{j,i}-1}-1\right)}{(\widetilde{M}_{j,\beta_j}^{(d,\overline{d},\ell)}(h_i))-1} \tag{5.25}$$

$$\gamma^{(d,\ell)} = \frac{(1-\rho_{j,\beta_j}^{(d)})t_i(\widetilde{M}_{j,\beta_j}^{(\overline{d},\ell)}(h_i))^{L_i+1}}{\left[h_i - \Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(h_i)-1)\right](\widetilde{M}_{j,\beta_j}^{(d)}(h_i))^{L_{j,i}+1}} \tag{5.26}$$

$$\widetilde{M}^{(d)}_{j,\beta_j}(h_i) = \frac{\alpha^{(d)}_{j,\beta_j} e^{\eta_{j,\beta_j} - h_i\tau}}{\alpha^{(d)}_{j,\beta_j} - h_i}, \tag{5.27}$$

$$\widetilde{M}^{(\bar{d},\ell)}_{j,\beta_j}(h_i) = \frac{\alpha^{(\bar{d})}_{j,\beta_j,\ell} e^{\eta_{j,\beta_j,\ell} - h_i\tau}}{\alpha^{(\bar{d})}_{j,\beta_j,\ell} - h_i} \tag{5.28}$$

$$\widetilde{M}^{(e,\ell)}_{j,\nu_j}(h_i) = \frac{\alpha^{(e)}_{j,\nu_j,\ell} e^{\eta_{j,\nu_j,\ell} - h_i\tau}}{\alpha^{(e)}_{j,\nu_j,\ell} - h_i}, \tag{5.29}$$

$$\widetilde{M}^{(d,\bar{d})}_{j,\beta_j,\ell}(h) = \frac{\alpha^{(d)}_{j,\beta_j}(\alpha^{(\bar{d})}_{j,\beta_j,\ell} - h) e^{\eta^{(d)}_{j,\beta_j} h}}{\alpha^{(\bar{d})}_{j,\beta_j,\ell}(\alpha^{(d)}_{j,\beta_j} - h) e^{\eta^{(\bar{d})}_{j,\beta_j,\ell} h}}, \ \ \forall j, \beta_j \tag{5.30}$$

$$\bar{c}_\ell = \frac{(1 - e^{-\lambda_{i,\ell}\omega_{i,\ell}}) e^{-h_i\sigma}}{1 - \frac{\lambda_i}{\lambda_i + h_i}\left(1 - e^{-(\lambda_i + h_i)\omega_i}\right)} \tag{5.31}$$

$$\widetilde{a}_\ell = \frac{e^{-\lambda_{i,\ell}\omega_{i,\ell}}}{1 - \frac{\lambda_i}{\lambda_i + h_i}\left(1 - e^{-(\lambda_i + h_i)\omega_i}\right)} \tag{5.32}$$

$$\bar{a}_\ell = \widetilde{a}_\ell e^{-h_i(\sigma + d_s + (L_i - 1)\tau)} \tag{5.33}$$

Proof The detailed steps are provided in Appendix A.16.

■

We note that $\delta^{(e,\ell)} = \delta^{(\bar{d},\ell)} = 0$, if the storage server nodes are not hosting the requested video files and $\delta^{(\bar{d},d,\ell)}$ has nonzero value only if some sWe can also derive the mean stall duration for video file i in a similar fashion. The interested reader is referred to Appendix A.19 for detailed treatment of this metric.

5.5 Optimization Problem Formulation and Proposed Algorithm

5.5.1 Problem Formulation

We define $\boldsymbol{\pi} = (\pi_{i,j,\ell} \forall i = 1, \cdots, r \text{ and } j = 1, \cdots, m, \ell = 1, \cdots, R), \boldsymbol{p} = (p_{i,,j,\nu_j,\ell}, (\nu_j, \ell) \in \{(\nu_j, \ell) : \nu_j \in \{1, \ldots, e^{(\ell)}_j\}, \ell \in \{1, \ldots, R\}\}, \forall i = 1, \cdots, r\,, j = 1, \cdots, m\,, \nu_j = 1, \ldots e^{(\ell)}_j, \ell = 1, \cdots, R), \boldsymbol{q} = (q_{i,,j,\beta_j,\ell} \forall i = 1, \cdots, r\,, j = 1, \cdots, m\,, \beta_j = 1, \ldots d_j, \ell =$

$1, \cdots, R), \boldsymbol{h} = (h_1, t_2, \ldots, h_r)$, $\boldsymbol{w}^{(\bar{d})} = (w^{(\bar{d})}_{j,1,\ell}, \ldots, w^{(d)}_{j,d_j,\ell},$ and $j = 1, \cdots, m, \ell = 1, \cdots, R)$, $\boldsymbol{w}^{(e)} = (w^{(e)}_{j,1,\ell}, w^{(e)}_{j,2,\ell}, \ldots, w^{(e)}_{j,e^{(\ell)}_j,\ell},$ and $j = 1, \cdots, m, \ell = 1, \ldots, R)$, $\boldsymbol{w}^{(d)} = (w^{(d)}_{j,1,\ell}, \ldots, w^{(d)}_{j,d_j,\ell},$ and $j = 1, \cdots, m), \boldsymbol{L} = (L_{j,i}, \forall i = 1,$ and $j = 1, \cdots, m)$ and $\boldsymbol{\omega} = (\omega_{i,\ell}, \forall i, \ell)$. Our goal is to minimize the SDTP over the choice of cache and datacenter access decisions, bandwidth allocation weights, portion (number) of cached segments, time window over which we maintain the video files at edge-cache and auxiliary bound parameters.

To incorporate for weighted fairness and differentiated services, we assign a positive weight $\kappa_{i,\ell}$ for each file i. Without loss of generality, each file i is weighted by the arrival rate $\lambda_{i,\ell}$ in the objective (so larger arrival rates are weighted higher). However, any other weights can be incorporated to accommodate for weighted fairness or differentiated services. Let $\overline{\lambda} = \sum_{i,\ell} \lambda_{i,\ell}$ be the total arrival rate. Hence, $\kappa_{i,\ell} = \lambda_{i,\ell}/\overline{\lambda}$ is the ratio of file i requests. Hence, the objective is the minimization of stall duration tail probability, averaged over all the file requests, and is given as $\sum_{i,\ell} \frac{\lambda_{i,\ell}}{\overline{\lambda}} \Pr\left(\Gamma^{(i)} \geq \sigma\right)$. By using the expression for SDTP in Section 5.4, the optimization problem can be formulated as follows.

$$\sum_{\ell=1}^{R}\sum_{i=1}^{r}\frac{\lambda_{i,\ell}}{\overline{\lambda}_{i,\ell}}\sum_{j=1}^{m}\pi_{i,j,\ell}\times\Bigg[\overline{c}_{\ell}+\widetilde{a}_{\ell}\,e^{-h_i\sigma}+\overline{a}_{\ell}\sum_{\nu_j=1}^{e_j}p_{i,j,\nu_j,\ell}\times$$

$$\sum_{\beta_j=1}^{d_j}q_{i,j,\beta_j,\ell}e^{h_iL_i\tau}\times\left(\delta^{(e)}+\delta^{(\overline{d})}+\delta^{(d,\overline{d})}\right) \tag{5.34}$$

s.t.

$$(5.1)-(5.12),(5.15),(5.22)-(5.33) \tag{5.35}$$

$$\rho^{(d)}_{j,\beta_j,\ell}=\sum_{i=1}^{r}\lambda_{i,\ell}\pi_{i,j}q_{i,j,\beta_j,\ell}e^{\lambda_{i,\ell}\omega_{i,\ell}}\frac{L_i-L_{j,i}}{\alpha^{(d)}_{j,\beta_j,\ell}}<1,\ \ \forall j,\beta_j \tag{5.36}$$

$$\rho^{(\overline{d})}_{j,\beta_j,\ell}=\sum_{i=1}^{r}\lambda_{i,\ell}\pi_{i,j,\ell}q_{i,j,\beta_j,\ell}e^{\lambda_{i,\ell}\omega_{i,\ell}}\frac{L_i-L_{j,i}}{\alpha^{(\overline{d})}_{j,\beta_j,\ell}}<1\,\forall j,\beta_j,\ell \tag{5.37}$$

$$\rho^{(e)}_{j,\nu_j,\ell}=\sum_{i=1}^{r}\lambda_{i,\ell}\pi_{i,j,\ell}p_{i,j,\nu_j,\ell}e^{\lambda_{i,\ell}\omega_{i,\ell}}\frac{L_{j,i}}{\alpha^{(e)}_{j,\nu_j,\ell}}<1,\ \ \forall j,\nu_j,\ell \tag{5.38}$$

$$\sum_{i}L_{j,i}\leq C_j,\ L_{j,i}\geq 0,\ \ \forall i,j \tag{5.39}$$

$$h_i<\alpha^{(d)}_{j,\beta_j},h_i<\alpha^{(\overline{d})}_{j,\beta_j,\ell},h_i<\alpha^{(e)}_{j,\nu_j,\ell},\ \ \forall i,j,\nu_j,\ell \tag{5.40}$$

$$0<h_i-\Lambda^{(d)}_{j,\beta_j}(B^{(d)}_{j,\beta_j,\ell}(h_i)-1),\ \ \forall i,j,\beta_j,\ell \tag{5.41}$$

$$0<h_i-\Lambda^{(\overline{d})}_{j,\beta_j,\ell}(B^{(\overline{d})}_{j,\beta_j,\ell}(h_i)-1),\ \ \forall i,j,\beta_j \tag{5.42}$$

$$0<h_i-\Lambda^{(e)}_{j,\nu_j,\ell}(B^{(e)}_{j,\nu_j,\ell}(h_i)-1),\ \ \forall i,j,\nu_j,\ell \tag{5.43}$$

$$L_{j,i}\in\mathbb{Z} \tag{5.44}$$

var $\ \boldsymbol{\pi},\boldsymbol{q},\boldsymbol{p},\boldsymbol{h},\boldsymbol{w}^{(c)},\boldsymbol{w}^{(d)},\boldsymbol{w}^{(e)},\boldsymbol{L},\boldsymbol{\omega}$

Here, in (5.35), equations (1)−(3) give the feasibility constraints on the bandwidth allocation, while equations (4)−(6) define the MGFs of the service time distributions, equations (7) − (10) give the feasibility of the two-stage probabilistic scheduling and (11) − (12) define the arrival rates at the different queues. Constraints (5.36)–(5.38) ensure the stability of the systems queue (do not blow up to infinity). Constraints (5.40)–(5.43) ensure that the moment generating functions exist. We note that some optimization variables can be combined to form a single optimization variable which results in having only five independent and separable variables as shown below. In

the next subsection, we will describe the proposed algorithm for this optimization problem.

5.5.2 Proposed Algorithm

We first note that the two-stage probabilistic scheduling variables are independent and separable, thus we can combine them and define a single variable $\widetilde{\boldsymbol{\pi}}$ such that $\widetilde{\boldsymbol{\pi}} = (\boldsymbol{\pi}\,, \boldsymbol{p}\,, \boldsymbol{q})$. Similarly, since the bandwidth allocation weights are independent and separable, we concatenate them in a single optimization variable $\boldsymbol{w}$, where $\boldsymbol{w} = \left(\boldsymbol{w}^{(e)}\,, \boldsymbol{w}^{(\overline{d})}\,, \boldsymbol{w}^{(d)}\right)$. Hence, the weighted SDTP optimization problem given in (5.34)-(5.44) is optimized over five set of variables: server and PSs scheduling probabilities $\widetilde{\boldsymbol{\pi}}$ (two-stage scheduling probabilities), auxiliary parameters $\boldsymbol{h}$, bandwidth allocation weights $\boldsymbol{w}$, cache placement $\boldsymbol{L}$, and edge cache window size optimization $\boldsymbol{\omega}$.

Clearly, the problem is non-convex in all the parameters jointly, which can be easily seen in the terms which are product of the different variables. Since the problem is non-convex, we propose an iterative algorithm to solve the problem. The proposed algorithm divides the problem into five sub-problems that optimize one variable while fixing the remaining four. These sub-problems are labeled as (i) Server and PSs Access Optimization: optimizes $\widetilde{\boldsymbol{\pi}}$, for given $\boldsymbol{h}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$, (ii) Auxiliary Variables Optimization: optimizes $\boldsymbol{h}$ for given $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$, (iii) Bandwidth Allocation Optimization: optimizes $\boldsymbol{w}$ for given $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$. (iv) Cache Placement Optimization: optimizes $\boldsymbol{L}$ for given $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{\omega}$, and $\boldsymbol{w}$, (v) Edge-cache Window Size Optimization: optimizes $\boldsymbol{\omega}$ for given $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{L}$, and $\boldsymbol{w}$. The algorithm is summarized as follows.

1. **Initialization**: Initialize $\boldsymbol{h}, \widetilde{\boldsymbol{\pi}}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$ and $\boldsymbol{L}$ in the feasible set.
2. **While Objective Converges**
   (a) Run Server Access Optimization using current values of $\boldsymbol{h}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$ to get new values of $\widetilde{\boldsymbol{\pi}}$

(b) Run Auxiliary Variables Optimization using current values of $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$ to get new values of $\boldsymbol{h}$

(c) Run Bandwidth Allocation Optimization using current values of $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{L}$, and $\boldsymbol{\omega}$, to get new values of $\boldsymbol{w}$.

(d) Run Cache Placement Optimization using current values of $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{w}$, and $\boldsymbol{\omega}$ to get new values of $\boldsymbol{L}$.

(e) Run Edge-cache Window Size Optimization using current values of $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{w}$, and $\boldsymbol{L}$ to get new values of $\boldsymbol{\omega}$.

The proposed algorithm performs an alternating optimization over the different aforementioned dimensions, such that each sub-problem is shown to have convex constraints and thus can be efficiently solved using the iNner cOnVex Approximation (NOVA) algorithm proposed in [57]. The subproblems are explained in detailed in Appendix A.17.

We first initialize $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{w}$, $\boldsymbol{h}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$ $\forall i, j, \nu_j, \beta_j$ such that the choice is feasible for the problem. Then, we do alternating minimization over the five sub-problems defined above. Since each sub-problem can only decrease the objective (properties of convergence of subproblems to a stationary point is given in Appendix A.17) and the overall problem is bounded from below, we have the following result.

Theorem 5.5.1 *The proposed algorithm converges to a stationary solution.*

Appendix A.20 describes how our algorithm can be used in an online fashion to keep track of the systems dynamics at the edge-cache.

5.6 Implementation and Evaluation

In this section, we evaluate our proposed algorithm for weighted stall duration tail probability.

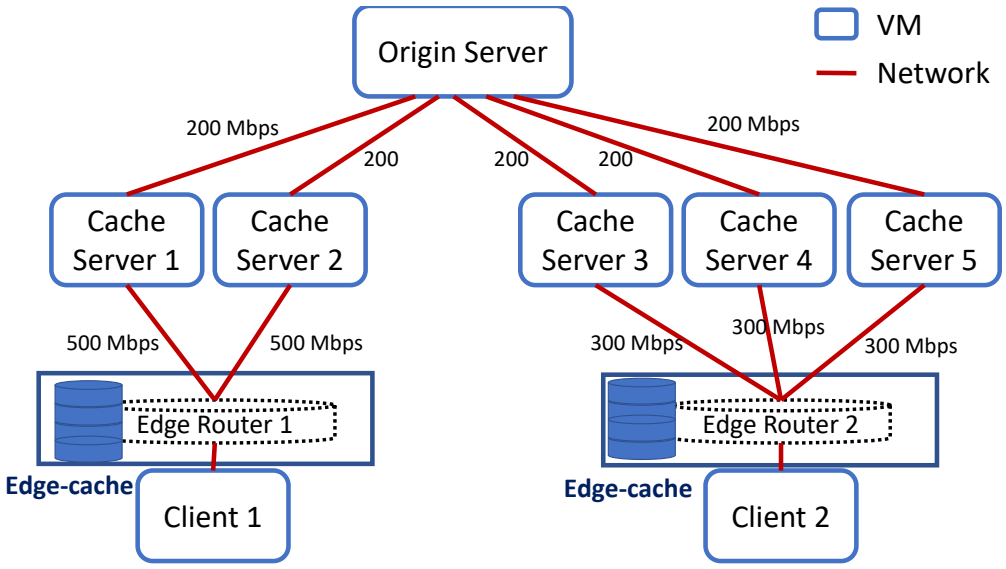

Fig. 5.4.: Testbed in the cloud.

5.6.1 Testbed Configuration and Parameter Setup

We construct an experimental environment in a virtualized cloud environment managed by Openstack [77] to investigate our proposed SDTP framework. We allocated one VM for an origin server and 5 VMs for cache servers intended to simulate two locations (e.g., different states). We implement the proposed online caching mechanism in the edge cache that takes the inputs of $\omega_{i,\ell}$ at each edge router. When a video file is requested, it is stored in the edge-cache for a window size of $\omega_{i,\ell}$ time

Table 5.1.: The value of α_j used in the evaluation results with units of 1/ms. We set $\eta^{(d)}_{j,\beta_j} = \eta^{(\bar{d})}_{j,\beta_j} = \eta^{(e)}_{j,\nu_j} = 14$ ms.

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
82.00	76.53	71.06	65.6	60.13	54.66

Node 7	Node 8	Node 9	Node 10	Node 11	Node 12
49.20	44.28	39.36	34.44	29.52	24.60

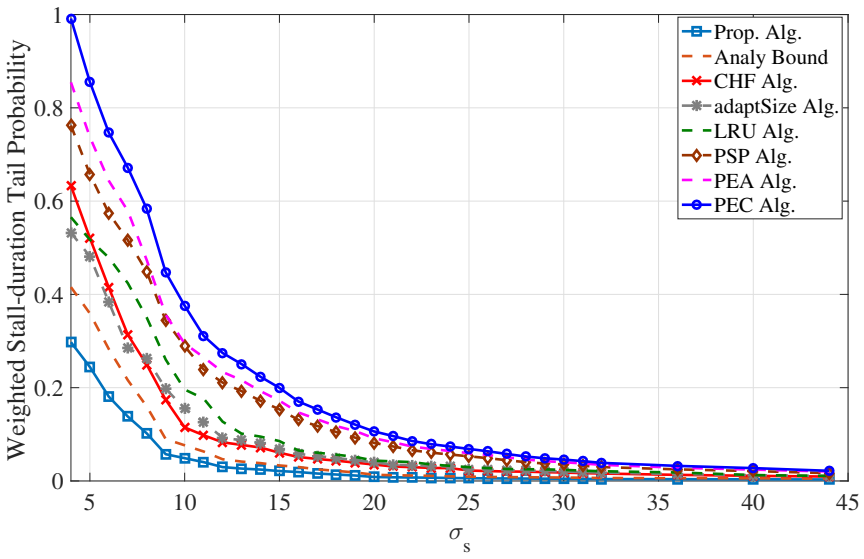

Fig. 5.5.: Weighted SDTP versus σ_s.

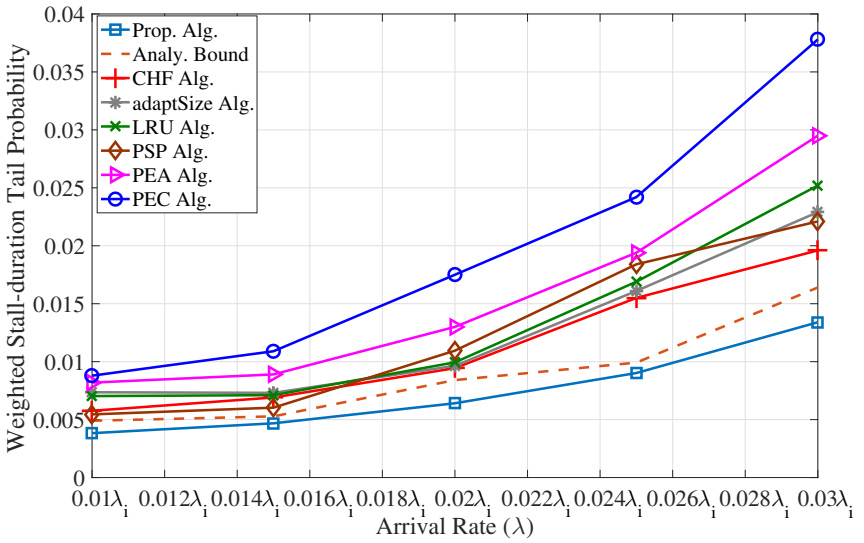

Fig. 5.6.: Weighted stall-duration tail probability versus arrival rate of video files. We vary the arrival rate of the files from $0.01\lambda_i$ to $0.03\lambda_i$ with an increment step of 0.002, where λ_i is the base arrival rate.

Table 5.2.: Testbed Configuration

Cluster Information	
Control Plane	Openstack Kilo
VM flavor	1 VCPU, 2GB RAM, 20G storage (HDD)
Software Configuration	
Operating System	Ubuntu Server 16.04 LTS
Origin Server(s)	Apache Web Server [82]: Apache/2.4.18 (Ubuntu)
Cache Server(s)	Apace Traffic Server [83] 6.2.0 (build # 100621)
Client	Apache JMeter [84] with HLS plugin [85]

units (unless requested again in this window). For the future requests within ω_i or concurrent user requests, the requests for the video chunks are served from the edge-cache, and thus future/concurrent users would experience lower stall duration. If the file can be accessed from the edge router, higher caching level is not used for this request which consequently reduces the traffic at the core backbone servers. If the file cannot be accessed from the edge router, it goes to the distributed cache. We assume some segments, i.e., $L_{j,i}$, of video file i are stored in the distributed cache node j, and are served from the cache nodes. The non-cached segments are served from the data-center. The schematic of our testbed is illustrated in Figure 5.4. Since the two edge-routers are likely in different states, they may not share the cache servers which is the setup we study in the experiments. We note that the theoretical approach proposed earlier is general and can work with shared cache servers across multiple edge routers.

One VM per location is used for generating client workloads. Table 5.2 summarizes a detailed configuration used for the experiments. For client workload, we exploit a popular HTTP-traffic generator, Apache JMeter, with a plug-in that can generate traffic using HTTP Streaming protocol. We assume the amount of available bandwidth between origin server and each cache server is 200 Mbps, 500 Mbps between

cache server 1/2 and edge router 1, and 300 Mbps between cache server 3/4/5 and edge router 2. In this experiment, to allocate bandwidth to the clients, we throttle the client (i.e., JMeter) traffic according to the plan generated by our algorithm. We consider 1000 threads (i.e., users) and set $e_j^{(\ell)} = 40$ for all $\ell = 1, 2$, $d_j = 20$. Segment size τ is set to be equal to 8 seconds. Each edge cache is assumed to have a capacity, equivalent to 15% of the total size of the video files. Further, distributed cache servers can store up to 35% out of the total number of video file segments. The values of α_j and η_j are summarized in Table I.

Video files are generated based on Pareto distribution [62] (as it is a commonly used distribution for file sizes [63]) with shape factor of 2 and scale of 300, respectively. While we stick in the experiment to these parameters, our analysis and results remain applicable for any setting given that the system maintains stable conditions under the chosen parameters. Since we assume that the video file sizes are not heavy-tailed, the first 500 file-sizes that are less than 60 minutes are chosen. When generating video files, the size of each video file is rounded up to the multiple of 8 seconds. For the arrival rates, we use the data from our production system for 500 hot files from two edge routers, and use those arrival rates. The aggregate arrival rates at edge router 1 and router 2 are $\Lambda_1 = 0.01455s^{-1}$, $\Lambda_2 = 0.02155s^{-1}$, respectively.

In order to generate the policy for the implementation, we assume uniform scheduling, $\pi_{i,j} = k/n$, $p_{j,\nu_j} = 1/e_j$, $q_{j,\beta_j} = 1/d_j$. Further, we choose $t_i = 0.01$, $w_{j,\nu_j}^{(e)} = 1/e_j$, $w_{j,\beta_j}^{(\bar{d})} = 1/d_j$ and $w_{j,\beta_j}^{(d)} = 1/d_j$. However, these choices of the initial parameters may not be feasible. Thus, we modify the parameter initialization to be closest norm feasible solutions. Using the initialization, the proposed algorithm is used to obtain the parameters. These parameters are then used to control the bandwidth allocation, distributed cache content placement, the probabilistic scheduling parameters, and the edge caching window sizes. Based on these parameters, the proposed online algorithm is implemented. Since we assume the arrivals of video files are Poisson (and hence inter-arrival time is exponential with λ_i for file i), we generate a sequence of 10000 video file arrivals/requests corresponding to the different files at each edge router.

Upon an arrival of a video file at edge-cache, we apply our proposed online mechanism. For each segment, we used JMeter built-in reports to estimate the downloaded time of each segment and then plug these times into our model to obtain the stall duration which will be used for evaluation of the proposed method.

5.6.2 Baselines

We compare our proposed approach with multiple strategies, which are described as follows.

1. *Projected Equal Server-PSs Scheduling, Optimized Auxiliary variables, Cache Placement, Edge-cache Window-Size, and Bandwidth Wights (PEA):* Starting with the initial solution mentioned above, the problem in (5.34) is optimized over the choice of $\boldsymbol{h}$, $\boldsymbol{w}$, $\boldsymbol{L}$, and $\boldsymbol{\omega}$ (using Algorithms 4, 5, 6, and 7 respectively) using alternating minimization. Thus, the values of $\pi_{i,j}$, p_{i,j,ν_j}, and q_{i,j,β_j} will be approximately close to k/n, $1/e_j$, and $1/d_j$, respectively, for all i, j, ν_j, β_j.

2. *Projected Proportional Service-Rate, Optimized Auxiliary variables, Bandwidth Wights, Edge-cache Window-Size, and Cache Placement (PSP):* In the initialization, the access probabilities among the servers, are given as $\pi_{i,j} = \frac{\mu_j}{\sum_j \mu_j}$, $\forall i, j$. This policy assigns servers proportional to their service rates. The choice of all parameters are then modified to the closest norm feasible solution. Using this initialization, the problem in (5.34) is optimized over the choice of $\boldsymbol{h}$, $\boldsymbol{w}$, $\boldsymbol{L}$, and $\boldsymbol{\omega}$, (using Algorithms 4, 5, 6, and 7, respectively) using alternating minimization.

3. *Projected Equal Caching, Optimized Scheduling Probabilities, Auxiliary variables and Bandwidth Allocation Weights (PEC):* In this strategy, we divide the cache size equally among the video files. Thus, the size of each file in the cache is the same (unless file is smaller than the cache size divided by the number of files). Using this initialization, the problem in (5.34) is optimized over the

choice of $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{w}$, and $\boldsymbol{\omega}$ (using Algorithms 3, 4, 5, and 7, respectively) using alternating minimization.

4. *Caching Hot Files, Optimized Scheduling Probabilities, Auxiliary variables, Edge-cache Window-Size, and Bandwidth Allocation Weights (CHF):* In this strategy, we cache entirely the files that have the largest arrival rates in the storage cache server. Such hot file caching policies have been studied in the literature, see [12] and references therein. Using this initialization, the problem in (5.34) is optimized over the choice of $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{w}$, and $\boldsymbol{\omega}$ (using Algorithms 3, 4, 5, and 7, respectively) using alternating minimization.

5. *Caching based on Least-Recently-Used bases at edge-cache and Caching-Hottest files at storage nodes, Optimized Scheduling Probabilities, Auxiliary variables, Storage Cache Placement, and Bandwidth Allocation Weights (LRU):* In this strategy, a file is entirely cached in the edge-cache servers upon request if space permits; otherwise, the least-recently used file(s) is removed first to evacuate the needed space for the new file. Further, the hottest files are partially cached in the distributed storage cache servers. Such hot file caching policies have been studied in the literature, e.g., [12] and references therein. Using this initialization, the problem in (5.34) is optimized over the choice of $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, and $\boldsymbol{w}$, (using Algorithms 3, 4, and 5, respectively) using alternating minimization.

6. *Caching at edge-cache based on adaptSize policy [53] and Caching-Hottest files at storage nodes, Optimized Scheduling Probabilities, Auxiliary variables, Storage Cache Placement, and Bandwidth Allocation Weights (adaptSize)*: This policy is a probabilistic admission policy in which a video file is admitted into the cache with probability $e^{-size/c}$ so as larger objects are admitted with lower probability and the parameter c is tuned to maximize the object hit rate (OHR), defined as the probability that a requested file is found in the cache. In particular, given a c and an estimate on the arrival rate for the requests for each video file, one can estimate the probability that a given file will be served from the edge-cache.

One can then use these probabilities to compute the OHR as a function of c and then optimize. The value of c is recomputed after a certain number of file requests, using a sliding window approach. We refer the reader to [53] for a more in-depth description.

7. *Caching at edge-cache based on variant of LRU policy [52], Caching-Hottest files at storage nodes, Optimized Scheduling Probabilities, Auxiliary variables, Storage Cache Placement, and Bandwidth Allocation Weights (xLRU)*: We denote by $xLRU$ one of the these policies: $qLRU$, $kLRU$, and kRandom. A $qLRU$ policy is the same as LRU except that files are only added with probability q. In $kLRU$, requested files must traverse $k-1$ additional virtual LRU caches before it is added to the actual cache. kRandom is the same as $kLRU$ except files are evicted from the cache at random. The other optimization parameters are optimized the same way as in the adaptSize policy.

5.6.3 Experimental Results

SDTP performance for different σ: Figure 5.5 shows the decay of weighted SDTP $\sum_{i=1}^{r} \frac{\lambda_i}{\overline{\lambda}_i} \mathbb{P}(\Gamma^{(i)} > \sigma)$ with σ (in seconds) for the considered policies. Notice that SDTP Policy solves the optimal weighted stall tail probability via proposed alternating optimization algorithm. Also, this figure represents the complementary cumulative distribution function (ccdf) of the proposed algorithm as well as the selected baselines. We further observe that uniformly accessing servers and simple service-rate-based scheduling are unable to optimize the request scheduler based on factors like chunk placement, request arrival rates, different stall weights, thus leading to much higher SDTP. Moreover, the figure shows that an entire video file does not have to be present in the edge-cache. That's because when the user requests a cached video, it is served by first sending the portion of the video locally present at edge-cache while obtaining the remainder from the distributed cache servers and/or the origin server, and transparently passing it on to the client. In addition, we see that the analytical (of-

fline) SDTP is very close to the actual (online) SDTP measurement on our testbed. Further, since adaptSize policy does not intelligently incorporate the arrival rates in adding/evicting the video files, it fails to significantly reduce the SDTP. To the best of our knowledge, this is the first work to jointly consider all key design degrees of freedom, including bandwidth allocation among different parallel streams, cache content placement, the request scheduling, window-size of the edge-cache and the modeling variables associated with the SDTP bound.

Arrival Rates Comparisons: Figure 5.6 shows the effect of increasing system workload, obtained by varying the arrival rates of the video files from $0.01s^{-1}$ to $0.03s^{-1}$ with an increment step of $0.002s^{-1}$ on the SDTP. We notice a significant improvement of the QoE metric with the proposed strategy as compared to the baselines. Further, the gap between the analytical offline bound and actual online SDTP is small which validates the tightness of our proposed SDTP bound. Further, while our algorithm optimizes the system parameters offline, this figure shows that an online version of our algorithm can be used to keep track of the systems dynamics and thus achieve an improved performance.

Effect of Number of files: Figure 5.13 shows the impact of varying the number of files from 150 to 550 on the weighted SDTP for the online algorithm. Clearly, weighted SDTP increases with the number of files, which brings in more workload (i.e., higher arrival rates). However, our optimization algorithm optimizes new files along with existing ones to keep overall weighted SDTP at a low level. We note that the proposed optimization strategy effectively reduces the tail probability and outperforms the considered baseline strategies. Thus, joint optimization over all optimization parameters help reduce the tail probability significantly. Also, the gap between online and offline performance is almost negligible which reflects the robustness of our algorithm.

Additional performance evaluation is provided in Appendix 5.7 and Appendix 5.8.

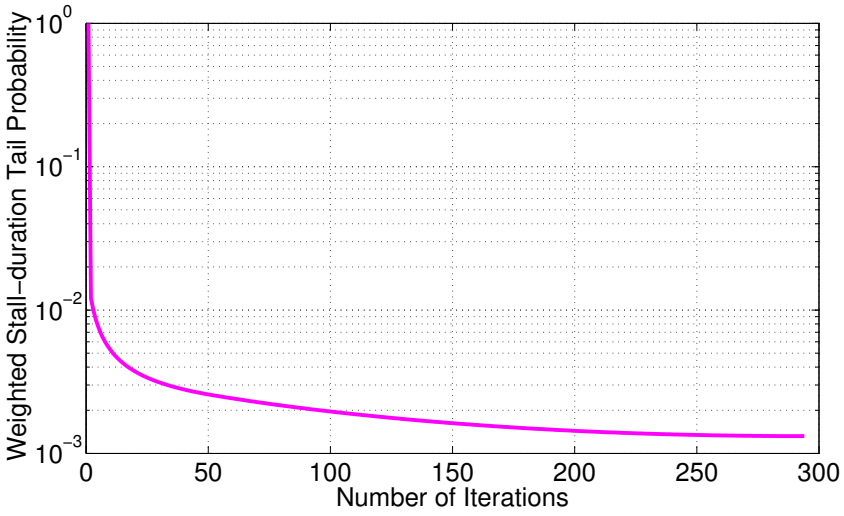

Fig. 5.7.: Convergence of weighted stall-duration tail probability.

5.7 Edge-cache Performance and further Evaluation

Convergence of the proposed algorithm: Figure 5.7 shows the convergence of our proposed SDTP algorithm, which alternatively optimizes the weighted SDTP of all files over scheduling probabilities $\widetilde{\boldsymbol{\pi}}$, auxiliary variables $\boldsymbol{t}$, bandwidth allocation weights $\boldsymbol{w}$, cache server placement $\boldsymbol{L}$, and window-size ω_i. We see that for $r = 500$ video files of size 600s with $m = 5$ cache storage nodes, the weighted stall duration tail probability converges within a few iterations.

Effect of scaling up the bandwidth of the cache servers and datacenter: The effect of increasing the server bandwidth on the weighted SDTP is plotted in Figure 5.8. Intuitively, increasing the storage node bandwidth will increase the service rate of the storage nodes by assigning higher bandwidth to the users, thus, reducing the weighted SDTP.

Effect of the bound percentage ϵ in the SDTP: Figure 5.9 plots the weighted SDTP versus ϵ, i.e., probability that the cache size is exceeded. We see that the SDTP increases significantly with an increase in ϵ. This is because as ϵ increases, there are more edge capacity constraint violations and the the online adaptations may not remain the optimal choice.

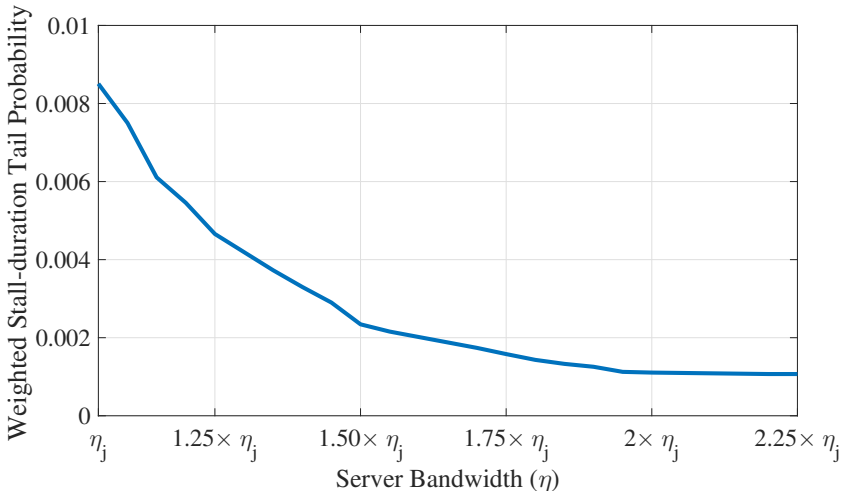

Fig. 5.8.: Weighted SDTP versus the server bandwidth. We vary the server bandwidth from η_j to $2.25\eta_j$ with an increment step of 0.25, where $\eta_j = 20$MBps.

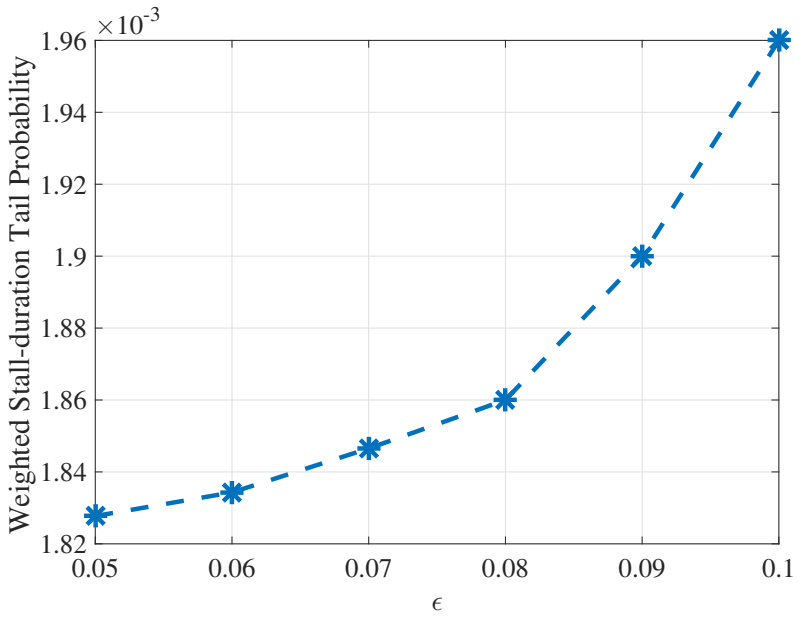

Fig. 5.9.: Weighted SDTP versus the percentage bound on the number of video files (i.e., maximum capacity) in the edge cache ϵ. The percentage of the capacity bound is changed from 0.05 to 0.1 for a cache capacity of $0.20 \times C_{tot}$.

In the following figures, a trace-based implementation is performed, where the video ID, time requests, video lengths, etc. are obtained from one-week traces of a

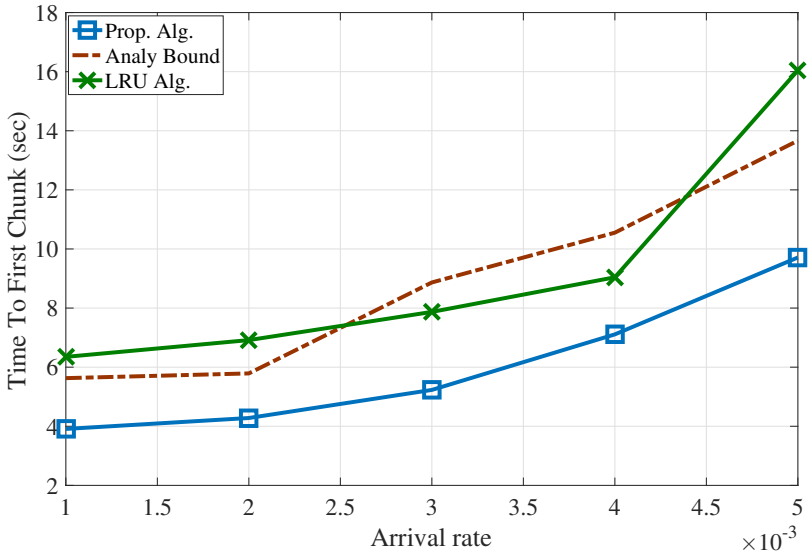

Fig. 5.10.: Time to the first chunk for different arrival rates for 1000 video files.

production system from the major service provider in the US. We note that the arrival process is not Poisson in this case, while the proposed approach still outperform the considered baseline approaches.

Effect of the arrival rates on the TTFC: Figure 5.10 shows the effect of different video arrival rates on the TTFC for different-size video lengths. The different sizes for video files are obtained from real traces of a major video service provider. We compared our proposed online algorithm with the analytical offline bound and LRU-based (explained in Section IV, B) policies. We see that the TTFC increases with arrival rates, as expected, however, since the TTFC is more significant at high arrival rates, we notice a significant improvement in the download time of the first chunk by about 60% at the highest arrival rate in Figure 5.10 as compared to the LRU policy.

Effect of arrival Rates on the MSD: The effect of different video arrival rates on the mean stall duration for different-size video length is captured in Figure 5.11. We compared our proposed online algorithm with five baseline policies and we see that the proposed algorithm outperforms all baseline strategies for the QoE metric of mean stall duration. Thus, bandwidth, size of the time-window, access and placement of

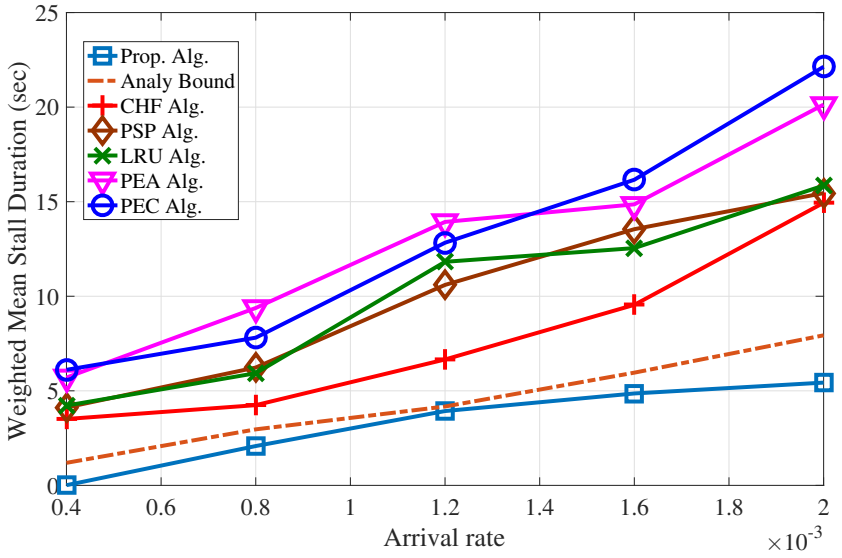

Fig. 5.11.: Mean stall duration versus arrival rates.

files in the storage caches are important for the reduction of mean stall duration. Further, obviously, the mean stall duration increases with arrival rates, as expected. Since the mean stall duration is more significant at high arrival rates, we notice a significant improvement in mean stall duration (approximately 15s to about 5s) at the highest arrival rate in Figure 5.11 as compared to the LRU policy

Effect of edge-cache capacity: We study the miss-rate (percentage of how many video file requests are not served from the edge-cache) performance of the edge-cache. Clearly, the miss-rate decreases with the increasing size of the capacity of the edge-cache. However, when the edge-cache capacity is approximately 35% of the entire video sizes, the miss-rate is around 20%. Further, adaptSize policy does not neither optimize the time to live window of files ω_i's nor intelligently incorporate the arrival rates in adding/evicting the video files, and thus its performance becomes less sensitive to varying the cache size. The variant versions of LRU (qLRU with $q = 0.67$, kLRU and kRandom with $k = 6$) obtain closer performance compared to that of the basic LRU where kLRU performs that best among them as it somehow maintains a

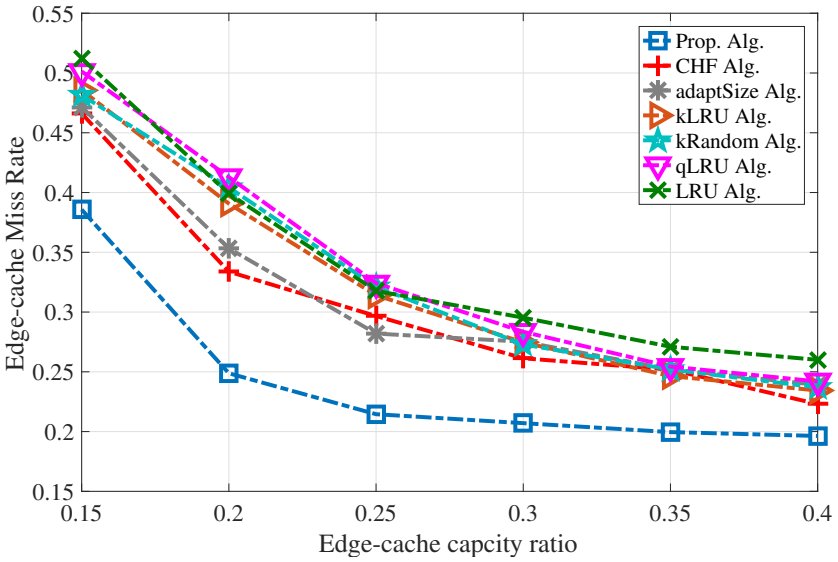

Fig. 5.12.: Edge-cache miss rate versus edge-cache capacity ratio.

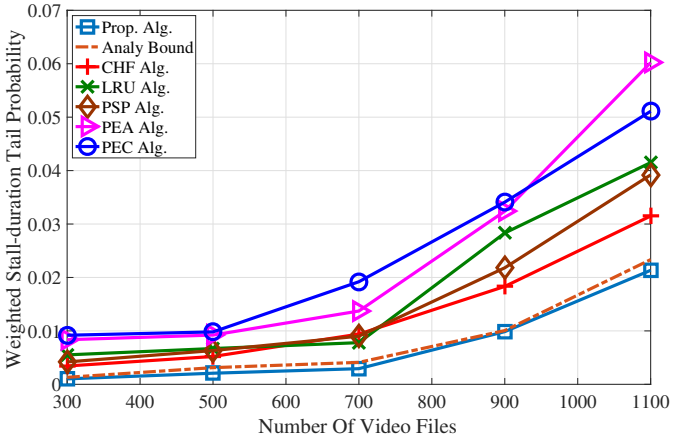

Fig. 5.13.: Comparison of implementation results of our SDTP Algorithm to Analytical SDTP and PEA-based SDTP.

window (k-requests) for admitting a file into the cache and adapts LRU policy in the eviction process.

5.8 Joint Mean-Tail Optimization

We wish to jointly minimize the two QoE metrics (MSD and SDTP) over the choice of server-PSs scheduling, bandwidth allocation, edge-cache window-size and auxiliary variables. Since this is a multi-objective optimization, the objective can be modeled as a convex combination of the two QoE metrics.

The first objective is the minimization of the mean stall duration, averaged over all the file requests, and is given as $\sum_{i,\ell} \frac{\lambda_{i,\ell}}{\overline{\lambda}} \mathbb{E}\left[\Gamma^{(i,\ell)}\right]$. The second objective is the minimization of stall duration tail probability, averaged over all the video file requests, and is given as $\sum_{i,\ell} \frac{\lambda_{i,\ell}}{\overline{\lambda}} \Pr\left(\Gamma^{(i,\ell)} \geq x\right)$. Using the expressions for the mean stall duration and the stall duration tail probability, respectively, optimization of a convex combination of the two QoE metrics can be formulated as follows.

$$\sum_{\ell=1}^{R}\sum_{i=1}^{r}\frac{\lambda_{i,\ell}}{\lambda}\left[\theta \times \Pr(\Gamma^{(i,\ell)} \geq \sigma) + (1-\theta) \times \mathbb{E}\left[\Gamma^{(i,\ell)}\right]\right] \tag{5.45}$$

$$\textbf{s.t.} \tag{5.46}$$

$$(5.35) - (5.44) \tag{5.47}$$

$$g_i < \alpha^{(\overline{d})}_{j,\beta_j,\ell}, g_i < \alpha^{(e)}_{j,\nu_j,\ell}, \ \ \forall i,j,\nu_j,\ell \tag{5.48}$$

$$0 < g_i - \Lambda^{(d)}_{j,\beta_j}(B^{(d)}_{j,\beta_j,\ell}(g_i) - 1), \ \ \forall i,j,\beta_j,\ell \tag{5.49}$$

$$0 < g_i - \Lambda^{(\overline{d})}_{j,\beta_j,\ell}(B^{(\overline{d})}_{j,\beta_j,\ell}(g_i) - 1), \ \ \forall i,j,\beta_j \tag{5.50}$$

$$0 < g_i - \Lambda^{(e)}_{j,\nu_j,\ell}(B^{(e)}_{j,\nu_j,\ell}(g_i) - 1), \ \ \forall i,j,\nu_j,\ell \tag{5.51}$$

$$\text{var} \quad \boldsymbol{\pi}, \boldsymbol{q}, \boldsymbol{p}, \boldsymbol{h}, \boldsymbol{g}, \boldsymbol{w}^{(c)}, \boldsymbol{w}^{(d)}, \boldsymbol{w}^{(e)}, \boldsymbol{L}, \boldsymbol{\omega}.$$

Clearly, the above optimization problem is non-convex in all the parameters jointly. This is can be easily seen in the terms which are product of the different variables. Since the problem is non-convex, we propose an iterative algorithm to solve the problem. This algorithm performs an alternating optimization over the different aforementioned dimensions, such that each sub-problem is shown to have convex constraints and thus can be efficiently solved using NOVA algorithm [57]. The subproblems are explained in detailed in Appendix A.17.

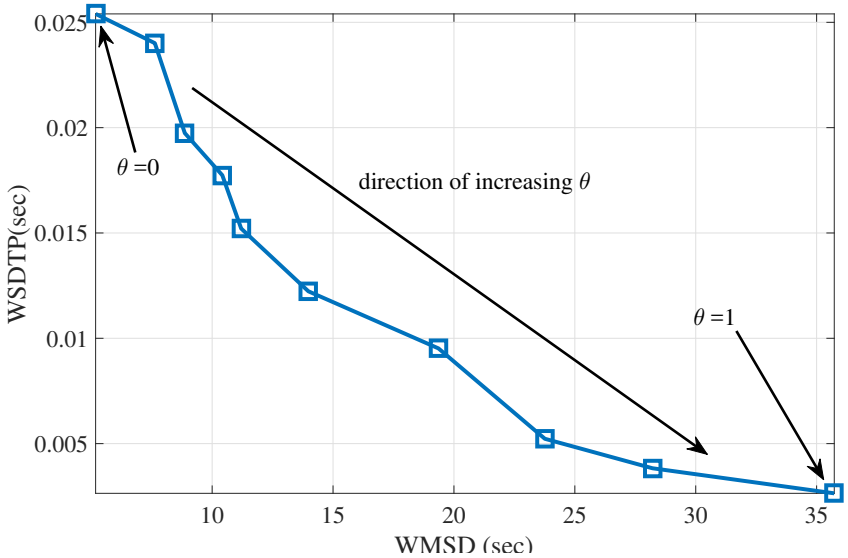

Fig. 5.14.: Tradeoff between weighted mean stall duration and weighted stall-duration tail probability.

Mean-Tail tradeoff: There is a tradeoff between the MSD and SDTP. Hence, we now investigate this tradeoff in order to get a better understanding of how this traddeoff can be compromised. To do so, we vary θ in the above optimization problem to get a tradeoff between MSD and SDTP. Intuitively, if the mean stall duration decreases, the stall duration tail probability also reduces, as depicted in Figure 5.14. Therefore, a question arises whether the optimal point for decreasing the mean stall duration and the stall duration tail probability is the same? Based on our real video traces, we answer this question in negative since we find that at the desgin values that optimize the mean stall duration, the stall duration tail probability is 10+ times higher as compared to the optimal stall duration tail probability. Similarly, the optimal mean stall duration is 7 times lower as compared to the mean stall duration at the design values that optimizes the stall duration tail probability. As a result, an efficient tradeoff point between the two QoE metrics can be chosen based on the point on the curve that is appropriate for the clients.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

This dissertation proposes a CDN-based edge-cache-aided over-the-top multicast video streaming system, where the video content is partially stored on distributed cache servers and access-dependent online edge caching strategy is used at the edge-cache. The content at the distributed cache servers can be either erasure-coded or coded using repetition code.

First, we consider video streaming over cloud where the content is erasure-coded on the distributed servers. Two QoE metrics related to the stall duration, mean stall duration and stall duration tail probability are characterized with upper bounds. Both download and play times of each video segment are characterized to evaluate the QoE metrics. An optimization problem that optimizes the convex combination of the two QoE metrics for the choice of placement and access of contents from the servers is formulated.

Next, we consider video quality as a QoE metric in our optimization problem. Besides stall measures (mean and tail probability of stall), average quality of the streamed video is optimized. A two-stage probabilistic scheduling is proposed for the choice of servers and the parallel streams between the server and the edge router. Using the two-stage probabilistic scheduling and probabilistic quality assignment for the videos, an upper bound on the mean stall duration is derived. An optimization problem that minimizes a convex combination of the two QoE metrics is formulated, over the choice of two-stage probabilistic scheduling, probabilistic quality assignment, bandwidth allocation, and auxiliary variables.

Third, we consider multi-stage caching system where content can be obtained from edge-cache, distributed storage servers and/or a central node. Using this model,

weighted stall duration tail probability is optimized by considering two-stage probabilistic scheduling for the choice of servers and the parallel streams between the server and the edge router. Using the two-stage probabilistic scheduling and the edge caching mechanism, an upper bound on the stall duration tail probability is characterized. Further, an optimization problem that minimizes the weighted stall duration tail probability is formulated, over the choice of two-stage probabilistic scheduling, bandwidth allocation, cache placement, edge-cache parameters, and the auxiliary variables in the bound.

For all different scenarios, an efficient algorithm is proposed to solve the optimization problem and the experimental results on a virtualized cloud system managed by Openstack depict the improved performance of our proposed algorithm as compared to the state-of-the-other algorithm and some competitive baselines.

6.2 Future Work

A server does not need to serve different video requests one after the other. It may be better to serve video segments out of order from a queue thus helping stall durations since the later video requests do not have to wait for finishing chunks of earlier requests which have later deadlines. Exploiting this flexibility is an open problem.

Possible extensions to accommodate multiple quality levels and different chunk sizes are discussed in Appendix A.21. However, a complete treatment of adaptive bit-rate video streaming is left as a future work.

We note that the current video streaming algorithms use adaptive bit-rate (ABR) strategies to change the video qualities of segments within a video [54, 55]. One of the strategies look at the buffer usage at the client to determine the quality of the next segment [54]. Incorporating efficient ABR streaming algorithms is an interesting future work. The main challenge in this extension is to incorporate the client behavior which makes the arrival process non-memoryless thus making the analysis complex.

Finally, considering the decoding time by combining data in the calculations is left as a future work.

REFERENCES

REFERENCES

[1] Marketsandmarkets, "Solution, by service, by platform, by user type, by deployment type, by revenue model, by industry, and by region - global forecast to 2021," *http://www.marketsandmarkets.com/Market-Reports/video-streaming-market-181135120.html*, May 2016.

[2] D. Mowrey, "Cloud video trends to watch in 2017," *http://www.multichannel.com/blog/mcn-guest-blog/cloud-video-trends-watch-2017/409903*, Jan 2017.

[3] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. Springer-Verlag, 2002.

[4] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *Information Theory, IEEE Transactions on*, vol. 56, no. 9, pp. 4539–4551, Sept 2010.

[5] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," in *Proceedings of the 39th international conference on Very Large Data Bases.*, 2013.

[6] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12. USENIX Association, 2012.

[7] A. Fikes, "Storage architecture and challenges (talk at the google faculty summit)," http://bit.ly/nUylRW, Tech. Rep., 2010.

[8] I. Sandvine, "Global internet phenomena report," *North America and Latin America*, 2016.

[9] Y. Shen, "The shift to content delivery networks (cdns) supports more and better customer video experiences," 2014.

[10] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache content-selection policies for streaming video services," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[11] J. Dean and L. A. Barroso, "The tail at scale," in *Communications of the ACM*, 2013.

[12] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, 2002.

[13] Y. L. M. Bramson and B. Prabhakar, "Randomized load balancing with general service time distributions," in *Proceedings of ACM Sigmetrics*, 2010.

[14] G. K. A. G. J. L. Y. Lu, Q. Xie and A. Greenberg, "Joinidle-queue: A novel load balancing algorithm for dynamically scalable web services," in *29th IFIP-PERFORMANCE*, 2010.

[15] P. A. R. J. M. K. Aquilera, HP Labs. and L. Xu, "Using erasure codes efficiently for storage in a distributed system," in *Proceedings of Dependable Systems and Networks (DSN 2005)*, 2005.

[16] J. Li, "Adaptive erasure resilient coding in distributed storage," in *Multimedia and Expo, 2006 IEEE International Conference*, 2006.

[17] A. Fallahi and E. Hossain, "Distributed and energy-aware mac for differentiated services wireless packet networks: a general queuing analytical framework," *IEEE Transactions on Mobile Computing*, 2007.

[18] N. Taylor and Z. Ives, "Using erasure codes efficiently for storage in a distributed system," in *IEEE ICED Conference*, 2010.

[19] E. Ziouva and T. Antoankopoulos, "Csma/ca performance under high traffic conditions: throughput and delay analysis," *Computer Comm*, vol. 25, pp. 313–321, 2002.

[20] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proceedings of the First IPTPS*, 2002.

[21] R. G. C. Angllano and M. Grangetto, "Exploiting rateless codes in cloud storage systems," *IEEE Transactions on Parallel and Distributed Systems*, 2014.

[22] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 2, pp. 3–14, Sep. 2014. [Online]. Available: http://doi.acm.org/10.1145/2667522.2667524

[23] ——, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2443–2457, 2016.

[24] S. Chen, Y. Sun, U. Kozat, L. Huang, P. Sinha, G. Liang, X. Liu, and N. Shroff, "When queuing meets coding: Optimal-latency data retrieving scheme in storage clouds," in *Proceedings of IEEE Infocom*, 2014.

[25] Y. L. G. Joshi and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," arXiv:1305.3945v1, Tech. Rep., 2013.

[26] T. M. queue: analyzing latency performance of codes and redundant requests, "Tahoe-lafs docs," arXiv:1211.5405, Tech. Rep., 2012.

[27] H. Z. L. Huang, S. Pawar and K. Ramchandran, "Codes can reduce queueing delay in data centers," *Journals CORR*, vol. 1202.1359, 2012.

[28] F. Baccelli, A. Makowski, and A. Shwartz, "The fork-join queue and related systems with synchronization constraints: stochastic ordering and computable bounds," *Advances in Applied Probability*, pp. 629–660, 1989.

[29] V. Aggarwal, J. Fan, and T. Lan, "Taming tail latency for erasure-coded, distributed storage systems," in *Proc. IEEE Infocom*, Jul 2017.

[30] K. Lee, L. Yan, A. Parekh, and K. Ramchandran, "A vod system for massively scaled, heterogeneous environments: Design and implementation," in *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2013, pp. 1–10.

[31] Z. Huang, C. Mei, L. E. Li, and T. Woo, "Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 201–205.

[32] J. He, Y. Wen, J. Huang, and D. Wu, "On the cost–qoe tradeoff for cloud-based video streaming under amazon ec2's pricing models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 669–680, 2014.

[33] H.-Y. Chang, K.-B. Chen, and H.-C. Lu, "A novel resource allocation mechanism for live cloud-based video streaming service," *Multimedia Tools and Applications*, pp. 1–18, 2016.

[34] N. Oza and N. Gohil, "Implementation of cloud based live streaming for surveillance," in *Communication and Signal Processing (ICCSP), 2016 International Conference on*. IEEE, 2016, pp. 0996–0998.

[35] M. Chen, "Amvsc: a framework of adaptive mobile video streaming in the cloud," in *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012, pp. 2042–2047.

[36] X. Wang, M. Chen, T. T. Kwon, L. Yang, and V. C. Leung, "Ames-cloud: a framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 811–820, 2013.

[37] H. Hu, Y. Wen, T. S. Chua, J. Huang, W. Zhu, and X. Li, "Joint Content Replication and Request Routing for Social Video Distribution Over Cloud CDN: A Community Clustering Method," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 7, pp. 1320–1333, July 2016.

[38] W. Hu, Z. Wang, M. Ma, and L.-F. Sun, "Edge video cdn: A wi-fi content hotspot solution," *Journal of Computer Science and Technology*, vol. 31, no. 6, pp. 1072–1086, Nov 2016.

[39] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 339–350.

[40] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan, "Prometheus: toward quality-of-experience estimation for mobile apps from passive network measurements," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014, p. 18.

[41] A. Balachandran, V. Aggarwal, E. Halepovic, J. Pang, S. Seshan, S. Venkataraman, and H. Yan, "Modeling web quality-of-experience on cellular networks," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 213–224.

[42] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1076–1089, 2017.

[43] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky," in *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 2009, pp. 25–34.

[44] Z. Liu, Z. Guo, D. Gupta, X. Tang, J. Wang, W. Ye, and Y. Zhao, "Apparatus and method of video streaming," Oct. 4 2018, uS Patent App. 15/475,682.

[45] M. Garcia-Pineda, S. Felici-Castell, and J. Segura-Garcia, "Using factor analysis techniques to find out objective video quality metrics for live video streaming over cloud mobile media services," *Network Protocols and Algorithms*, vol. 8, no. 1, 2016.

[46] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich, "Queueing system with selection of the shortest of two queues: An asymptotic approach," *Problemy Peredachi Informatsii*, vol. 32, no. 1, pp. 20–34, 1996.

[47] T. Hellemans and B. Van Houdt, "On the power-of-d-choices with least loaded server selection," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 2, p. 27, 2018.

[48] L. Ying, R. Srikant, and X. Kang, "The power of slightly more than one sample in randomized load balancing," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 1131–1139.

[49] P. Flajolet, L. Thimonier, and D. Gardy, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," Ph.D. dissertation, INRIA, 1987.

[50] R. Fagin, "Asymptotic miss ratios over independent references," *Journal of Computer and System Sciences*, vol. 14, no. 2, pp. 222–250, 1977.

[51] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. IEEE, 2012, pp. 310–315.

[52] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 1, no. 3, p. 12, 2016.

[53] D. S. Berger, R. K. Sitaraman, and M. Harchol-Balter, "Adaptsize: Orchestrating the hot object memory cache in a content delivery network." in *NSDI*, 2017, pp. 483–498.

[54] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

[55] A. Elgabli, V. Aggarwal, F. Qian, S. Hao, and S. Sen, "Robust svc video streaming in cellular environments," *Submitted to IEEE/ACM Transactions on Networking*, May 2017.

[56] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Bobtail: Avoiding long tails in the cloud," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2013)*, 2013.

[57] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization?part i: Theory," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1929–1944, 2017.

[58] P. J. Denning and T. G. Lewis, "Exponential laws of computing growth," *Commun. ACM*, vol. 60, no. 1, pp. 54–65, Dec. 2016. [Online]. Available: http://doi.acm.org/10.1145/2976758

[59] A. Zwart and O. J. Boxma, "Sojourn time asymptotics in the m/g/1 processor sharing queue," *Queueing systems*, vol. 35, no. 1-4, pp. 141–166, 2000.

[60] M. Kuczma, *An introduction to the theory of functional equations and inequalities: Cauchy's equation and Jensen's inequality*. Springer Science & Business Media, 2009.

[61] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster," in *Presented as part of the 5th {USENIX} Workshop on Hot Topics in Storage and File Systems*, 2013.

[62] B. C. Arnold, *Pareto distribution*. Wiley Online Library, 2015.

[63] V. Ramaswami, K. Jain, R. Jana, and V. Aggarwal, "Modeling heavy tails in traffic sources for network performance evaluation," in *Computational Intelligence, Cyber Security and Computational Models*, ser. Advances in Intelligent Systems and Computing. Springer India, 2014, vol. 246, pp. 23–44.

[64] "Four reasons we choose amazon's cloud as our computing platform," *Netflix "Tech" Blog, December*, 2010.

[65] V. Aggarwal, V. Gopalakrishnan, R. Jana, K. Ramakrishnan, and V. Vaishampayan, "Optimizing cloud resources for delivering iptv services through virtualization," *Multimedia, IEEE Transactions on*, vol. 15, no. 4, pp. 789–801, 2013.

[66] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "Mp-dash: Adaptive video streaming over preference-aware multipath," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. ACM, 2016, pp. 129–143.

[67] A. Alabbasi and V. Aggarwal, "Joint information freshness and completion time optimization for vehicular networks," *arXiv preprint arXiv:1811.12924*, 2018.

[68] C. Kreuzberger, D. Posch, and H. Hellwagner, "A scalable video coding dataset and toolchain for dynamic adaptive streaming over http," in *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, 2015, pp. 213–218.

[69] "Sandvine Intelligent Networks: Global Internet Phenomena Report: http:www.sandvine.com."

[70] X. Li, X. Wang, K. Li, Z. Han, and V. C. Leung, "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Transactions on Wireless Communications*, vol. 16, no. 10, pp. 6926–6939, 2017.

[71] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, and M. Gerla, "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with qoe support," *Sensors*, vol. 18, no. 2, p. 329, 2018.

[72] F. Perez-Sorrosal, M. Patiño-Martinez, R. Jimenez-Peris, and B. Kemme, "Elastic si-cache: consistent and scalable caching in multi-tier architectures," *The VLDB Journal-The International Journal on Very Large Data Bases*, vol. 20, no. 6, pp. 841–865, 2011.

[73] H. Khedher, E. Abd-Elrahman, H. Afifi, and M. Marot, "Optimal and cost efficient algorithm for virtual cdn orchestration," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, 2017, pp. 61–69.

[74] P. A. Frangoudis, L. Yala, and A. Ksentini, "Cdn-as-a-service provision over a telecom operator's cloud," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 702–716, 2017.

[75] G. Liang and U. Kozat, "Fast cloud:pushing the envelope on delay performance of cloud storage with coding," *IEEE/ACM Transactions on Networking*, pp. 124–137, 2013.

[76] E. Friedlander and V. Aggarwal, "Generalization of lru cache replacement policy with applications to video streaming," *arXiv preprint arXiv:1806.10853*, 2018.

[77] "OpenStack: Open source software for creating private and public clouds," http://www.openstack.org/.

[78] V. Aggarwal, Y.-F. R. Chen, T. Lan, and Y. Xiang, "Sprout: A functional caching approach to minimize service latency in erasure-coded storage," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3683–3694, 2017.

[79] V. Aggarwal, R. Caldebank, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and F. Yu, "The effectiveness of intelligent scheduling for multicast video-on-demand," in *Proceedings of the 17th ACM International Conference on Multimedia*, ser. MM '09, 2009, pp. 421–430. [Online]. Available: http://doi.acm.org/10.1145/1631272.1631330

[80] M. Schwartz, *Telecommunication networks: protocols, modeling and analysis*. Addison-Wesley Reading, MA, 1987, vol. 7.

[81] A. O. Al-Abbasi and V. Aggarwal, "Video streaming in distributed erasure-coded storage systems: Stall duration analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1921–1932, 2018.

[82] "Apache HTTP Server Project," https://httpd.apache.org.

[83] "Apache Traffic Server," http://trafficserver.apache.org.

[84] "Apache JMeter," http://jmeter.apache.org/.

[85] "HLS plug-in for Apache JMeter," https://jmeter-plugins.org.

[86] "Nonlinear bipartite matching," *Discrete Optimization*, vol. 5, no. 1, pp. 53 – 65, 2008.

[87] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache content-selection policies for streaming video services," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[88] A. O. Al-Abbasi and V. Aggarwal, "Optimized video streaming over cloud: A stall-quality trade-off," *CoRR*, 2018. [Online]. Available: https://arxiv.org/abs/1806.09466

APPENDICES

A. APPENDIX

A.1 Proof of Lemma 1

$$\begin{aligned}
\overline{R}_j(s) &= \sum_{i=1}^{r} \frac{\pi_{ij}\lambda_i}{\Lambda_j} \mathbb{E}\left[e^{-s(ST_{i,j})}\right] \\
&= \sum_{i=1}^{r} \frac{\pi_{ij}\lambda_i}{\Lambda_j} \mathbb{E}\left[e^{-s\left(\sum_{\nu=1}^{L_i} Y_j^{(\nu)}\right)}\right] \\
&= \sum_{i=1}^{r} \frac{\pi_{ij}\lambda_i}{\Lambda_j} \left(\mathbb{E}\left[e^{-s\left(Y_j^{(1)}\right)}\right]\right)^{L_i} \\
&= \sum_{i=1}^{r} \frac{\pi_{ij}\lambda_i}{\Lambda_j} \left(\frac{\alpha_j e^{-\beta_j s}}{\alpha_j + s}\right)^{L_i}
\end{aligned} \tag{A.1}$$

A.2 Proof of Lemma 2

This follows by substituting $t = -s$ in (4.15) and $B_j(t)$ is given by (4.12) and $M_j(t)$ is given by (4.5). This expressions holds when $t - \Lambda_j\left(B_j(t) - 1\right) > 0$ and $t < 0 \, \forall j$, since the moment generating function does not exist if the above does not hold.

A.3 Proof of Lemma 3

$$
\begin{aligned}
& H_{ij} \\
= & \sum_{\ell=1}^{L_i} \left(\frac{e^{-t_i(d_s+(\ell-1)\tau)} \left(1-\rho_j\right) t_i}{t_i - \Lambda_j \left(B_j(t_i)-1\right)} \left(\frac{\alpha_j e^{t_i \beta_j}}{\alpha_j - t_i} \right)^{\ell} \right) \\
= & \frac{e^{-t_i d_s} \left(1-\rho_j\right) t_i}{t_i - \Lambda_j \left(B_j(t_i)-1\right)} \sum_{\ell=1}^{L_i} \left(e^{-t_i(\ell-1)\tau} \left(\frac{\alpha_j e^{t_i \beta_j}}{\alpha_j - t_i} \right)^{\ell} \right) \\
= & \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_j\right) t_i}{t_i - \Lambda_j \left(B_j(t_i)-1\right)} \sum_{\ell=1}^{L_i} \left(e^{-t_i \tau} \frac{\alpha_j e^{t_i \beta_j}}{\alpha_j - t_i} \right)^{\ell} \\
= & \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_j\right) t_i}{t_i - \Lambda_j \left(B_j(t_i)-1\right)} \sum_{\ell=1}^{L_i} \left(\frac{\alpha_j e^{t_i \beta_j - t_i \tau}}{\alpha_j - t_i} \right)^{\ell} \\
= & \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_j\right) t_i}{t_i - \Lambda_j \left(B_j(t_i)-1\right)} \times \\
& \left(M_j(t_i) e^{-t_i \tau} \frac{1 - \left(M_j(t_i)\right)^{Li} e^{-t_i L_i \tau}}{1 - M_j(t_i) e^{-t_i \tau}} \right) \\
= & \frac{e^{-t_i(d_s-\tau)} \left(1-\rho_j\right) t_i \widetilde{M}_j(t_i)}{t_i - \Lambda_j \left(B_j(t_i)-1\right)} \frac{1 - \left(\widetilde{M}_j(t_i)\right)^{L_i}}{\left(1 - \widetilde{M}_j(t_i)\right)}
\end{aligned}
\tag{A.2}
$$

A.4 Proof of Theorem A.19.1

We first find an upper bound on F_{ij} as follows.

$$\begin{aligned}
F_{ij} &= \mathbb{E}\left[\max_z e^{t_i p i j z}\right] \\
&\overset{(d)}{\leq} \sum_z \mathbb{E}\left[e^{t_i p i j z}\right] \\
&\overset{(e)}{=} e^{t_i(d_s+(L_i-1)\tau)} + \\
&\sum_{z=2}^{L_i+1} \frac{e^{t_i(L_i-z+1)\tau}\left(1-\rho_j\right)t_i}{t_i-\Lambda_j\left(B_j(t_i)-1\right)}\left(\frac{\alpha_j e^{t_i\beta_j}}{\alpha_j-t_i}\right)^{z-1} \\
&\overset{(f)}{=} e^{t_i(d_s+(L_i-1)\tau)} + \\
&\sum_{\ell=1}^{L_i} \frac{e^{t_i(L_i-\ell)\tau}\left(1-\rho_j\right)t_i}{t_i-\Lambda_j\left(B_j(t_i)-1\right)}\left(\frac{\alpha_j e^{t_i\beta_j}}{\alpha_j-t_i}\right)^{\ell}
\end{aligned} \tag{A.3}$$

where (d) follows by bounding the maximum by the sum, (e) follows from (4.22), and (f) follows by substituting $\ell = z-1$.

Further, substituting the bounds (A.3) and (4.35) in (A.72), the mean stall duration is bounded as follows.

$$\begin{aligned}
& \mathbb{E}\left[\Gamma^{(i)}\right] \\
\leq\ & \frac{1}{t_i}\log\left(\sum_{j=1}^{m}\pi_{ij}\left(e^{t_i(d_s+(L_i-1)\tau)}\right.\right. \\
& \left.\left.+\sum_{\ell=1}^{L_i} e^{t_i(L_i-\ell)\tau} Z_{i,j}^{(\ell)}(t_i)\right)\right) - \left(d_s+(L_i-1)\tau\right) \\
=\ & \frac{1}{t_i}\log\left(\sum_{j=1}^{m}\pi_{ij}\left(e^{t_i(d_s+(L_i-1)\tau)}\right.\right. \\
& \left.\left.+\sum_{\ell=1}^{L_i} e^{t_i(L_i-\ell)\tau} Z_{i,j}^{(\ell)}(t_i)\right)\right) - \frac{1}{t_i}\log\left(e^{t_i(d_s+(L_i-1)\tau)}\right) \\
=\ & \frac{1}{t_i}\log\left(\sum_{j=1}^{m}\pi_{ij}\left(1+\sum_{\ell=1}^{L_i} e^{-t_i(d_s+(\ell-1)\tau)} Z_{i,j}^{(\ell)}(t_i)\right)\right)
\end{aligned} \tag{A.4}$$

A.5 Proof of Theorem 3.6.1

Substituting (3.29) in (3.27), we get

$$\begin{aligned}
& \Pr\left(T_i^{(L_i)} \geq \overline{x}\right) \\
\leq & \sum_j \pi_{ij} \mathbb{P}\left(\max_z p_{ijz} \geq \overline{x}\right) \\
\leq & \sum_j \pi_{ij} \frac{F_{ij}}{e^{t_i \overline{x}}} \\
\overset{(g)}{\leq} & \sum_j \frac{\pi_{ij}}{e^{t_i \overline{x}}} \left(e^{t_i(d_s+(L_i-1)\tau)} + H_{ij}\right) \\
= & \sum_j \frac{\pi_{ij}}{e^{t_i(x+d_s+(L_i-1)\tau)}} \left(e^{t_i(d_s+(L_i-1)\tau)} + H_{ij}\right) \\
= & \sum_j \frac{\pi_{ij}}{e^{t_i x}} \left(1 + e^{-t_i(d_s+(L_i-1)\tau)} H_{ij}\right) \qquad \text{(A.5)}
\end{aligned}$$

where (g) follows from (A.3) and H_{ij} is given by (4.42).

A.6 Description of the Algorithms for the Three Sub-Problems

A.6.1 Access Optimization

Given the placement and the auxiliary variables, this subproblem can be written as follows.

Input: $\boldsymbol{t}$, $\boldsymbol{S}$

Objective: min (4.45)

s.t. (4.48), (4.49), (4.51), (4.51), (3.49), (4.58)

var. $\boldsymbol{\pi}$

In order to solve this problem, we have used iNner cOnVex Approximation (NOVA) algorithm proposed in [57] to solve this sub-problem. The key idea for this algorithm is that the non-convex objective function is replaced by suitable convex approximations at which convergence to a stationary solution of the original non-convex optimization

is established. NOVA solves the approximated function efficiently and maintains feasibility in each iteration. The objective function can be approximated by a convex one (e.g., proximal gradient-like approximation) such that the first order properties are preserved [57], and this convex approximation can be used in NOVA algorithm.

Let $\widetilde{U}(\boldsymbol{\pi};\boldsymbol{\pi^{\nu}})$ be the convex approximation at iterate $\boldsymbol{\pi^{\nu}}$ to the original non-convex problem $U(\boldsymbol{\pi})$, where $U(\boldsymbol{\pi})$ is given by (4.45). Then, a valid choice of $\widetilde{U}(\boldsymbol{\pi};\boldsymbol{\pi^{\nu}})$ is the first order approximation of $U(\boldsymbol{\pi})$, e.g., (proximal) gradient-like approximation, i.e.,

$$\widetilde{U}(\boldsymbol{\pi},\boldsymbol{\pi^{\nu}}) = \nabla_{\boldsymbol{\pi}} U(\boldsymbol{\pi^{\nu}})^T (\boldsymbol{\pi}-\boldsymbol{\pi^{\nu}}) + \frac{\tau_u}{2}\|\boldsymbol{\pi}-\boldsymbol{\pi^{\nu}}\|^2, \tag{A.6}$$

where τ_u is a regularization parameter. Note that all the constraints (4.48), (4.49), (4.51), (4.51), (3.49), and (4.58) are linear in $\boldsymbol{\pi_{i,j}}$. The NOVA Algorithm for optimizing $\boldsymbol{\pi}$ is described in Algorithm 3. Using the convex approximation $\widetilde{U}(\boldsymbol{\pi};\boldsymbol{\pi^{\nu}})$, the minimization steps in Algorithm 3 are convex, with linear constraints and thus can be solved using a projected gradient descent algorithm. A step-size (γ) is also used in the update of the iterate $\boldsymbol{\pi}^{\nu}$. Note that the iterates $\left\{\boldsymbol{\pi}^{(\nu)}\right\}$ generated by the algorithm are all feasible for the original problem and, further, convergence is guaranteed, as shown in [57] and described in the following lemma.

Lemma 18 *For fixed placement $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{t}$, the optimization of our problem over $\boldsymbol{\pi}$ generates a sequence of decreasing objective values and therefore is guaranteed to converge to a stationary point.*

A.6.2 Auxiliary Variables Optimization

Given the placement and the access variables, this subproblem can be written as follows.

Input: $\boldsymbol{\pi}$, $\boldsymbol{S}$

Objective: min (4.45)

s.t. (3.45), (4.56), (3.47),(4.57), (3.49), (4.58),

Algorithm 1: NOVA Algorithm to solve Access Optimization sub-problem

1. **Initialize** $\nu = 0$, $k = 0$,$\gamma^{\nu} \in (0, 1]$, $\epsilon > 0$,$\boldsymbol{\pi}^0$ such that $\boldsymbol{\pi}^0$ is feasible ,
2. **while** $\text{obj}\,(k) - \text{obj}\,(k-1) \geq \epsilon$
3. *//Solve for $\boldsymbol{\pi}^{\nu+1}$ with given $\boldsymbol{\pi}^{\nu}$*
4. **Step 1**: Compute $\widehat{\boldsymbol{\pi}}\,(\boldsymbol{\pi}^{\nu})$, the solution of $\widehat{\boldsymbol{\pi}}\,(\boldsymbol{\pi}^{\nu}) = \underset{\boldsymbol{\pi}}{\text{argmin}}\, \widetilde{\boldsymbol{U}}\,(\boldsymbol{\pi}, \boldsymbol{\pi}^{\nu})$ s.t. (4.48), (4.49), (4.51), (4.51), (3.45), (3.49), solved using projected gradient descent
5. **Step 2**: $\boldsymbol{\pi}^{\nu+1} = \boldsymbol{\pi}^{\nu} + \gamma^{\nu}\,(\widehat{\boldsymbol{\pi}}\,(\boldsymbol{\pi}^{\nu}) - \boldsymbol{\pi}^{\nu})$.
6. *//update index*
7. **Set** $\nu \leftarrow \nu + 1$
8. **end while**
9. **output:** $\widehat{\boldsymbol{\pi}}\,(\boldsymbol{\pi}^{\nu})$

var. $\boldsymbol{t}$

Similar to Access Optimization, this optimization can be solved using NOVA algorithm. The constraints (3.45) and (4.56) are linear in $\boldsymbol{t}$. The next two Lemmas show that the constraints (3.47), (4.57), (3.49), and (4.58) are convex in $\boldsymbol{t}$ respectively.

Lemma 19 *The constraints* (3.47) *and* (4.57) *are convex with respect to* $\boldsymbol{t}$.

Proof The proof is provided in Appendix A.7. ■

Lemma 20 *The constraints* (3.49) *and* (4.58) *are convex with respect to* $\boldsymbol{t}$.

Proof The proof is provided in Appendix A.8. ■

Algorithm 4 shows the used procedure to solve for $\boldsymbol{t}$. Let $\overline{U}\,(\boldsymbol{t}; \boldsymbol{t}^{\boldsymbol{\nu}})$ be the convex approximation at iterate $\boldsymbol{t}^{\boldsymbol{\nu}}$ to the original non-convex problem $U\,(\boldsymbol{t})$, where $U\,(\boldsymbol{t})$ is

Algorithm 2: NOVA Algorithm to solve Auxiliary Variables Optimization sub-problem

1. **Initialize** $\nu = 0$,$\gamma^{\nu} \in (0, 1]$, $\epsilon > 0$, $\boldsymbol{t}^0$ such that $\boldsymbol{t}^0$ is feasible,
2. **while** $\text{obj}(\nu) - \text{obj}(\nu - 1) \geq \epsilon$
3. *//Solve for $\boldsymbol{t}^{\nu+1}$ with given $\boldsymbol{t}^{\nu}$*
4. **Step 1**: Compute $\widehat{\boldsymbol{t}}(\boldsymbol{t}^{\nu})$, the solution of $\widehat{\boldsymbol{t}}(\boldsymbol{t}^{\nu}) = \underset{\boldsymbol{t}}{\text{argmin}}\ \overline{\boldsymbol{U}}(\boldsymbol{t}, \boldsymbol{t}^{\nu})$, s.t. (3.45), (4.57), and (3.49) using projected gradient descent
5. **Step 2**: $\boldsymbol{t}^{\nu+1} = \boldsymbol{t}^{\nu} + \gamma^{\nu}\left(\widehat{\boldsymbol{t}}(\boldsymbol{t}^{\nu}) - \boldsymbol{t}^{\nu}\right)$.
6. *//update index*
7. **Set** $\nu \leftarrow \nu + 1$
8. **end while**
9. **output:** $\widehat{\boldsymbol{t}}(\boldsymbol{t}^{\nu})$

given by (4.45), assuming other parameters constant. Then, a valid choice of $\overline{U}(\boldsymbol{t}; \boldsymbol{t^{\nu}})$ is the first order approximation of $U(\boldsymbol{t})$, i.e.,

$$\overline{U}(\boldsymbol{t}, \boldsymbol{t^{\nu}}) = \nabla_{\boldsymbol{t}} U(\boldsymbol{t^{\nu}})^T (\boldsymbol{t} - \boldsymbol{t^{\nu}}) + \frac{\tau_t}{2} \|\boldsymbol{t} - \boldsymbol{t^{\nu}}\|^2 . \tag{A.7}$$

where τ_t is a regularization parameter. The detailed steps can be seen in Algorithm 4. Since all the constraints (3.45), (4.57),and (3.49) have been shown to be convex in $\boldsymbol{t}$, the optimization problem in Step 1 of Algorithm 4 can be solved by the standard projected gradient descent algorithm.

A.6.3 Placement Optimization

Given $\boldsymbol{\pi}$ and $\boldsymbol{t}$, this subproblem finds a permutation of the placement of files on the different servers. Let the given $\boldsymbol{\pi}$ be denoted as $\boldsymbol{\pi}' = \{\pi'_{ij} \forall i, j\}$ and the placement corresponding to this access be $\boldsymbol{\mathcal{S}}' = (\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_r)$. We find a permutation of

the servers m for each file i, and call it $\zeta_i(j)$ is a permutation of the servers from $j \in \{1, \cdots m\}$ to $\zeta_i(j) \in \{1, \cdots m\}$. Further, having the mapping of the servers for each file, the new access probabilities are $\pi_{ij} = \pi'_{i,\zeta_i(j)}$. Having these access probabilities, the new placement of the files will be $\mathcal{S}_i = \{\zeta_i(j) \forall j \in \mathcal{S}'_i\}$. We note that the constraints (4.51), (4.51), and (3.44) for the access from the modified placement of the servers will already be satisfied. The Placement Optimization subproblem is to find the optimal permutations $\zeta_i(j)$. The problem can be formally written as follows.

Objective: min (4.45)

s.t. (4.48), (4.49), (3.49), $\pi_{ij} = \pi'_{i,\zeta_i(j)}$, ζ_i is a permutation on
$\{1, \cdots, m\}$ $\forall$ $i \in \{1, \cdots, r\}$

var. $\zeta_i(j) \quad \forall j \in \{1, \cdots, m\}$ and $i \in \{1, \cdots, r\}$

We note that the optimization problem is to find r permutations and is a discrete optimization problem. We first consider optimizing only over one of the permutation ζ_i. Let ζ_i be written as an indicator function $x^{(i)}_{u,v}$ which is 1 if $v = \zeta_i(u)$ and zero otherwise. Then, the new $\pi_{ij} = \sum_u x^{(i)}_{j,u} \pi'_{iu}$ while for other files $k \neq i$, π_{ij} remains the same. With the new values of π_{ij}, the only optimization variables are $x^{(i)}_{j,u}$. The constraints for $x^{(i)}_{u,v}$ are $\sum_v x^{(i)}_{u,v} = \sum_u x^{(i)}_{u,v} = 1$ and $x^{(i)}_{u,v} \in \{0, 1\}$. We note that this is a non-linear bipartite matching problem [86]. All the r permutations taken together result in rm^2 discrete optimization variables that we wish to optimize.

In general, we have the constraints $\pi_{ij} = \sum_u x^{(i)}_{j,u} \pi'_{iu}$ and $\sum_v x^{(i)}_{u,v} = \sum_u x^{(i)}_{u,v} = 1$ for all $i \in \{1, \cdots, r\}$, $u, v \in \{1, \cdots, m\}$, where binary $x^{(i)}_{u,v}$ for each i, u, v are the decision variables. In order to solve the non-linear problem with integer constraints, we use NOVA algorithm, where a term $\left(1 + e^{(\alpha_c x)}\right)^{-1} - \left(1 + e^{(\alpha_c (x-1))}\right)^{-1}$ is added in the objective for each constraint (to make the problem smooth), where α_c is a large number and C is large enough to force the solutions to be binary. NOVA algorithm guarantees convergence for any given value of C and thus for large enough C, we will obtain the stationary point that has integer constraints.

A.7 Proof of Lemma 19

The constraints (3.47) and (4.57) are separable for each $\widetilde{t}_i$ and $\bar{t}_i$ and thus it is enough to prove convexity of $C(t) = \alpha_j \left(e^{(\beta_j - \tau)t} - 1\right) + t$. Thus, it is enough to prove that $C''(t) \geq 0$.

The first derivative of $C(t)$ is given as

$$C'(t) = \alpha_j \left((\beta_j - \tau)\, e^{(\beta_j - \tau)t}\right) + 1 \tag{A.8}$$

Differentiating it again, we get the second derivative as follows.

$$C''(t) = \alpha_j \left(\beta_j - \tau\right)^2 e^{(\beta_j - \tau)t} \tag{A.9}$$

Since $\alpha_j > 0$, $C''(t)$ given in (A.9) is non-negative, which proves the Lemma.

A.8 Proof of Lemma 20

The constraints (3.49) and (4.58) are separable for each each $\widetilde{t}_i$ and $\bar{t}_i$, and thus it is enough to prove convexity of $E(t) = \sum_{f=1}^{r} \pi_{fj} \lambda_f \left(\frac{\alpha_j e^{\beta_j t}}{\alpha_j - t}\right)^{L_f} - (\Lambda_j + t)$ for $t < \alpha_j$. Thus, it is enough to prove that $E''(t) \geq 0$ for $t < \alpha_j$. We further note that it is enough to prove that $D''(t) \geq 0$, where $D(t) = \frac{e^{L_f \beta_j t}}{(\alpha_j - t)^{L_f}}$. Hence, the first derivative of $D(t)$ is given as

$$D^{'}(t) = \frac{L_f e^{L_f \beta_j t} \left[\beta_j + (\alpha_j - t)^{-1}\right]}{(\alpha_j - t)^{L_f}} > 0 \tag{A.10}$$

Note that $D'(t) > 0$ since $\alpha_j > t$. Differentiating it again to get the second derivative, we get the second derivative as follows.

$$\begin{aligned} D^{''}(t) = & \frac{L_f \beta_j e^{L_f \beta_j t}}{(\alpha_j - t)^{L_f + 2}} \times \\ & \left[\beta_j + (1 + L_f)\left(\alpha_j - t\right)^{-1} \left(1 + \frac{1}{\beta_j \left(\alpha_j - t\right)}\right)\right] \end{aligned} \tag{A.11}$$

Since $\alpha_j > t$, $D''(t)$ given in (A.11) is non-negative, which proves the Lemma.

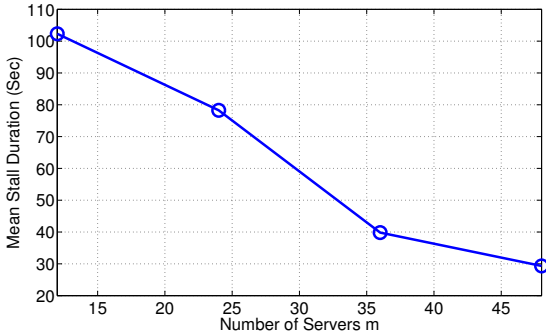

Fig. A.1.: Mean stall duration for 2000 files and different number of servers m

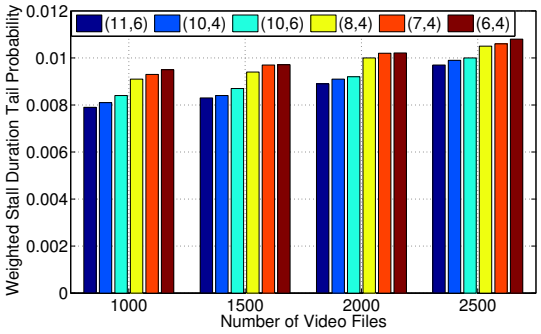

Fig. A.2.: Weighted stall duration tail probability for different coding with different video lengths.

A.9 Additional Simulation Figures

In this section, in addition to the variations studied earlier, we will explore the effects of changing some other system parameters, i.e., the number of servers, the number of video files, the increase of video request arrival rates, and the code choice on the stall durations.

Effect of number of servers: Figure A.1 depicts the mean stall duration for increasing number of servers (12, 24, 36, 48). We note that the mean stall duration decreases with increase of servers.

Effect of encoding parameters: Figure A.2 depicts the weighted stall duration tail probability for varying the number of files, and for different choices of code

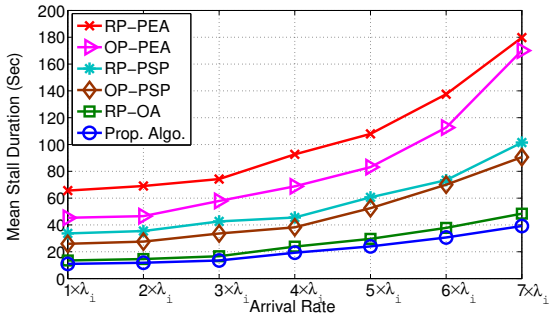

Fig. A.3.: Mean Stall Duration for replication-based setup ($k = 1$). We set $m = 24$ servers, $r = 2000$ video files, arrival rate is varied from $1 \times \lambda_i$ to $7 \times \lambda_i$, where λ_i is the base arrival rate. The video file sizes are Pareto-based distributed, i.e., can be anywhere between 1-120 minutes.

parameters. We first note that the weighted stall duration tail probability is higher for larger number of files. Further, we note that the code with larger n for the same value of k performs better. This is because larger value of n gives more choice for the selection of servers. Thus, $(11, 6)$ performs better than $(10, 6)$ and $(8, 4)$ performs better than $(7, 4)$. Among $(10, 6)$ and $(8, 4)$, the additional redundancy is 4. With the same number of parity symbols, it is better to have larger value of k since smaller chunks are obtained from each server helping stall durations. Since the replication has $k = 1$, this analysis thus shows that an erasure code with the same redundancy can help achieve better stall durations.

Performance with Repetition Coding: Figure A.3 shows the effect of different video arrival rates on the mean stall duration for different-size video length when each file uses $(3, 1)$ erasure-code (which is triple-replication). We compare our proposed algorithm with the five baseline policies and see that the proposed algorithm outperforms all baseline strategies for the QoE metric of mean stall duration. Thus, both access and placement of files are important for reducing the mean stall duration. We see that the mean stall duration of all approaches increases with arrival rates. However, since the mean stall duration is more significant at high arrival rates,

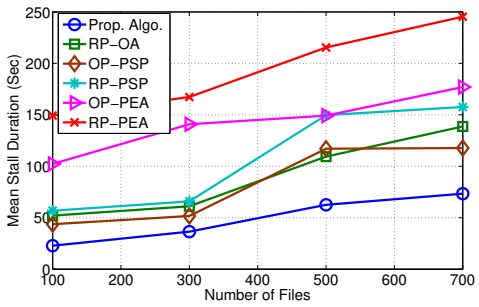

Fig. A.4.: Mean stall duration for different number of video files with different video lengths.

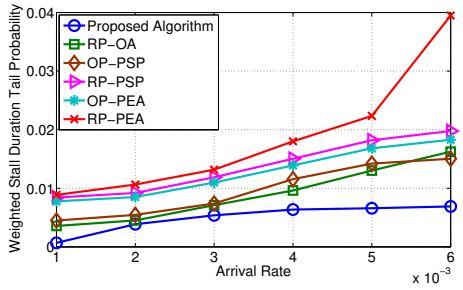

Fig. A.5.: Stall duration tail probability for different arrival rates for video files ($x = 150$ s).

we see the significant improvement in the mean stall duration of our approach as compared to the considered baselines.

Effect of Arrival Rates Figure A.5 demonstrates the effect of increasing workload, obtained by varying the arrival rates of the video files from 0.25λ to 2λ, where λ is the base arrival rate, on the stall duration tail probability for video lengths generated based on Pareto distribution defined above. We notice a significant improvement of the QoE metric with the proposed strategy as compared to the baselines. At the arrival rate of 2λ, the proposed strategy reduces the stall duration tail probability by about 100% as compared to the random placement and projected equal access policy.

Convergence of Stall Duration Tail Probability Figure A.6 demonstrates the convergence of our proposed algorithm for different values of x. Considering

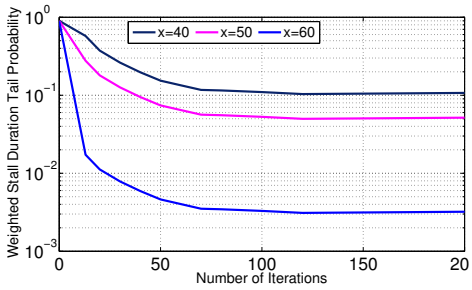

Fig. A.6.: Stall Duration Tail Probability for different number of iterations.

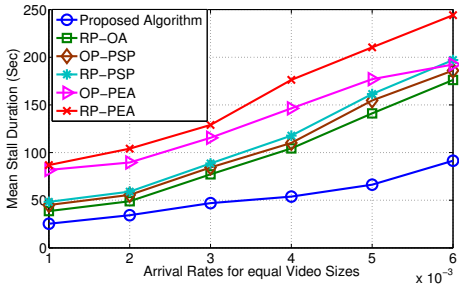

Fig. A.7.: Mean stall duration for different video arrival rates for 600s video files.

$r = 1000$ files of length 300s each with $m = 12$ storage nodes, the stall duration tail probability converges to the optimal value within less than 200 iterations.

Effect of the Number of Video Files Figure A.4 demonstrates the impact of varying the number of video files from 100 files to 700 files on the mean stall duration, where the video lengths are generated according to Pareto distribution with the same parameter defined earlier (scale of 300, and shape of 2). We note that the proposed optimization strategy effectively reduces the mean stall duration and outperforms the considered baseline strategies. Thus, joint optimization over all three variables $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\pi}$, and $\boldsymbol{t}$ helps reduce the mean stall duration significantly.

A.10 Extension to more streams between the server and the edge router

In this section, we investigate extending the proposed approach to the case when there are y parallel streams from each server to the edge router. Multiple streams can

help obtain parallel video files thus helping one file not wait behind the other. We label the y streams from server j as $\nu_j \in \{1, \cdots, y\}$ (graphically depicted in Figure A.8). The analysis in this work considers only one stream between the server and the edge router. We now show how the analysis can be adapted when there are multiple streams. We first note that the scheduling need to decide not only the server j but also the parallel stream ν_j. We assume that the parallel stream ν_j is chosen equally likely. Further, the multiple streams are obtained through equal bandwidth splits, and thus the service time parameters would be different for streams as compared to the server. For instance, the service rate would be a factor of y of the service rate from the server due to the bandwidth split. Thus, the probability of choosing server j and stream ν_j is

$$q_{i,j,\nu_j} = \pi_{i,j}/y, \tag{A.12}$$

where $\pi_{i,j}$ is the probability of choosing server j. Using this, we note that the analysis of download time from a server can be modified to download time from a stream of a server and the steps can be directly extended. The ordered statistics can use the above probabilistic scheduling to choose a stream of a server and thus the entire analysis can be easily extended.

Since the optimization also has the same parameters, we show an improvement of the mean stall duration with the number of parallel streams in Fig. A.9. The choice of the number of streams y can be determined by the practical limitations (*e.g.*, number of ports possible at the server). A more detailed analysis of the parallel streams, exploiting the flexibility of splitting of bandwidths among the different streams, choosing one of the multiple parallel streams for each video are being considered in the following chapters.

A.11 Impact Of Caching

So far, our analysis did not account for caching. In this Appendix, we present how our model can be extended to accommodate for the impact of caching. Caching

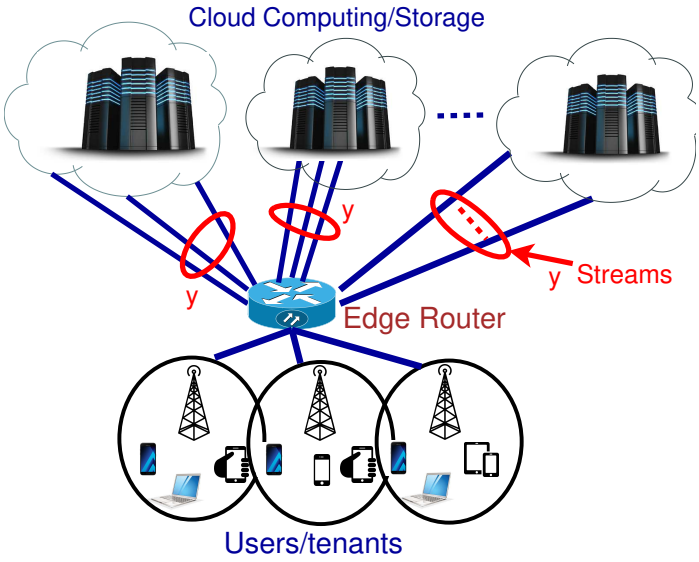

Fig. A.8.: An Illustration of a distributed storage system where a server has y parallel streams to the edge router.

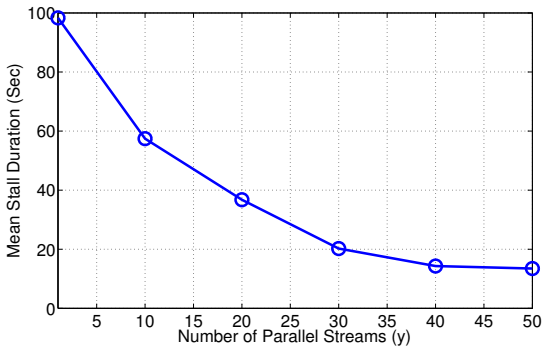

Fig. A.9.: Mean stall duration for different number of parallel streams.

content at the network edge, closer to the customers, can further help reducing the stall duration and thus improve the QoE. However, caching the video content has to address a number of crucial challenges that differ from caching of web objects, see for instance [87] and the references therein for detailed treatment of this aspect.

There are two methods for caching the video files. The first involves caching the complete video file (all L_i video chunks of file i) at edge routers. The second method involves caching partial chunks, i.e., $L_{j,i}$, where $L_{j,i} \leq L_i$, for video file i. Most of the current caching schemes cache entire files (for example, hot files). Our analysis can accommodate both of these methods. In the first method, the video file is entirely cached, and is thus not requested from the servers. This is equivalent to changing the arrival rate of these files to zero, i.e., $\lambda_i = 0$. In the second method, only the later $(L_i - L_{j,i})$ are needed from the servers. This can be easily incorporated by requesting the video of length $(L_i - L_{j,i})$, while the first chunk can wait for an additional $\tau L_{j,i}$ time which can be accounted by adding $\tau L_{j,i}$ in the startup delay for this file.

Mathematically, we can show that for $g \in L_{j,i} + 1, \cdots, L_i$, where $L_{j,i} < g \leq L_i$, the random download time of the remaining $(L_i - L_{j,i})$ segments from server j is given as

$$D_{i,j}^{(g)} = W_j + \sum_{v=L_{j,i}+1}^{g} Y_j^{(v)}. \tag{A.13}$$

Since video file i consists of $(L_i - L_{j,i})$ segments stored at server j, the total service time for video file i, denoted by $ST_{i,j}$, is given as

$$ST_{i,j} = \sum_{v=L_{j,i}+1}^{L_i} Y_{i,j}^{(v)} \tag{A.14}$$

Hence, the service time of the video files at server j is given by

$$R_j = \begin{cases} ST_{i,j} & \text{with prob. } \frac{\lambda_i \pi_{i,j}}{\Lambda_j} \quad \forall i \end{cases} \tag{A.15}$$

where Λ_j is the total arrival rate at server j. Also, we can show that the MGF of the service time for all video files from server j is given by

$$B_j(t) = \mathbb{E}[e^{tR_j}] = \sum_{i=1}^{r} \frac{\lambda_i \pi_{i,j}}{\Lambda_j} \left(\frac{\alpha_j e^{\eta_j t}}{\alpha_j - t} \right)^{(L_i - L_{j,i})} \tag{A.16}$$

Further, the current load intensity at server j, ρ_j, is as follows

$$\rho_j = \Lambda_j B_j'(0) = \sum_{i=1}^{r} \lambda_i \pi_{i,j} (L_i - L_{j,i}) \left(\eta_j + \frac{1}{\alpha_j} \right) \tag{A.17}$$

Similar to the previous analysis, since the arrival is Poisson and the service time is shifted-exponentially distributed, the MGF of the waiting time at queue server j can be calculated usingthe Pollaczek-Khinchine formula, i.e.,

$$\mathbb{E}\left[e^{tW_j}\right] = \frac{(1-\rho_j)t}{t - \Lambda_j (B_j(t) - 1)} \tag{A.18}$$

From the MGF of W_j and the service time, the MGF of the download time of segment g from server j for file i is then

$$\mathbb{E}\left[e^{tD_{i,j}^{(g)}}\right] = \frac{(1-\rho_j)t}{t - \Lambda_j (B_j(t) - 1)} \left(\frac{\alpha_j e^{\eta_j t}}{\alpha_j - t} \right)^{g}. \tag{A.19}$$

Having characterized the download time for chunk g, we can then determine the stall durations and evaluate the QoE metrics as in Equations (A.75) and (3.30). The details are omitted here as they can easily follow.

A.12 End-to-End Analysis

In this Appendix, we show how our analysis can be extended to consider the last hop from the edge-router to the user. If the last hop is considered, the download time of the chunk q for video file i, if requested from server j can be written as follows

$$D_{i,j}^{(q)} = W_j + \sum_{v=1}^{q} Y_j^{(v)} + \frac{\tau \mathbb{R}_i}{\mathbb{C}_i},$$

where W_j is the waiting time in the queue of server j, $Y_j^{(v)}$ is the service time for the chunk v, τ is the chunk size in seconds, $\mathbb{R}_i$ is the bit-rate for user i, and $\mathbb{C}_i$ is the average bandwidth when downloading chunk q. Thus, as long as $\frac{\tau \mathbb{R}_i}{\mathbb{C}_i}$ can be bounded, this is the additional stall duration (or additional startup delay). In most wired setups, the capacity for the last hop may not be a bottleneck, and thus this term is negligible and not varying significantly with q. Even for wireless network in homes, the average bandwidth numbers are much higher than the video rate, and thus this additional term may not be a bottleneck. Thus, the analysis can be easily extended to the last hop. Since the last hop is dependent on the user and the cloud provider wishes to optimize the system such that it does the best delivery in the part controlled by the provider, we did not explicitly consider the last hop. However, as long as the last hop capacity is higher than the data rate of the video, the last hop does not affect the analysis except a small additional delay.

A.13 Download and Play Times of a Segment not Requested in ω_i

Since we assume the edge-router index, we will omit ℓ in this section. In order to characterize the stall duration tail probability, we need to find the download time and the play time of different video segments, for any server j and streams with the choice of β_j and ν_j, assuming that they are not requested in ω_i. The optimization over these decision variables will be considered in this work.

A.13.1 Download Times of first $L_{j,i}$ Segments

We consider a queueing model, where $W^{(e)}_{j,\nu_j}$ denotes the random waiting time of all video files in the queue of $PS^{(e,j)}_{\nu_j}$ before file i request is served, and $Y^{(e,g)}_{i,j,\nu_j}$ be the (random service time of a coded chunk g for file i from server j and queue ν_j. Then, for $g \leq L_{j,i}$, the random download time of the first $L_{j,i}$ segments $g \in \{1, \ldots, L_{j,i}\}$ if file i from stream $PS^{(e,j)}_{\nu_j}$ is given as

$$D^{(g)}_{i,j,\beta_j,\nu_j} = W^{(e)}_{j,\nu_j} + \sum_{v=1}^{g} Y^{(e,v)}_{i,j,\nu_j} \tag{A.20}$$

Since video file i consists of $L_{j,i}$ segments stored at cache server j, the total service time for video file i request at queue $PS^{(e,j)}_{\nu_j}$, denoted by ST_{i,j,ν_j}, is given as

$$ST_{i,j,\nu_j} = \sum_{v=1}^{L_{j,i}} Y^{(e,v)}_{i,j,\nu_j} \tag{A.21}$$

Hence, the service time of the video files at the parallel stream $PS^{(e,j)}_{\nu_j}$ is given as

$$R^{(e)}_{j,\nu_j} = \left\{ ST^{(e,L_{j,i})}_{i,j,\nu_j} \quad \text{with prob. } \frac{\lambda_i \pi_{i,j} p_{i,j,\nu_j} e^{\lambda_i \omega_i}}{\Lambda^{(e)}_{j,\nu_j}} \quad \forall i \right. \tag{A.22}$$

We can show that the moment generating function of the service time for all video files from parallel stream $PS^{(e,j)}_{\nu_j}$ is given by

$$B^{(e)}_{j,\nu_j}(t) = \mathbb{E}[e^{tR^{(e)}_{j,\nu_j}}] = \sum_{i=1}^{r} \frac{\lambda_i \pi_{i,j} p_{i,j,\nu_j} e^{\lambda_i \omega_i}}{\Lambda^{(e)}_{j,\nu_j}} \left(\frac{\alpha^{(e)}_{j,\nu_j} e^{\eta^{(e)}_{j,\nu_j} t}}{\alpha^{(e)}_{j,\nu_j} - t} \right)^{L_{j,i}} \tag{A.23}$$

Further, based on our 2-stage scheduling policy, the load intensity at $PS^{(e,j)}_{\nu_j}$ is as follows

$$\rho^{(e)}_{j,\nu_j} = \Lambda^{(e)}_{j,\nu_j} B^{(e)'}_{j,\nu_j}(0) \tag{A.24}$$

$$= \sum_{i=1}^{r} \lambda_i \pi_{i,j} p_{i,j,\nu_j} e^{\lambda_i \omega_i} L_{j,i} \left(\eta^{(e)}_{j,\nu_j} + \frac{1}{\alpha^{(e)}_{j,\nu_j}} \right) \tag{A.25}$$

Since the arrival is Poisson and the service time is shifted-exponentially distributed, the moment generating function (MGF) of the waiting time at queue $PS^{(e,j)}_{\nu_j}$ can be calculated usingthe Pollaczek-Khinchine formula, i.e.,

$$\mathbb{E}\left[e^{tW^{(e)}_{j,\nu_j}} \right] = \frac{(1 - \rho^{(e)}_{j,\nu_j})t}{t - \Lambda^{(e)}_{j,\nu_j}(B^{(e)}_{j,\nu_j}(t) - 1)} \tag{A.26}$$

From the MGF of $W_{j,\nu_j}^{(e)}$ and the service time, the MGF of the download time of segment g from the queue $PS_{\nu_j}^{(e,j)}$ for file i is then

$$\mathbb{E}\left[e^{tD_{i,j,\beta_j,\nu_j}^{(g)}}\right] = \frac{(1-\rho_{j,\nu_j}^{(e)})t}{t-\Lambda_{j,\nu_j}^{(e)}(B_{j,\nu_j}^{(e)}(t)-1)}\left(\frac{\alpha_{j,\nu_j}^{(e)}e^{\eta_{j,\nu_j}^{(e)}t}}{\alpha_{j,\nu_j}^{(e)}-t}\right)^{g}. \tag{A.27}$$

We note that the above is defined only when MGFs exist, i.e.,

$$t < \alpha_{j,\nu_j}^{(e)} \tag{A.28}$$

$$0 < t-\Lambda_{j,\nu_j}^{(e)}(B_{j,\nu_j}^{(e)}(t)-1) \tag{A.29}$$

A.13.2 Download Times of last $(L_i - L_{j,i})$ Segments

Since the later video segments $(L_i - L_{j,i})$ are downloaded from the data center, we need to schedule them to the β_j streams using the proposed probabilistic scheduling policy. We first determine the time it takes for chunk g to depart the first queue (i.e., β_j queue at datacenter). For that, we define the time of chunk g to depart the first queue as

$$E_{i,j,\beta_j}^{(g)} = W_{j,\beta_j}^{(d)} + \sum_{v=L_{j,i}+1}^{L_i} Y_{j,\beta_j}^{(d,v)}, \tag{A.30}$$

where $W_{j,\beta_j}^{(d)}$ is the waiting time from $PS_{\beta_j}^{(d,j)}$ for the earlier video segments, and $Y_{j,\beta_j}^{(d,v)}$ is the service time for obtaining segment v from the queue of $PS_{\beta_j}^{(d,j)}$. Using similar analysis for that of deriving the MGF of download time of chunk g as in the last section, we obtain

$$\mathbb{E}\left[e^{tE_{i,j,\beta_j}^{(g)}}\right] = \frac{(1-\rho_{j,\beta_j}^{(d)})t}{t-\Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(t)-1)}\left(\frac{\alpha_{j,\beta_j}^{(d)}e^{\eta_{j,\beta_j}^{(d)}t}}{\alpha_{j,\beta_j}^{(d)}-t}\right)^{g-L_{j,i}-1}, \tag{A.31}$$

where the load intensity at queue β_j at datacenter, $\rho_{j,\beta_j}^{(d)}$

$$\rho_{j,\beta_j}^{(d)} = \sum_{i=1}^{r}\lambda_i\pi_{i,j}q_{i,j,\beta_j}e^{\lambda_i\omega_i}\left(L_i-L_{j,i}\right)\left(\eta_{j,\beta_j}^{(d)}+\frac{1}{\alpha_{j,\beta_j}^{(d)}}\right) \tag{A.32}$$

$$B_{j,\beta_j}^{(d)}(t) = \sum_{i=1}^{r}\frac{\lambda_i\pi_{i,j}q_{i,j,\beta_j}e^{\lambda_i\omega_i}}{\Lambda_{j,\beta_j}^{(d)}}\left(\frac{\alpha_{j,\beta_j}^{(d)}e^{\eta_{j,\beta_j}^{(d)}t}}{\alpha_{j,\beta_j}^{(d)}-t}\right)^{L_i-L_{j,i}} \tag{A.33}$$

To find the download time of video segments from the second queue (at cache server j), we notice that the download time for segment g includes the waiting to download all previous segments and the idle time if the segment g is not yet downloaded from the first queue ($PS_{\beta_j}^{(d,j)}$), as well as the service time of the segment from $PS_{\beta_j}^{(\bar{d},j)}$. Then, the download time of the video segments from the second queue (i.e.,$PS_{\beta_j}^{(\bar{d},j)}$) can be derived by a set of recursive equations, with download time of the first (initial) segment $(L_{j,i}+1)$ as

$$D_{i,j,\beta_j,\nu_j}^{(L_{j,i}+1)} = \max(W_{j,\beta_j}^{(\bar{d})}, E_{j,\beta_j}^{(L_{j,i}+1)}) + Y_{j,\beta_j}^{(\bar{d},L_{j,i}+1)}, \tag{A.34}$$

where $W_{j,\beta_j}^{(\bar{d})}$ is the waiting time from queue $PS_{\beta_j}^{(\bar{d},j)}$ for the previous video segments, and $Y_{j,\beta_j}^{(\bar{d},v)}$ is the required service time for obtaining segment v from the queue of $PS_{\beta_j}^{(\bar{d},j)}$. The download time of the following segments $(g > L_{j,i}+1)$ is given by the following recursive equation

$$D_{i,j,\beta_j,\nu_j}^{(g)} = \max(D_{i,j,\beta_j,\nu_j}^{(g-1)}, E_{i,j,\beta_j}^{(g)}) + Y_{j,\beta_j}^{(\bar{d},g)}. \tag{A.35}$$

With the above recursive equations from $y = L_{j,i}$ to $y = g$, we can obtain that

$$D_{i,j,\beta_j,\nu_j}^{(g)} = \max_{y=L_{j,i}}^{g} U_{i,j,\beta_j,g,y}, \tag{A.36}$$

where

$$U_{i,j,\beta_j,g,L_{j,i}} = W_{j,\beta_j}^{(\bar{d})} + \sum_{h=L_{j,i}+1}^{g} Y_{j,\beta_j}^{(\bar{d},h)} \tag{A.37}$$

Similarly, for $y > L_{j,i}$, we have

$$U_{i,j,\beta_j,g,y} = E_{i,j,\beta_j}^{(y)} + \sum_{h=y}^{g} Y_{j,\beta_j}^{(\bar{d},h)}. \tag{A.38}$$

It is easy to see that the moment generating function of $U_{i,j,\beta_j,g,y}$ for $y = L_{j,i}$ is given by

$$\mathbb{E}[e^{tU_{i,j,\beta_j,g,L_{j,i}}}] = \frac{(1-\rho_{j,\beta_j}^{(\bar{d})})t}{t - \Lambda_{j,\beta_j}^{(c)}(B_{j,\beta_j}^{(\bar{d})}(t)-1)} \left(\frac{\alpha_{j,\beta_j}^{(\bar{d})} e^{\eta_{j,\beta_j}^{(\bar{d})} t}}{\alpha_{j,\beta_j}^{(\bar{d})} - t} \right)^{g-L_{j,i}}, \tag{A.39}$$

where the load intensity at queue β_j at cache server j, $\rho^{(\bar{d})}_{j,\beta_j}$ is given by

$$\rho^{(\bar{d})}_{j,\beta_j} = \sum_{i=1}^{r} \lambda_i \pi_{i,j} q_{i,j,\beta_j} e^{\lambda_i \omega_i} \left(L_i - L_{j,i}\right) \left(\eta^{(\bar{d})}_{j,\beta_j} + \frac{1}{\alpha^{(\bar{d})}_{j,\beta_j}}\right) \tag{A.40}$$

$$B^{(\bar{d})}_{j,\beta_j}(t) = \sum_{i=1}^{r} \frac{\lambda_i \pi_{i,j} q_{i,j,\beta_j} e^{\lambda_i \omega_i}}{\Lambda^{(c)}_{j,\beta_j}} \left(\frac{\alpha^{(\bar{d})}_{j,\beta_j} e^{\eta^{(\bar{d})}_{j,\beta_j} t}}{\alpha^{(\bar{d})}_{j,\beta_j} - t}\right)^{L_i - L_{j,i}} \tag{A.41}$$

Similarly, the moment generating function of $U_{i,j,\beta_j,g,y}$ for $y > L_{j,i}$ is given as

$$\mathbb{E}[e^{tU_{i,j,\beta_j,g,y}}] = \overline{W}^{(d)}_{j,\beta_j} \times \left(\frac{\alpha^{(d)}_{j,\beta_j} e^{\eta^{(d)}_{j,\beta_j} t}}{\alpha^{(d)}_{j,\beta_j} - t}\right)^{y - L_{j,i} - 1} \left(\frac{\alpha^{(\bar{d})}_{j,\beta_j} e^{\eta^{(\bar{d})}_{j,\beta_j} t}}{\alpha^{(\bar{d})}_{j,\beta_j} - t}\right)^{g - y + 1} \tag{A.42}$$

where $\overline{W}^{(d)}_{j,\beta_j}$ $\overline{W}^{(d)}_{j,\beta_j} = \frac{(1-\rho^{(d)}_{j,\beta_j}) t B^{(d)}_{j,\beta_j}(t)}{t - \Lambda^{(d)}_{j,\beta_j}(B^{(d)}_{j,\beta_j}(t) - 1)}$. We further note that these moment generating functions are only defined when the MGF functions exist, i.e.,

$$t < \alpha^{(d)}_{j,\beta_j}; \quad 0 < t - \Lambda^{(d)}_{j,\beta_j}(B^{(d)}_{j,\beta_j}(t) - 1) \tag{A.43}$$

$$t < \alpha^{(\bar{d})}_{j,\beta_j}; \quad 0 < t - \Lambda^{(\bar{d})}_{j,\beta_j}(B^{(\bar{d})}_{j,\beta_j}(t) - 1) \tag{A.44}$$

A.13.3 Play Times of different Segments

Next, we find the play time of different video segments. Recall that $D^{(g)}_{i,j,\beta_j,\nu_j}$ is the download time of segment g from ν_j and β_j queues at client i. We further define $T^{(g)}_{i,j,\beta_j,\nu_j}$ as the time that segment g begins to play at the client i, given that it is downloaded from β_j and ν_j queues. This start-up delay of the video is denoted by d_s. Then, the first segment is ready for play at the maximum of the startup delay and the time that the first segment can be downloaded. This means

$$T^{(1)}_{i,j,\beta_j,\nu_j} = \max\left(d_s,\, D^{(1)}_{i,j,\beta_j,\nu_j}\right). \tag{A.45}$$

For $1 < g \leq L_i$, the play time of segment g of video file i is given by the maximum of (i) the time to download the segment and (ii) the time to play all previous segment

plus the time to play segment g (i.e., τ seconds). Thus, the play time of segment g of video file i, when requested from server j and from ν_j and β_j queues, can be expressed as

$$T^{(q)}_{i,j,\beta_j,\nu_j} = \max\ \left(T^{(q-1)}_{i,j,\beta_j,\nu_j} + \tau,\ D^{(q)}_{i,j,\beta_j,\nu_j}\right). \tag{A.46}$$

This results in a set of recursive equations, which further yield by

$$\begin{aligned} T^{(L_i)}_{i,j,\beta_j,\nu_j} &= \max\ \left(T^{(L_i-1)}_{i,j,\beta_j,\nu_j} + \tau,\ D^{(L_i)}_{i,j,\beta_j,\nu_j}\right) \\ &= \max\ \left(T^{(L_i-2)}_{i,j,\beta_j,\nu_j} + 2\tau,\ D^{(L_i-1)}_{i,j,\beta_j,\nu_j} + \tau,\ D^{(L_i)}_{i,j,\beta_j,\nu_j}\right) \\ &= \max\left(d_s + (L_i-1)\tau,\ \max_{z=2}^{L_i+1} D^{(z-1)}_{i,j,\beta_j,\nu_j} + (L_i - z + 1)\tau\right) \\ &= \max_{z=1}^{L_i+1} \mathcal{F}_{i,j,\beta_j,\nu_j,z} \end{aligned} \tag{A.47}$$

where $\mathcal{F}_{i,j,\beta_j,\nu_j,z}$ is expressed as

$$\mathcal{F}_{i,j,\beta_j,\nu_j,z} = \begin{cases} d_s + (L_i-1)\tau & , z = 1 \\ D^{(z-1)}_{i,j,\beta_j,\nu_j} + (L_i - z + 1)\tau & 1 < z \le L_i \end{cases} \tag{A.48}$$

We now get the MGFs of the $\mathcal{F}_{i,j,\nu_j,\beta_j,z}$ to use in characterizing the play time of the different segments. Towards this goal, we plug Equation (A.48) into $\mathbb{E}\left[e^{t\mathcal{F}_{i,j,\nu_j,\beta_j,z}}\right]$ and obtain

$$\mathbb{E}\left[e^{t\mathcal{F}_{i,j,\nu_j,\beta_j,z}}\right] = \tag{A.49}$$
$$\begin{cases} e^{(d_s+(L_i-1)\tau)t} & , z = 1 \\ e^{(L_i-z+1)\tau t}\mathbb{E}\left[e^{tD^{(z-1)}_{i,j,\beta_j,\nu_j}}\right] & 1 < z \le (L_i+1) \end{cases}$$

where $\mathbb{E}\left[e^{tD^{(z-1)}_{i,j,\beta_j,\nu_j}}\right]$ can be calculated using equation (A.27) when $1 < z \le (L_{j,i}+1)$ and using equation (A.42) when $z > L_{j,i}+1$.

The last segment should be completed by time $d_s + L_i\tau$ (which is the time at which the playing of the $L_i - 1$ segment finishes). Thus, the difference between the play time of the last segment $T^{(L_i)}_{i,j,\beta_j,\nu_j}$ and $d_s + (L_i - 1)\,\tau$ gives the stall duration. We note that the stalls may occur before any segment and hence this difference will give the

sum of durations of all the stall periods before any segment. Thus, the stall duration for the request of file i from β_j queue, ν_j queue and server j, i.e., $\Gamma_U^{(i,j,\beta_j,\nu_j)}$ is given as

$$\Gamma_U^{(i,j,\beta_j,\nu_j)} = T_{i,j,\beta_j,\nu_j}^{(L_i)} - d_s - (L_i - 1)\,\tau \tag{A.50}$$

Next, we use this expression to derive a tight bound on the SDTP.

A.14 Proof of Lemma 15

From (5.17), we can get

$$e^{h_i\Gamma_{tot}^{(i)}} \stackrel{d}{=} \begin{cases} \max\left(e^{h_i\left(\Gamma_{tot}^{(i)}-\widetilde{t}_i\right)}, 1\right) & 0 \le \widetilde{t}_i \le \omega_i \\ e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}} & \widetilde{t}_i > \omega_i \end{cases} \tag{A.51}$$

By taking the expectation of both sides in (A.51), we can write

$$\mathbb{E}\left[e^{h_i\Gamma_{tot}^{(i)}}|\widetilde{t}_i\right] \le \begin{cases} 1 + \mathbb{E}\left[e^{h_i\left(\Gamma_{tot}^{(i)}-\widetilde{t}_i\right)}\right] & 0 \le \widetilde{t}_i \le \omega_i \\ \mathbb{E}[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}] & \widetilde{t}_i > \omega_i \end{cases}, \tag{A.52}$$

where the expectation in the second case is over the choice of (j,β_j,ν_j) in addition to the queue statistics with arrival and departure rates. Since the arrivals at edge-cache of video files are Poisson, the time till first request for file i, i.e., $\widetilde{t}_i$, is exponentially distributed with rate λ_i. By averaging over $\widetilde{t}_i$, we have

$$\begin{aligned} \mathbb{E}\left[e^{h_i\Gamma_{tot}^{(i)}}\right] \le & \int_0^{\omega_i} \left(1 + \mathbb{E}\left[e^{h_i\left(\Gamma_{tot}^{(i)}-\widetilde{t}_i\right)}\right]\right) \lambda_i e^{-\lambda_i\widetilde{t}_i} d\widetilde{t}_i \\ & + \int_{\omega_i}^{\infty} \lambda_i \mathbb{E}[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}] e^{-\lambda_i\widetilde{t}_i} d\widetilde{t}_i \end{aligned} \tag{A.53}$$

Performing the integration and simplifying the expressions, we get

$$\begin{aligned} \mathbb{E}\left[e^{h_i\Gamma_{tot}^{(i)}}\right] \le & \left(1 - e^{-\lambda_i\omega_i}\right) + \mathbb{E}\left[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}\right] e^{-\lambda_i\omega_i} \\ & + \frac{\lambda_i \mathbb{E}\left[e^{h_i\Gamma_{tot}^{(i)}}\right]}{\lambda_i + h_i}\left(1 - e^{-(\lambda_i+h_i)\omega_i}\right) \end{aligned} \tag{A.54}$$

This can be further simplified as follows.

$$\begin{aligned}\mathbb{E}\left[e^{h_i\Gamma_{tot}^{(i)}}\right] &\leq \frac{\left(1-e^{-\lambda_i\omega_i}\right)+e^{-\lambda_i\omega_i}\mathbb{E}\left[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}\right]}{1-\frac{\lambda_i}{\lambda_i+h_i}\left(1-e^{-(\lambda_i+h_i)\omega_i}\right)} \\ &\overset{(a)}{=} \frac{c}{b}+\frac{a}{b}\mathbb{E}\left[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}\right] \\ &\overset{(b)}{=} \widetilde{c}+\widetilde{a}\,\mathbb{E}\left[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}\right] \end{aligned} \tag{A.55}$$

where (a) and (b) follow by setting $c=1-e^{-\lambda_i\omega_i}$, $a=e^{-\lambda_i\omega_i}$, $b=\left[1-\frac{\lambda_i}{\lambda_i+h_i}\left(1-e^{-(\lambda_i+h_i)\omega_i}\right)\right]$, $\widetilde{c}=c/b$ and $\widetilde{a}=a/b$. We also recall that the expectation in $\mathbb{E}\left[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}\right]$ is over the choice of (j,β_j,ν_j) and the queue arrival/departure statistics.

A.15 Proof of Lemma 17

We have

$$\begin{aligned} & \mathbb{E}\left[e^{t_iD_{i,j,\beta_j,\nu_j}^{(v)}}|(j,\beta_j,\nu_j)\right] \\ \overset{(a)}{=} & \mathbb{E}\left[e^{t_i(\max_{y=L_{j,i}}^{v}U_{i,j,\beta_j,g,y})}|(j,\beta_j,\nu_j)\right] \\ = & \mathbb{E}\left[\max_{y=L_{j,i}}^{v}e^{t_iU_{i,j,\beta_j,g,y}}|(j,\beta_j,\nu_j)\right] \\ \leq & \sum_{y=L_{j,i}}^{v}\mathbb{E}\left[e^{t_iU_{i,j,\beta_j,g,y}}|(j,\beta_j,\nu_j)\right] \\ = & \mathbb{E}\left[e^{t_iU_{i,j,\beta_j,g,L_{j,i}}}|(j,\beta_j,\nu_j)\right]+ \\ & \sum_{w=L_{j,i}+1}^{v}\frac{(1-\rho_{j,\beta_j}^{(d)})t_i}{t_i-\Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(t_i)-1)}\times \\ & \left(\frac{\alpha_{j,\beta_j}^{(d)}e^{\eta_{j,\beta_j}^{(d)}}}{\alpha_{j,\beta_j}^{(d)}-t_i}\right)^{w-L_{j,i}-1}\left(\frac{\alpha_{j,\beta_j,\ell}^{(\bar{d})}e^{\eta_{j,\beta_j,\ell}^{(\bar{d})}}}{\alpha_{j,\beta_j,\ell}^{(\bar{d})}-t_i}\right)^{v-w+1}, \end{aligned} \tag{A.56}$$

where (a) follows from (A.36), the inequality above follows by replacing the $\max_y(.)$ by $\sum_y(.)$. Moreover, the last step follows from (A.42). Hence, we can write

$$\begin{aligned}
&\mathbb{E}\left[e^{t_i D^{(v)}_{i,j,\beta_j,\nu_j}} | (j,\beta_j,\nu_j)\right] \\
&\leq \frac{(1-\rho^{(\bar{d})}_{j,\beta_j})t}{t-\Lambda^{(c)}_{j,\beta_j}(B^{(\bar{d})}_{j,\beta_j}(t)-1)}\left(\frac{\alpha^{(\bar{d})}_{j,\beta_j}e^{\eta^{(\bar{d})}_{j,\beta_j}t}}{\alpha^{(\bar{d})}_{j,\beta_j}-t}\right)^{g-L_{j,i}} + \\
&\sum_{w=L_{j,i}+1}^{v}\frac{(1-\rho^{(d)}_{j,\beta_j})t_i}{t_i-\Lambda^{(d)}_{j,\beta_j}(B^{(d)}_{j,\beta_j}(t_i)-1)}\times \\
&\left(\frac{\alpha^{(d)}_{j,\beta_j}e^{\eta^{(d)}_{j,\beta_j}}}{\alpha^{(d)}_{j,\beta_j}-t_i}\right)^{w-L_{j,i}-1}\left(\frac{\alpha^{(\bar{d})}_{j,\beta_j}e^{\eta^{(\bar{d})}_{j,\beta_j}}}{\alpha^{(\bar{d})}_{j,\beta_j}-t_i}\right)^{v-w+1},
\end{aligned} \tag{A.57}$$

this proves the statement of the Lemma.

A.16 Proof of Theorem 5.4.1

The SDTP for the request of file i can be bounded using Markov Lemma as follows

$$\mathbb{P}\left(\Gamma^{(i)}_{tot}\geq\sigma\right)\leq\frac{\mathbb{E}\left[e^{h_i\Gamma^{(i)}_{tot}}\right]}{e^{h_i\sigma}} \tag{A.58}$$

This can be further simplified as follows

$$
\begin{aligned}
&\mathbb{P}\left(\Gamma_{tot}^{(i)} \geq \sigma\right) \\
&\overset{(c)}{\leq} \widetilde{c}\, e^{-h_i\sigma} + \widetilde{a}\, e^{-h_i\sigma}\mathbb{E}\left[e^{h_i\Gamma_U^{(i,j,\beta_j,\nu_j)}}\right] \\
&= \widetilde{c}\, e^{-h_i\sigma} + \widetilde{a}\, e^{-h_i\sigma}\mathbb{E}\left[e^{h_i(T_{i,j,\beta_j,\nu_j}^{(L_i)} - (d_s+(L_i-1)\tau))}\right] \\
&= \widetilde{c}\, e^{-h_i\sigma} + \widetilde{a}\, e^{-h_i\sigma}e^{-h_i(d_s+(L_i-1)\tau)}\mathbb{E}\left[e^{h_i T_{i,j,\beta_j,\nu_j}^{(L_i)}}\right] \\
&\overset{(d)}{=} \overline{c} + \overline{a}\,\mathbb{E}\left[e^{h_i \max\limits_{z}\left(\mathcal{F}_{i,j,\beta_j,\nu_j,z}\right)}\right] \\
&= \overline{c} + \overline{a}\,\mathbb{E}\left[\max_{z}\left(e^{h_i\mathcal{F}_{i,j,\beta_j,\nu_j,z}}\right)\right] \\
&\overset{(e)}{\leq} \overline{c} + \overline{a}\sum_{z=1}^{L_i+1}\mathbb{E}\left[e^{h_i\mathcal{F}_{i,j,\beta_j,\nu_j,z}}\right] \\
&\overset{(f)}{=} \overline{c} + \overline{a}\left(e^{h_i(d_s+(L_i-1)\tau)} + \right. \\
&\left. + \sum_{v=1}^{L_i} e^{h_i(L_i-v)\tau}\mathbb{E}\left[e^{h_i D_{i,j,\beta_j,\nu_j}^{(v)}}\right]\right)
\end{aligned}
\tag{A.59}
$$

where $\mathcal{F}_{i,j,\beta_j,\nu_j,z}$ and $D_{i,j,\beta_j,\nu_j}^{(v)}$ are given in Appendix A.13 in Equations (A.48) and (A.27), respectively. Further, (c) follows from (5.18), (d) follows from (A.47) and by setting $\overline{c} = \widetilde{c}\, e^{-h_i\sigma}$, $\overline{a} = \widetilde{a}\, e^{-h_i(\sigma+d_s+(L_i-1)\tau)}$, (e) follows by upper bounding the maximum by the sum, and (f) follows from (A.48). Using the two-stage probabilistic scheduling, the SDTP for video file i is further bounded by

$$
\begin{aligned}
&\Pr\left(\Gamma_{tot}^{(i)} \geq \sigma\right) \leq \\
&\overline{c} + \overline{a}\, e^{h_i(d_s+(L_i-1)\tau)} + \overline{a}\sum_{j=1}^{m}\pi_{i,j}\sum_{\nu_j=1}^{e_j}p_{i,j,\nu_j} \\
&\sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j}\sum_{v=1}^{L_i} e^{h_i(L_i-v)\tau}\mathbb{E}\left[e^{h_i D_{i,j,\beta_j,\nu_j}^{(v)}}|(j,\beta_j,\nu_j)\right].
\end{aligned}
\tag{A.60}
$$

Using Lemmas 16 and 17 for $\mathbb{E}\left[e^{h_i D_r^{(v)}} | (j, \beta_j, \nu_j)\right]$, we obtain the following.

$$
\begin{aligned}
&\Pr\left(\Gamma_{tot}^{(i)} \geq \sigma\right) \leq \\
&\sum_{j=1}^{m} \pi_{i,j} \times \Bigg[\overline{c} + \overline{a}\, e^{h_i(d_s+(L_i-1)\tau)} + \overline{a} \times \\
&\sum_{\nu_j=1}^{e_j} p_{i,j,\nu_j} \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j} \Bigg(\sum_{v=1}^{L_{j,i}} e^{t_i(L_i-v)\tau} \mathbb{E}\left[e^{t_i D_r^{(v)}}\right] + \\
&\sum_{v=1}^{L_{j,i}} e^{t_i(L_i-v)\tau} \mathbb{E}\left[e^{t_i D_r^{(v)}}\right] \Bigg) \Bigg] \\
&\overset{(f)}{\leq} \sum_{j=1}^{m} \pi_{i,j} \times \Bigg[\overline{c} + \overline{a}\, e^{h_i(d_s+(L_i-1)\tau)} + \overline{a} \times \\
&\sum_{\nu_j=1}^{e_j} p_{i,j,\nu_j} \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j} \Bigg(\sum_{v=1}^{L_{j,i}} e^{h_i(L_i-v)\tau} \times \\
&\frac{(1-\rho_{j,\beta_j}^{(e)}) t_i (M_{j,\nu_j}^{(e)}(h_i))^v}{h_i - \Lambda_{j,\beta_j}^{(e)}(B_{j,\beta_j}^{(e)}(h_i)-1)} + \\
&+ \frac{e^{h_i(L_i-v)\tau}(1-\rho_{j,\beta_j}^{(\overline{d})}) t_i (M_{j,\nu_j}^{(\overline{d})}(h_i))^{L_i-L_{j,i}}}{h_i - \Lambda_{j,\beta_j}^{(\overline{d})}(B_{j,\beta_j}^{(\overline{d})}(h_i)-1)} + \\
&+ \sum_{v=L_{j,i}+1}^{L_i} \sum_{w=L_{j,i}+1}^{v} e^{h_i(L_i-v)\tau} \times \\
&\frac{(1-\rho_{j,\beta_j}^{(d)}) t_i (M_{j,\beta_j}^{(d)}(h_i))^{w-L_{j,i}-1}}{\left[h_i - \Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(h_i)-1)\right] (M_{j,\beta_j}^{(\overline{d})}(h_i)^{w-L_i-1}} \Bigg) \Bigg] \\
&= \sum_{j=1}^{m} \pi_{i,j} \times \Bigg[\overline{c} + \overline{a}\, e^{h_i(d_s+(L_i-1)\tau)} + \overline{a} \times \\
&\sum_{\nu_j=1}^{e_j} p_{i,j,\nu_j} \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j} e^{h_i L_i \tau} \times \\
&\Bigg(\frac{\widetilde{M}_{j,\nu_j}^{(e)}(h_i)(1-\rho_{j,\beta_j}^{(e)}) t_i ((\widetilde{M}_{j,\nu_j}^{(e)}(h_i))^{L_{j,i}} - 1)}{(h_i - \Lambda_{j,\beta_j}^{(e)}(B_{j,\beta_j}^{(e)}(h_i)-1))(\widetilde{M}_{j,\nu_j}^{(e)}(h_i)) - 1)} \\
&+ \frac{(1-\rho_{j,\beta_j}^{(\overline{d})}) t_i (\widetilde{M}_{j,\nu_j}^{(\overline{d})}(h_i))^{L_{j,i}-L_i}}{h_i - \Lambda_{j,\beta_j}^{(\overline{d})}(B_{j,\beta_j}^{(\overline{d})}(h_i)-1)} + \sum_{v=L_{j,i}+1}^{L_i}
\end{aligned}
$$

$$\sum_{w=L_{j,i}+1}^{v} \frac{(1-\rho_{j,\beta_j}^{(d)})t_i(\widetilde{M}_{j,\beta_j}^{(\bar{d})}(h_i))^{L_i+1}(\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))^w}{\left[h_i-\Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(h_i)-1)\right](\widetilde{M}_{j,\beta_j}^{(d)}(h_i))^{L_{j,i}+1}}\Bigg)\Bigg] \tag{A.61}$$

$$= \sum_{j=1}^{m} \pi_{i,j} \times \Bigg[\overline{c} + \overline{a}\, e^{h_i(d_s+(L_i-1)\tau)} + \overline{a} \times$$

$$\sum_{\nu_j=1}^{e_j} p_{i,j,\nu_j} \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j} e^{h_i L_i \tau} \times$$

$$\Bigg(\frac{\widetilde{M}_{j,\nu_j}^{(e)}(h_i)(1-\rho_{j,\beta_j}^{(e)})t_i((\widetilde{M}_{j,\nu_j}^{(e)}(h_i))^{L_{j,i}}-1)}{(h_i-\Lambda_{j,\beta_j}^{(e)}(B_{j,\beta_j}^{(e)}(h_i)-1))(\widetilde{M}_{j,\nu_j}^{(e)}(h_i))-1)}$$

$$+ \frac{(1-\rho_{j,\beta_j}^{(\bar{d})})t_i(\widetilde{M}_{j,\nu_j}^{(\bar{d})}(h_i))^{L_{j,i}-L_i}}{h_i-\Lambda_{j,\beta_j}^{(\bar{d})}(B_{j,\beta_j}^{(\bar{d})}(h_i)-1)} +$$

$$\frac{(1-\rho_{j,\beta_j}^{(d)})t_i(\widetilde{M}_{j,\beta_j}^{(\bar{d})}(h_i))^{L_i+1}}{\left[h_i-\Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(h_i)-1)\right](\widetilde{M}_{j,\beta_j}^{(d)}(h_i))^{L_{j,i}+1}} \times$$

$$\Bigg(\frac{(\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))^{L_i-L_{j,i}}-(L_i-L_{j,i})}{(\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))-1} +$$

$$+ \frac{\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i)\left((\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))^{L_i-L_{j,i}-1}-1\right)}{(\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))-1}\Bigg)\Bigg)\Bigg] \tag{A.62}$$

where step (f) follows by substitution of the moment generating functions, and the remaining of the steps use the sum of geometric and Arithmetico-geometric sequences. Note that the subscript ℓ is omitted in the above derivation for simplicity. Further, $\delta^{(e)} = \frac{\widetilde{M}_{j,\nu_j}^{(e)}(h_i)(1-\rho_{j,\beta_j}^{(e)})t_i((\widetilde{M}_{j,\nu_j}^{(e)}(h_i))^{L_{j,i}}-1)}{(h_i-\Lambda_{j,\beta_j}^{(e)}(B_{j,\beta_j}^{(e)}(h_i)-1))(\widetilde{M}_{j,\nu_j}^{(e)}(h_i))-1)}$, $\delta^{(\bar{d})} = \frac{(1-\rho_{j,\beta_j}^{(\bar{d})})t_i(\widetilde{M}_{j,\nu_j}^{(\bar{d})}(h_i))^{L_{j,i}-L_i}}{h_i-\Lambda_{j,\beta_j}^{(\bar{d})}(B_{j,\beta_j}^{(\bar{d})}(h_i)-1)}$, $\delta^{(d,\bar{d})} = \gamma^{(d)}\left(\frac{(\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))^{L_i-L_{j,i}}-(L_i-L_{j,i})}{(\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))-1}+\xi_{i,j,\beta_j}^{(d,\bar{d})}\right)$, $\xi_{i,j,\beta_j}^{(d,\bar{d})} = \frac{\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i)\left((\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))^{L_i-L_{j,i}-1}-1\right)}{(\widetilde{M}_{j,\beta_j}^{(d,\bar{d})}(h_i))-1}$ and $\gamma^{(d)} = \frac{(1-\rho_{j,\beta_j}^{(d)})t_i(\widetilde{M}_{j,\beta_j}^{(\bar{d})}(h_i))^{L_i+1}}{\left[h_i-\Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(h_i)-1)\right](\widetilde{M}_{j,\beta_j}^{(d)}(h_i))^{L_{j,i}+1}}$. Further, $\widetilde{M}_{j,\beta_j}^{(d)}(h_i) = \frac{\alpha_{j,\beta_j}^{(d)} e^{\eta_{j,\beta_j}-h_i\tau}}{\alpha_{j,\beta_j}^{(d)}-h_i}$, $\widetilde{M}_{j,\beta_j}^{(\bar{d})}(h_i) = \frac{\alpha_{j,\beta_j}^{(\bar{d})} e^{\eta_{j,\beta_j}-h_i\tau}}{\alpha_{j,\beta_j}^{(\bar{d})}-h_i}$, $\widetilde{M}_{j,\nu_j}^{(e)}(h_i) = \frac{\alpha_{j,\nu_j}^{(e)} e^{\eta_{j,\nu_j}-h_i\tau}}{\alpha_{j,\nu_j}^{(e)}-h_i}$. This proves the statement of the Theorem.

A.17 Sub-problems Optimization

In this section, we explain how each sub-optimization problem is solved.

Server-PSs Access Optimization

Given the bandwidth allocation weights, the cache placement, edge-cache window size, and the auxiliary variables, this sub-problem can be written as follows.

Input: $\boldsymbol{h}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, **and** $\boldsymbol{L}$

Objective: min (5.34)

s.t. (5.35)– (5.38), –(5.40)– (5.43)

var. $\widetilde{\boldsymbol{\pi}}$

In order to solve this problem, we use iNner cOnVex Approximation (NOVA) algorithm proposed in [57]. The key idea for this algorithm is that the non-convex objective function is replaced by suitable convex approximations at which convergence to a stationary solution of the original non-convex optimization is established. NOVA solves the approximated function efficiently and maintains feasibility in each iteration. The objective function can be approximated by a convex one (e.g., proximal gradient-like approximation) such that the first order properties are preserved [57], and this convex approximation can be used in NOVA algorithm.

Let $\widetilde{U_q}\left(\widetilde{\boldsymbol{\pi}}, \widetilde{\boldsymbol{\pi}}^{\boldsymbol{\nu}}\right)$ be the convex approximation at iterate $\boldsymbol{\pi}^{\boldsymbol{\nu}}$ to the original non-convex problem $U\left(\widetilde{\boldsymbol{\pi}}\right)$, where $U\left(\widetilde{\boldsymbol{\pi}}\right)$ is given by (5.34). Then, a valid choice of $U\left(\widetilde{\boldsymbol{\pi}}; \boldsymbol{\pi}^{\boldsymbol{\nu}}\right)$ is the first order approximation of $U\left(\widetilde{\boldsymbol{\pi}}\right)$, e.g., (proximal) gradient-like approximation, i.e.,

$$\widetilde{U_q}\left(\widetilde{\boldsymbol{\pi}}, \widetilde{\boldsymbol{\pi}}^{\boldsymbol{\nu}}\right) = \nabla_{\widetilde{\boldsymbol{\pi}}} U\left(\widetilde{\boldsymbol{\pi}}^{\nu}\right)^T \left(\widetilde{\boldsymbol{\pi}} - \widetilde{\boldsymbol{\pi}}^{\nu}\right) + \frac{\tau_u}{2} \left\|\widetilde{\boldsymbol{\pi}} - \widetilde{\boldsymbol{\pi}}^{\nu}\right\|^2, \tag{A.63}$$

where τ_u is a regularization parameter. Note that all the constraints (5.35)– (5.38) are separable and linear in $\widetilde{\pi}_{i,j,k}$. The NOVA Algorithm for optimizing $\widetilde{\boldsymbol{\pi}}$ is described in Algorithm 3. Using the convex approximation $\widetilde{U_\pi}\left(\boldsymbol{\pi}; \boldsymbol{\pi}^{\boldsymbol{\nu}}\right)$, the minimization steps in Algorithm 3 are convex, with linear constraints and thus can be solved using a

projected gradient descent algorithm. A step-size (γ) is also used in the update of the iterate $\widetilde{\boldsymbol{\pi}}^{\nu}$. Note that the iterates $\{\boldsymbol{\pi}^{\nu}\}$ generated by the algorithm are all feasible for the original problem and, further, convergence is guaranteed, as shown in [57] and described in lemma 21.

In order to use NOVA, there are some assumptions (given in [57]) that have to be satisfied in both original function and its approximation. These assumptions can be classified into two categories. The first category is the set of conditions that ensure that the original problem and its constraints are continuously differentiable on the domain of the function, which are satisfied in our problem. The second category is the set of conditions that ensures that the approximation of the original problem is uniformly strongly convex on the domain of the function. The latter set of conditions are also satisfied as the chosen function is strongly convex and its domain is also convex. To see this, we need to show that the constraints (5.36)–(5.40) form a convex domain in $\widetilde{\boldsymbol{\pi}}$ which is easy to see from the linearity of the constraints. Further details on the assumptions and function approximation can be found in [57]. Thus, the following result holds.

Lemma 21 *For fixed $\boldsymbol{h}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$, the optimization of our problem over $\widetilde{\boldsymbol{\pi}}$ generates a sequence of decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Auxiliary Variables Optimization

Given the probability distribution of the server-PSs scheduling probabilities, the bandwidth allocation weights, edge-cache window size, and the cache placement, this subproblem can be written as follows.

Input: $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$

Objective: min (5.34)

s.t. (5.35), (5.40)–(5.43),

var. $\boldsymbol{h}$

Algorithm 3: NOVA Algorithm to solve Server Access and PSs selection Optimization sub-problem

1. **Initialize** $\nu = 0$, $k = 0$,$\gamma^{\nu} \in (0, 1]$, $\epsilon > 0$,$\widetilde{\boldsymbol{\pi}}^{0}$ such that $\widetilde{\boldsymbol{\pi}}^{0}$ is feasible ,
2. **while** $\text{obj}\,(k) - \text{obj}\,(k-1) \geq \epsilon$
3. *//Solve for $\widetilde{\boldsymbol{\pi}}^{\nu+1}$ with given $\widetilde{\boldsymbol{\pi}}^{\nu}$*
4. **Step 1**: Compute $\widehat{\boldsymbol{\pi}}\left(\widetilde{\boldsymbol{\pi}}^{\nu}\right)$, the solution of $\widehat{\boldsymbol{\pi}}\left(\widetilde{\boldsymbol{\pi}}^{\nu}\right) = \underset{\widetilde{\boldsymbol{\pi}}}{\text{argmin}}\, \widetilde{\boldsymbol{U}}\left(\widetilde{\boldsymbol{\pi}}, \widetilde{\boldsymbol{\pi}}^{\nu}\right)$ s.t. (5.35)–(5.38), (5.40)–(5.43) solved using projected gradient descent
5. **Step 2**: $\widetilde{\boldsymbol{\pi}}^{\nu+1} = \widetilde{\boldsymbol{\pi}}^{\nu} + \gamma^{\nu}\left(\widehat{\boldsymbol{\pi}}\left(\widetilde{\boldsymbol{\pi}}^{\nu}\right) - \widetilde{\boldsymbol{\pi}}^{\nu}\right)$.
6. *//update index*
7. **Set** $\nu \leftarrow \nu + 1$
8. **end while**
9. **output:** $\widehat{\boldsymbol{\pi}}\left(\widetilde{\boldsymbol{\pi}}^{\nu}\right)$

Similar to Server-PSs Access Optimization, this optimization can be solved using NOVA algorithm. The constraint (5.40) is linear in $\boldsymbol{h}$. Further, the next Lemma show that the constraints (5.41)– (5.43) are convex in $\boldsymbol{h}$, respectively.

Lemma 22 *The constraints* (5.41)–(5.43) *are convex with respect to* $\boldsymbol{h}$.

Proof The proof is given in Appendix A.18. ■

Algorithm 4 shows the used procedure to solve for $\boldsymbol{h}$. Let $\overline{U}\left(\boldsymbol{h}; \boldsymbol{h^{\nu}}\right)$ be the convex approximation at iterate $\boldsymbol{h^{\nu}}$ to the original non-convex problem $U\left(\boldsymbol{h}\right)$, where $U\left(\boldsymbol{h}\right)$ is given by (5.34), assuming other parameters constant. Then, a valid choice of $\overline{U}\left(\boldsymbol{h}; \boldsymbol{h^{\nu}}\right)$ is the first order approximation of $U\left(\boldsymbol{h}\right)$, i.e.,

$$\overline{U}\left(\boldsymbol{h}, \boldsymbol{h^{\nu}}\right) = \nabla_{\boldsymbol{h}} U\left(\boldsymbol{h^{\nu}}\right)^{T}\left(\boldsymbol{h} - \boldsymbol{h^{\nu}}\right) + \frac{\tau_h}{2}\left\|\boldsymbol{h} - \boldsymbol{h^{\nu}}\right\|^{2}. \tag{A.64}$$

where τ_h is a regularization parameter. The detailed steps can be seen in Algorithm 4. Since all the constraints (5.40)– (5.43) have been shown to be convex in $\boldsymbol{h}$, the

Algorithm 4: NOVA Algorithm to solve Auxiliary Variables Optimization sub-problem

1. **Initialize** $\nu = 0$, $\gamma^{\nu} \in (0, 1]$, $\epsilon > 0$, $\boldsymbol{h}^0$ such that $\boldsymbol{h}^0$ is feasible,
2. **while** $\text{obj}(\nu) - \text{obj}(\nu - 1) \geq \epsilon$
3. *//Solve for $\boldsymbol{h}^{\nu+1}$ with given $\boldsymbol{h}^{\nu}$*
4. **Step 1**: Compute $\widehat{\boldsymbol{h}}(\boldsymbol{h}^{\nu})$, the solution of $\widehat{\boldsymbol{h}}(\boldsymbol{t}^{\nu}) = \underset{\boldsymbol{h}}{\text{argmin}}\ \overline{\boldsymbol{U}}(\boldsymbol{h}, \boldsymbol{h}^{\nu})$, s.t. (5.35), (5.40)–(5.43), using projected gradient descent
5. **Step 2**: $\boldsymbol{h}^{\nu+1} = \boldsymbol{h}^{\nu} + \gamma^{\nu}\left(\widehat{\boldsymbol{h}}(\boldsymbol{h}^{\nu}) - \boldsymbol{h}^{\nu}\right)$.
6. *//update index*
7. **Set** $\nu \leftarrow \nu + 1$
8. **end while**
9. **output:** $\widehat{\boldsymbol{h}}(\boldsymbol{h}^{\nu})$

optimization problem in Step 1 of Algorithm 4 can be solved by the standard projected gradient descent algorithm.

Lemma 23 *For fixed $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{w}$, $\boldsymbol{\omega}$, and $\boldsymbol{L}$, the optimization of our problem over $\boldsymbol{h}$ generates a sequence of monotonically decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Bandwidth Allocation Weights Optimization

Given the auxiliary variables, the server access and PSs selection probabilities, edge-cache window size, and cache placement, this subproblem can be written as follows.

Input: $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{L}$, $\boldsymbol{\omega}$, and $\boldsymbol{h}$

Objective: min (5.34)

$$\text{s.t. } (5.35)\text{–}(5.37),\ (5.40)\text{–}(5.43),$$
$$\text{var. } \boldsymbol{w}$$

This optimization problem can be solved using NOVA algorithm. It is easy to notice that the constraints that exist in (5.35)– (5.37) are linear and thus convex with respect to $\boldsymbol{w}$. Further, the next two Lemmas show that the constraints (5.40)–(5.43), are convex in $\boldsymbol{w}$, respectively.

Lemma 24 *The constraints* (5.40)–(5.43) *are convex with respect to* $\boldsymbol{w}$.

Proof The proof is given in Appendix A.18. ■

Algorithm 5 shows the used procedure to solve for $\boldsymbol{w}$. Let $U_w(\boldsymbol{w};\boldsymbol{w}^{\boldsymbol{\nu}})$ be the convex approximation at iterate $\boldsymbol{w}^{\boldsymbol{\nu}}$ to the original non-convex problem $U(\boldsymbol{w})$, where $U(\boldsymbol{w})$ is given by (5.34), assuming other parameters constant. Then, a valid choice of $U_w(\boldsymbol{w};\boldsymbol{w}^{\boldsymbol{\nu}})$ is the first order approximation of $U(\boldsymbol{w})$, i.e.,

$$U_w(\boldsymbol{w},\boldsymbol{w}^{\boldsymbol{\nu}}) = \nabla_{\boldsymbol{w}} U(\boldsymbol{w}^{\boldsymbol{\nu}})^T(\boldsymbol{w}-\boldsymbol{w}^{\boldsymbol{\nu}}) + \frac{\tau_w}{2}\|\boldsymbol{w}-\boldsymbol{w}^{\boldsymbol{\nu}}\|^2. \tag{A.65}$$

where τ_t is a regularization parameter. The detailed steps can be seen in Algorithm 5. Since all the constraints have been shown to be convex, the optimization problem in Step 1 of Algorithm 5 can be solved by the standard projected gradient descent algorithm.

Lemma 25 *For fixed* $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{\omega}$, *and* $\boldsymbol{L}$, *the optimization of our problem over* $\boldsymbol{w}$ *generates a sequence of decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Cache Placement Optimization

Given the auxiliary variables, the server access and PS selection probabilities, edge-cache window size, and the bandwidth allocation weights, this subproblem can be written as follows.

Algorithm 5: NOVA Algorithm to solve Bandwidth Allocation Optimization sub-problem

1. **Initialize** $\nu = 0$, $\gamma^{\nu} \in (0, 1]$, $\epsilon > 0$, $\boldsymbol{w}^0$ such that $\boldsymbol{w}^0$ is feasible,
2. **while** $\text{obj}(\nu) - \text{obj}(\nu - 1) \geq \epsilon$
3. *//Solve for $\boldsymbol{w}^{\nu+1}$ with given $\boldsymbol{w}^{\nu}$*
4. **Step 1**: Compute $\widehat{\boldsymbol{w}}(\boldsymbol{w}^{\nu})$, the solution of $\widehat{\boldsymbol{w}}(\boldsymbol{w}^{\nu}) = \underset{\boldsymbol{b}}{\text{argmin}}\ \overline{\boldsymbol{U}}(\boldsymbol{w}, \boldsymbol{w}^{\nu})$, s.t. (5.35)–(5.37), (5.40)–(5.43), using projected gradient descent
5. **Step 2**: $\boldsymbol{w}^{\nu+1} = \boldsymbol{w}^{\nu} + \gamma^{\nu}(\widehat{\boldsymbol{w}}(\boldsymbol{w}^{\nu}) - \boldsymbol{w}^{\nu})$.
6. *//update index*
7. **Set** $\nu \leftarrow \nu + 1$
8. **end while**
9. **output:** $\widehat{\boldsymbol{w}}(\boldsymbol{w}^{\nu})$

Input: $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{h}$, $\boldsymbol{\omega}$ **and** $\boldsymbol{w}$

Objective: min (5.34)

s.t. (5.35)– (5.38), (5.39), (5.41)– (5.43)

var. $\boldsymbol{L}$

Similar to the aforementioned Optimization sub-problems, this optimization can be solved using NOVA algorithm. Constraints (5.36)– (5.38), are linear in $\boldsymbol{L}$, and hence, form a convex domain. Also, Constraint (5.39) is relaxed to have it convex. Furthermore, the constraints (5.41)– (5.43) are convex as shown in the following Lemmas in this subsection.

Algorithm 6 shows the used procedure to solve for $\boldsymbol{L}$. Let $U_L(\boldsymbol{L}; \boldsymbol{L}^{\boldsymbol{\nu}})$ be the convex approximation at iterate $\boldsymbol{L}^{\boldsymbol{\nu}}$ to the original non-convex problem $U(\boldsymbol{L})$, where

Algorithm 6: NOVA Algorithm to solve Cache Placement Optimization sub-problem

1. **Initialize** $\nu = 0$, $\gamma^{\nu} \in (0, 1]$, $\epsilon > 0$, $\boldsymbol{L}^0$ such that $\boldsymbol{L}^0$ is feasible,
2. **while** $\text{obj}(\nu) - \text{obj}(\nu - 1) \geq \epsilon$
3. *//Solve for $\boldsymbol{L}^{\nu+1}$ with given $\boldsymbol{L}^{\nu}$*
4. **Step 1**: Compute $\widehat{\boldsymbol{L}}(\boldsymbol{L}^{\nu})$, the solution of $\widehat{\boldsymbol{L}}(\boldsymbol{L}^{\nu}) = \underset{\boldsymbol{L}}{\text{argmin}}\ \overline{\boldsymbol{U}}(\boldsymbol{L}, \boldsymbol{L}^{\nu})$, s.t. (5.36)–(5.38), (5.39), (5.41)–(5.43), using projected gradient descent
5. **Step 2**: $\boldsymbol{L}^{\nu+1} = \boldsymbol{L}^{\nu} + \gamma^{\nu}\left(\widehat{\boldsymbol{L}}(\boldsymbol{L}^{\nu}) - \boldsymbol{L}^{\nu}\right)$.
6. *//update index*
7. **Set** $\nu \leftarrow \nu + 1$
8. **end while**
9. **output:** $\widehat{\boldsymbol{L}}(\boldsymbol{L}^{\nu})$

$U(\boldsymbol{L})$ is given by (5.34), assuming other parameters constant. Then, a valid choice of $U_L(\boldsymbol{L}; \boldsymbol{L}^{\boldsymbol{\nu}})$ is the first order approximation of $U(\boldsymbol{L})$, i.e.,

$$U_L(\boldsymbol{L}, \boldsymbol{L}^{\boldsymbol{\nu}}) = \nabla_{\boldsymbol{L}} U(\boldsymbol{L}^{\boldsymbol{\nu}})^T (\boldsymbol{L} - \boldsymbol{L}^{\boldsymbol{\nu}}) + \frac{\tau_L}{2} \|\boldsymbol{L} - \boldsymbol{L}^{\boldsymbol{\nu}}\|^2. \tag{A.66}$$

where τ_L is a regularization parameter. The detailed steps can be seen in Algorithm 5. Since all the constraints have been shown to be convex in $\boldsymbol{L}$, the optimization problem in Step 1 of Algorithm 5 can be solved by the standard projected gradient descent algorithm.

Lemma 26 *For fixed $\boldsymbol{h}$, $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{\omega}$ and $\boldsymbol{w}$, the optimization of our problem over $\boldsymbol{L}$ generates a sequence of monotonically decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Edge-cache Window size Optimization

Given the server access and PS selection probabilities, the bandwidth allocation weights, the cache placement, and the auxiliary variables, this sub-problem can be written as follows.

Input: $\boldsymbol{h}$, $\boldsymbol{w}$, $\widetilde{\boldsymbol{\pi}}$, **and** $\boldsymbol{L}$

Objective: min (5.34)

s.t. (5.35)–(5.38), (5.41)–(5.43)

var. $\boldsymbol{\omega}$

similarly, this optimization can be solved using NOVA algorithm. It is easy to show that Constraints (5.36)–(5.38), are convex in $\boldsymbol{\omega}$, and hence, form a convex domain. Further, the constraints (5.41)–(5.43) are convex as shown in Lemma 27.

Algorithm 6 shows the used procedure to solve for $\boldsymbol{\omega}$. Let $U_{\omega}\left(\boldsymbol{\omega};\boldsymbol{\omega^{\nu}}\right)$ be the convex approximation at iterate $\boldsymbol{\omega^{\nu}}$ to the original non-convex problem $U\left(\boldsymbol{\omega}\right)$, where $U\left(\boldsymbol{\omega}\right)$ is given by (5.34), assuming other parameters constant. Then, a valid choice of $U_{\omega}\left(\boldsymbol{\omega};\boldsymbol{\omega^{\nu}}\right)$ is the first order approximation of $U\left(\boldsymbol{\omega}\right)$, i.e.,

$$U_{\omega}\left(\boldsymbol{\omega},\boldsymbol{\omega^{\nu}}\right)=\nabla_{\boldsymbol{\omega}}U\left(\boldsymbol{\omega^{\nu}}\right)^{T}\left(\boldsymbol{\omega}-\boldsymbol{\omega^{\nu}}\right)+\frac{\tau_{\omega}}{2}\left\|\boldsymbol{\omega}-\boldsymbol{\omega^{\nu}}\right\|^{2}. \tag{A.67}$$

where τ_{ω} is a regularization parameter. The detailed steps can be seen in Algorithm 5. Since all the constraints have been shown to be convex in $\boldsymbol{\omega}$, the optimization problem in Step 1 of Algorithm 5 can be solved by the standard projected gradient descent algorithm.

Lemma 27 *For fixed* $\boldsymbol{h}$, $\widetilde{\boldsymbol{\pi}}$, $\boldsymbol{\omega}$ *and* $\boldsymbol{w}$, *the optimization of our problem over* $\boldsymbol{\omega}$ *generates a sequence of monotonically decreasing objective values and therefore is guaranteed to converge to a stationary point.*

Proof The proof is provided in Appendix A.18. ■

Algorithm 7: NOVA Algorithm to solve Edge-cache window size optimization sub-problem

1. **Initialize** $\nu = 0$, $\gamma^{\nu} \in (0, 1]$, $\epsilon > 0$, $\boldsymbol{\omega}^0$ such that $\boldsymbol{\omega}^0$ is feasible,
2. **while** $\text{obj}(\nu) - \text{obj}(\nu - 1) \geq \epsilon$
3. *//Solve for $\boldsymbol{\omega}^{\nu+1}$ with given $\boldsymbol{\omega}^{\nu}$*
4. **Step 1**: Compute $\widehat{\boldsymbol{\omega}}(\boldsymbol{\omega}^{\nu})$, the solution of $\widehat{\boldsymbol{\omega}}(\boldsymbol{\omega}^{\nu}) = \underset{\boldsymbol{\omega}}{\text{argmin}}\ \overline{\boldsymbol{U}}(\boldsymbol{\omega}, \boldsymbol{\omega}^{\nu})$, s.t. (5.36)–(5.38), (5.39), (5.41)–(5.43)
   using projected gradient descent
5. **Step 2**: $\boldsymbol{\omega}^{\nu+1} = \boldsymbol{\omega}^{\nu} + \gamma^{\nu}(\widehat{\boldsymbol{\omega}}(\boldsymbol{\omega}^{\nu}) - \boldsymbol{\omega}^{\nu})$.
6. *//update index*
7. **Set** $\nu \leftarrow \nu + 1$
8. **end while**
9. **output:** $\widehat{\boldsymbol{\omega}}(\boldsymbol{\omega}^{\nu})$

A.18 Proof of Results in Appendix A.17

A.18.1 Proof of Lemma 22

The constraints (5.41)–(5.43) are separable for each h_i and due to symmetry of the three constraints it is enough to prove convexity of $E(h) = \sum_{f=1}^{r} \pi_{f,j} q_{f,j,\beta_j} \lambda_f e^{-\lambda_i \omega_i} \left(\frac{\alpha}{\alpha - h_i}\right)^{L_f - L_{j,f}} - \left(\Lambda_{j,\beta_j} + h_i\right)$, assuming that the edge router ℓ is unfold, without loss of generality. Thus, it is enough to prove that $E''(h) \geq 0$. We further note that it is enough to prove that $D''(h) \geq 0$, where $D(h) = \left(\frac{\alpha}{\alpha - t_i}\right)^{L_f - L_{j,f}}$. This follows since

$$D'(h) = (L_f - L_{j,f})(1 - \frac{h}{\alpha})^{L_{j,f} - L_f - 1} \times (1/\alpha) \geq 0 \tag{A.68}$$

$$D''(h) = (L_f^2 - L_{j,f}^2 + L_f - L_{j,f})(1 - \frac{h}{\alpha})^{L_{j,f} - L_f - 2} \times (1/\alpha^2) \geq 0 \tag{A.69}$$

A.18.2 Proof of Lemma 24

The constraint (5.41)–(5.43) are separable for each $\alpha^{(d)}_{j,\beta_j}$, $\alpha^{(c)}_{j,\beta_j}$ and $\alpha^{(e)}_{j,\nu_j}$, respectively. Note that we omit the subscript ℓ for simplicity, w.l.o.g. Thus, it is enough to prove convexity of the following three equations

$$E_1(\alpha^{(d)}_{j,\beta_j}) = \sum_{f=1}^{r} \pi_{f,j}\lambda_f q_{f,j,\beta_j} e^{-\lambda_i \omega_i} \left(\frac{\alpha^{(d)}_{j,\beta_j}}{\alpha^{(d)}_{j,\beta_j} - h}\right)^{L_f - L_{j,f}} - \left(\Lambda^{(d)}_{j,\beta_j} + h\right)$$

$$E_2(\alpha^{(\bar{d})}_{j,\beta_j}) = \sum_{f=1}^{r} \pi_{f,j}\lambda_f q_{f,j,\beta_j} e^{-\lambda_i \omega_i} \left(\frac{\alpha^{(\bar{d})}_{j,\beta_j}}{\alpha^{(\bar{d})}_{j,\beta_j} - h}\right)^{L_f - L_{j,f}} - \left(\Lambda^{(\bar{d})}_{j,\beta_j} + h\right)$$

$$E_3(\alpha^{(e)}_{j,\nu_j}) = \sum_{f=1}^{r} \pi_{f,j}\lambda_f p_{f,j,\nu_j} e^{-\lambda_i \omega_i} \left(\frac{\alpha^{(e)}_{j,\nu_j}}{\alpha^{(e)}_{j,\nu_j} - h}\right)^{L_f - L_{j,f}} - \left(\Lambda^{(e)}_{j,\nu_j} + h\right)$$

for $h < \alpha^{(d)}_{j,\beta_j}$, $h < \alpha^{(c)}_{j,\beta_j}$, and $h < \alpha^{(e)}_{j,\nu_j}$, respectively. Since there is only a single index j, β_j, and ν_j, here, we ignore the subscripts and superscripts for the rest of this proof and prove for only one case due to the symmetry. Thus, it is enough to prove that $E_1''(\alpha) \geq 0$ for $h < \alpha$. We further note that it is enough to prove that $D_1''(\alpha) \geq 0$, where $D_1(\alpha) = \left(1 - \frac{h}{\alpha}\right)^{L_{j,i} - L_i}$. This holds since,

$$D_1^{'}(\alpha) = (L_{j,i} - L_i)(1 - \frac{h}{\alpha})^{L_{j,i} - L_i - 1} \times (t/\alpha^2) \tag{A.70}$$

$$D_1^{''}(\alpha) = ((L_{j,i} - L_i)^2 - L_{j,i} + L_i)(1 - \frac{h}{\alpha})^{L_{j,i} - L_i - 2} \times (h/\alpha^3) \geq 0 \tag{A.71}$$

A.19 Mean Stall Duration

In this section, a bound for the mean stall duration, for any video file i, is provided. Since probabilistic scheduling is one feasible strategy, the obtained bound is an upper bound to the optimal strategy.

Using equation (A.50), the stall duration for the request of file i from β_j queue, ν_j queue and server j, $\Gamma_U^{(i,j,\beta_j,\nu_j)}$ is given as

$$\Gamma_U^{(i,j,\beta_j,\nu_j)} = T_{i,j,\beta_j,\nu_j}^{(L_i)} - d_s - (L_i - 1)\,\tau \tag{A.72}$$

An exact evaluation for the play time of segment L_i is hard due to the dependencies between $\mathcal{F}_{i,j,\nu_j,\beta_j,z}$ (i.e., equation (A.48)) random variables for different values of j, ν_j, β_j and z, where $z \in (1, 2, ..., L_i + 1)$. Hence, we derive an upper-bound on the playtime of the segment L_i as follows. Using Jensen's inequality [60], we have for $g_i > 0$,

$$e^{g_i \mathbb{E}\left[T_i^{(L_i)}\right]} \leq \mathbb{E}\left[e^{g_i T_i^{(L_i)}}\right]. \tag{A.73}$$

Thus, finding an upper bound on the moment generating function for $T_i^{(L_i)}$ will lead to an upper bound on the mean stall duration. Hence, we will now bound the moment generating function for $T_i^{(L_i)}$. Using equation (A.62), we can show that

$$\begin{aligned}
\mathbb{E}\left[e^{g_i T_i^{(L_i)}}\right] \leq \sum_{j=1}^{m} \pi_{i,j} \times \Bigg[& \widetilde{c} + \widetilde{a}\, e^{g_i(d_s+(L_i-1)\tau)} + \overline{a} \times \\
& \sum_{\nu_j=1}^{e_j} p_{i,j,\nu_j} \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j} e^{g_i L_i \tau} \times \\
& \Bigg(\frac{\widetilde{M}_{j,\nu_j}^{(e)}(g_i)(1-\rho_{j,\beta_j}^{(e)}) g_i ((\widetilde{M}_{j,\nu_j}^{(e)}(g_i))^{L_{j,i}} - 1)}{(g_i - \Lambda_{j,\beta_j}^{(e)}(B_{j,\beta_j}^{(e)}(g_i)-1))(\widetilde{M}_{j,\nu_j}^{(e)}(g_i)) - 1)} \\
& + \frac{(1-\rho_{j,\beta_j}^{(\overline{d})}) t_i (\widetilde{M}_{j,\nu_j}^{(\overline{d})}(g_i))^{L_{j,i}-L_i}}{g_i - \Lambda_{j,\beta_j}^{(\overline{d})}(B_{j,\beta_j}^{(\overline{d})}(g_i)-1)} + \\
& \frac{(1-\rho_{j,\beta_j}^{(d)}) t_i (\widetilde{M}_{j,\beta_j}^{(\overline{d})}(g_i))^{L_i+1}}{\left[g_i - \Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j}^{(d)}(g_i)-1)\right](\widetilde{M}_{j,\beta_j}^{(d)}(g_i))^{L_{j,i}+1}} \times \\
& \Bigg(\frac{(\widetilde{M}_{j,\beta_j}^{(d,\overline{d})}(g_i))^{L_i - L_{j,i}} - (L_i - L_{j,i})}{(\widetilde{M}_{j,\beta_j}^{(d,\overline{d})}(g_i)) - 1} + \\
& + \frac{\widetilde{M}_{j,\beta_j}^{(d,\overline{d})}(g_i)\left((\widetilde{M}_{j,\beta_j}^{(d,\overline{d})}(g_i))^{L_i - L_{j,i}-1} - 1\right)}{(\widetilde{M}_{j,\beta_j}^{(d,\overline{d})}(g_i)) - 1} \Bigg) \Bigg) \Bigg] \\
= & \sum_{j=1}^{m} \pi_{i,j} \times M_D^{(i,j)}
\end{aligned} \tag{A.74}$$

Substituting (A.74) in (A.73), the mean stall duration is bounded as follows.

Theorem A.19.1 *The mean stall duration time for file i is bounded by*

$$\mathbb{E}\left[\Gamma^{(i)}\right] \leq \frac{1}{t_i} log\left(\sum_{j=1}^{m} \pi_{ij}\left(1 + M_D^{(i,j)}\right)\right) \tag{A.75}$$

for any $t_i > 0$, $\rho_{j,\nu_j,\ell}^{(e)} < 1$, $\rho_{j,\beta_j,\ell}^{(\overline{d})} < 1$, and $\rho_{j,\nu_j,\ell}^{(e)} < 1$ and the involved MGFs exist, $\forall j, \nu_j, \beta_j$.

We note that for the scenario, where the files are downloaded rather than streamed, a metric of interest is the mean download time. This is a special case of our approach

when the number of segments of each video is one, or $L_i = 1$. Thus, the mean download time of the file follows as a special case of Theorem A.19.1.

A.20 Online Algorithm for Edge-cache Placement

We note that for the setup of the edge cache, we assumed that the edge-cache has a capacity of $C_{e,\ell}$ seconds (ignoring the index of the edge cache). However, in the caching policy, we assumed that a file f is removed from the edge cache ℓ if it has not been requested in the last $\omega_{f,\ell}$ seconds. In the optimization, we found the parameters $\omega_{f,\ell}$, such that the cache capacity is exceeded with probability less than ϵ_ℓ. However, this still assumes that it is possible to exceed the cache capacity some times. This is, in practice, not possible. Thus, we will propose a mechanism to adapt the decision obtained by the optimization formulation so as to never exceed the edge cache capacity.

When a file i is requested, the last request of file i is first checked. If it has not been requested in the last $\omega_{i,\ell}$ seconds, it is obtained from the CDN. In order to do that, the space of the file is reserved in the edge-cache. If this reservation exceeds the capacity of the edge-cache, certain files have to be removed. Any file f that has not been requested in the last $\omega_{f,\ell}$ seconds is removed from the cache. If, even after removing these files, the space in the edge-cache is not enough for placing file i in the edge-cache, more files must be removed. Assume that $\mathcal{H}$ is the set contains all files in the edge-cache, and t_{f,l_t} is the last time file f has been requested. Then, if another file needs to be removed to make space for the newly requested file, the file $\text{argmin}_{f\in\mathcal{H}}(t_{f,l_t} + \omega_{f,\ell} - t_i)$ is removed. This continues till there is enough space for the new incoming file. Note that multiple files may be removed to make space for the incoming file, depending on the length of the new file. This is similar concept to LRU where a complete new file is added in the cache, and multiple small files may have to be removed to make space. The key part of the online adaptation so as not to violate the edge-cache capacity constraint is illustrated in Figure A.10. This

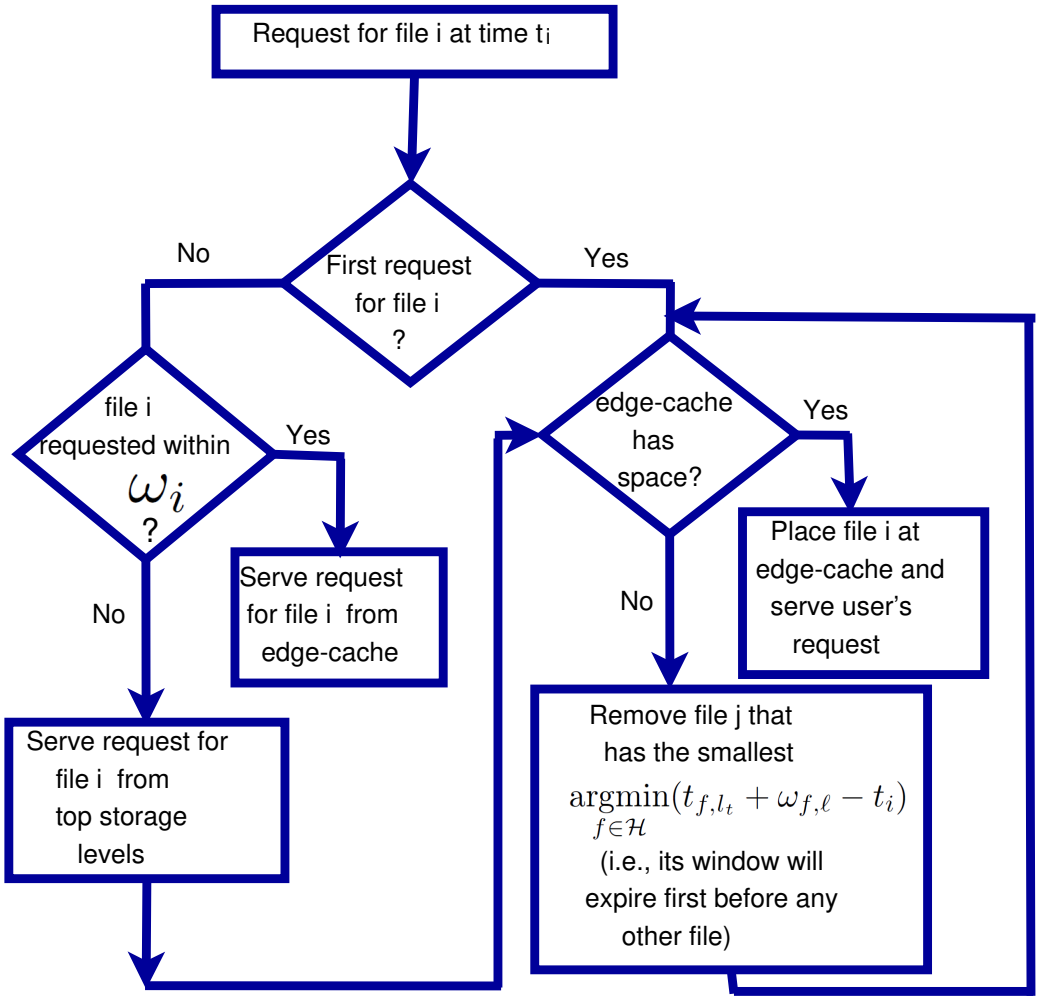

Fig. A.10.: A flowchart illustrates the online updates for an edge-cache when a file i is requested at time t_i. Here, t_{f,l_t} represents the time of the last request of file i, and $\mathcal{H}$ is the index set of all video files in the edge-cache.

flowchart illustrates the online updates for an edge-cache when a file i is requested at time t_i.

A.21 Extension to Different Quality levels

In this section, we show how our analysis can be extended to cover the scenario when the video can be streamed at different quality levels. We assume that each video

file is encoded to different qualities, *i.e.*, $Q \in \{1, 2, \cdots, V\}$, where V is the number of possible choices for the quality level. The L_i chunks of video file i at quality Q are denoted as $G_{i,Q,1}, \cdots, G_{i,Q,L_i}$. We will use a probabilistic quality assignment strategy, where a chunk of quality Q of size a_Q is requested with probability $b_{i,Q}$ for all $Q \in \{1, 2, \cdots, V\}$. We further assume all the chunks of the video are fetched at the same quality level. From Section 5.3.3, we can show that the for a file of quality Q requested from edge-router ℓ, we choose server j with probability $\pi^{(Q)}_{i,j,\ell}$. Further, we can show that the aggregate arrival rate at $PS^{(d,j)}_{\beta_j}$, $PS^{(\overline{d},j)}_{\beta_j,\ell}$, and $PS^{(e,j)}_{\nu_j,\ell}$, denoted as $\Lambda^{(d)}_{j,\beta_j}$, $\Lambda^{(\overline{d})}_{j,\beta_j,\ell}$,and $\Lambda^{(e)}_{j,\nu_j,\ell}$, respectively are given as follows.

$$\Lambda^{(d)}_{j,\beta_j} = \sum_{i=1}^{r}\sum_{Q=1}^{V} \lambda_i \pi^{(Q)}_{i,j} q^{(Q)}_{i,j,\beta_j} b_{i,Q} \tag{A.76}$$

$$\Lambda^{(c)}_{j,\beta_j} = \Lambda^{d}_{j,\beta_j} \tag{A.77}$$

$$\Lambda^{(e)}_{j,\nu_j} = \sum_{i=1}^{r}\sum_{Q=1}^{V} \lambda_i \pi^{(Q)}_{i,j} p^{(Q)}_{i,j,\nu_j} b_{i,Q} \tag{A.78}$$

Similarly, we can define

$$\alpha^{(d,Q)}_{j,\beta_j} = w^{(d)}_{j,\beta_j} \alpha^{(d,Q)}_{j}, \tag{A.79}$$

$$\alpha^{(\overline{d},Q)}_{j,\beta_j,\ell} = w^{(\overline{d})}_{j,\beta_j,\ell} \alpha^{(f_j,Q)}_{j,\ell}, \quad \alpha^{(e,Q)}_{j,\nu_j,\ell} = w^{(e)}_{j,\nu_j,\ell} \alpha^{(f_j,Q)}_{j}, \tag{A.80}$$

for all β_j, ν_j, Q, and ℓ. Note that $\alpha^{(.,Q)}_{j} = \alpha^{(.)}_{j}/a_\ell$ where $alpha^{(.)}_{j}$ is a constant service time parameter when $a_\ell = 1$. We further define the moment generating functions of the service times of $PS^{(d,j,Q)}_{\beta_j}$, $PS^{(\overline{d},j,Q)}_{\beta_j,\ell}$, and $PS^{(e,j,Q)}_{\nu_j,\ell}$ as $M^{(d,Q)}_{j,\beta_j}$, $M^{(\overline{d},Q)}_{j,\beta_j,\ell}$, and $M^{(\overline{d},Q)}_{j,\nu_j,\ell}$, which are defined as follows.

$$M^{(d,Q)}_{j,\beta_j,\ell} = \frac{\alpha^{(d,Q)}_{j,\beta_j} e^{\eta^{(d)}_{j,\beta_j} t}}{\alpha^{(d,Q)}_{j,\beta_j} - t}, \tag{A.81}$$

$$M^{(\overline{d},Q)}_{j,\beta_j,\ell} = \frac{\alpha^{(\overline{d},Q)}_{j,\beta_j,\ell} e^{\eta^{(\overline{d},Q)}_{j,\beta_j,\ell} t}}{\alpha^{(\overline{d},Q)}_{j,\beta_j,\ell} - t}, \tag{A.82}$$

$$M^{(e,Q)}_{j,\nu_j,\ell} = \frac{\alpha^{(e,Q)}_{j,\nu_j,\ell} e^{\eta^{(e,Q)}_{j,\nu_j,\ell} t}}{\alpha^{(e,Q)}_{j,\nu_j,\ell} - t} \tag{A.83}$$

where $\eta_{j,\cdot}^{(\cdot,Q)} = \eta_{j,\cdot}^{(\cdot)} \times a_\ell$ where $eta_{j,\cdot}^{(\cdot)}$ is a constant time shift parameter when $a_\ell = 1$. Following the same analysis as in Section 5.4, it is easy to show that the stall duration for the request of file i at quality Q from β_j queue, ν_j queue and server j, if not in the edge-cache, i.e., $\Gamma_U^{(i,j,\beta_j,\nu_j,Q)}$ is given as

$$\Gamma_U^{(i,j,\beta_j,\nu_j,Q)} = T_{i,j,\beta_j,\nu_j}^{(L_i,Q)} - d_s - (L_i - 1)\,\tau. \tag{A.84}$$

This expression is used to derive tight bounds on the QoE metrics. By simplifications and some algebraic manipulation,the following theorems can be derived.

Theorem A.21.1 *The mean stall duration for video file i streamed with quality Q requested through edge router ℓ is bounded by*

$$\mathbb{E}\left[\Gamma^{(i,\ell,Q)}\right] \leq \frac{1}{g_i} log\left(\sum_{j=1}^{m} \pi_{i,j}^{(\ell)}\left(1 + M_D^{(i,j,\ell)}\right)\right) \tag{A.85}$$

where:

$$\begin{aligned}
M_D^{(i,j,\ell,Q)} = {} & \widetilde{c}_\ell + \widetilde{a}_\ell\, e^{g_i(d_s+(L_i-1)\tau)} + \overline{a}_\ell \times \\
& \sum_{\nu_j=1}^{e_j} p_{i,j,\nu_j,\ell}^{(Q)} \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j,\ell}^{(Q)} e^{g_i L_i \tau} \times \\
& \left(\frac{\widetilde{M}_{j,\nu_j,\ell}^{(e,Q)}(g_i)(1-\rho_{j,\beta_j}^{(e)}) g_i((\widetilde{M}_{j,\nu_j,\ell}^{(e,\cdot,Q)}(g_i))^{L_{j,i}} - 1)}{(g_i - \Lambda_{j,\beta_j}^{(e)}(B_{j,\beta_j,\ell}^{(e)}(g_i) - 1))(\widetilde{M}_{j,\nu_j,\ell}^{(e,Q)}(g_i)) - 1)} \right. \\
& + \frac{(1-\rho_{j,\beta_j}^{(\bar{d})}) g_i (\widetilde{M}_{j,\nu_j,\ell}^{(\bar{d},Q)}(g_i))^{L_{j,i}-L_i}}{g_i - \Lambda_{j,\beta_j}^{(\bar{d})}(B_{j,\beta_j,\ell}^{(\bar{d})}(g_i) - 1)} + \\
& \frac{(1-\rho_{j,\beta_j}^{(d)}) g_i (\widetilde{M}_{j,\beta_j,\ell}^{(\bar{d},Q)}(g_i))^{L_i+1}}{\left[g_i - \Lambda_{j,\beta_j}^{(d)}(B_{j,\beta_j,\ell}^{(d)}(g_i) - 1)\right](\widetilde{M}_{j,\beta_j,\ell}^{(d,Q)}(g_i))^{L_{j,i}+1}} \times \\
& \left(\frac{(\widetilde{M}_{j,\beta_j,\ell}^{(d,\bar{d},Q)}(g_i))^{L_i-L_{j,i}} - (L_i - L_{j,i})}{(\widetilde{M}_{j,\beta_j,\ell}^{(d,\bar{d},Q)}(g_i)) - 1} + \right. \\
& \left. \left. \frac{\widetilde{M}_{j,\beta_j,\ell}^{(d,\bar{d},Q)}(g_i)\left((\widetilde{M}_{j,\beta_j,\ell}^{(d,\bar{d},Q)}(g_i))^{L_i-L_{j,i}-1} - 1\right)}{(\widetilde{M}_{j,\beta_j,\ell}^{(d,\bar{d},Q)}(g_i)) - 1} \right) \right)
\end{aligned} \tag{A.86}$$

and $\widetilde{c}$, $\widetilde{a}$, and $\overline{a}$ are defined earlier in Section 5.4, e.g., equation (5.18).

Theorem A.21.2 *The stall duration tail probability for video file i requested at quality Q and through edge router ℓ is bounded by*

$$Pr\left(\Gamma_{tot}^{(i,\ell,Q)} \geq \sigma\right) \leq$$
$$\sum_{j=1}^{m} \pi_{i,j,\ell}^{(Q)} \times \left[\overline{c}_\ell + \widetilde{a}_\ell\, e^{-h_i\sigma} + \overline{a}_\ell \sum_{\nu_j=1}^{e_j^{(\ell)}} p_{i,j,\nu_j,\ell}^{(Q)} \times \right.$$
$$\left. \sum_{\beta_j=1}^{d_j} q_{i,j,\beta_j,\ell}^{(Q)} e^{h_i L_i \tau} \times \left(\delta^{(e,\ell,Q)} + \delta^{(\overline{d},\ell,Q)} + \delta^{(d,\overline{d},\ell,Q)}\right)\right. \tag{A.87}$$

for $\rho_{j,\beta_j}^{(d)} < 1$, $\rho_{j,\beta_j,\ell}^{(\overline{d})} < 1$, $\rho_{j,\nu_j,\ell}^{(e)} < 1$, where the auxiliary variables in the statement of the Theorem are similarly defined as those in equations (5.22)-(5.33).

Having derived the MSD and SDTP, one can formulate a constrained optimization problem to jointly optimize a convex combination of all QoE metrics as follows

$$\begin{aligned} \min\ & \textstyle\sum_{\ell=1}^{R} \sum_{i=1}^{r} \frac{\lambda_{i,\ell}}{\overline{\lambda}_i} \left[\theta_1 \left(\sum_{Q=1}^{V} -b_{i,\ell} L_i a_\ell\right) + \right. \\ & \left. \textstyle\sum_{Q=1}^{V} \frac{b_{i,Q}}{t_i} \left(\theta_2 \times \Pr(\Gamma^{(i,\ell,Q)} \geq \sigma) + \theta_3 \times \mathbb{E}\left[\Gamma^{(i,\ell,Q)}\right]\right)\right] \end{aligned} \tag{A.88}$$

s.t.

$$(5.47) - (5.51) \tag{A.89}$$

$$\theta_1 + \theta_2 + \theta_3 = 1 \tag{A.90}$$

To solve this porblem, we still have to use alternating minization based approaches since the problem is not jointly convex in all the optimized parametes. Thus, we propose an iterative algorithm (similar to that explained in Appendix A.17) to solve the problem. The proposed algorithm divides the problem into sub-problems that optimizes one variable at a time while fixing the others. We refer the interested reader to [88] for detailed treatment of this problem.

We also note that similar methodology can be used to handle the scenarios where the chunks have different sizes. If the video files have different chunk size (in MB), our analysis can be easily extended to handle such cases as follows. Since a video file has different chunk sizes, the service time will be different from one chunk to another. However, one can still get the MGF of the service time in a similar fashion to those in (A.81)-(A.83). Thus, the service time of a chunk will be related to its size. For example, for a chunk indexed by κ and requested from the PS $PS_{\beta_j}^{(d,j,\kappa)}$, the MGF of the servce time will be $M_{j,\beta_j}^{(d,\kappa)}$. Hence, similar formula to that in (A.85) for the stall duration under different sizes for the chunks can be obtained.

VITA

VITA

Abubakr O. Al-Abbasi received the B.Sc. and M.Sc. degrees in electronics and electrical communications engineering from Cairo University, Cairo, Egypt, in 2010, and 2014, respectively. He is currently pursuing the Ph.D. degree in IE, Purdue University, USA. From 2011 to 2012, he was a Communications and Networks Engineer with Huawei Company. From 2014 to 2016, he was a Research Assistant with Qatar University, Doha, Qatar. His research interests are in the areas of wireless communications and networking, media streaming, machine learning with applications to communications, and signal processing for communications.