Towards cost-driven control and management for cloud centric media network

Jin, Yichao

2015

Jin, Y. (2015). Towards cost-driven control and management for cloud centric media network. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65835>

<https://doi.org/10.32657/10356/65835>

NANYANG TECHNOLOGICAL UNIVERSITY

Towards Cost-driven Control and Management for Cloud Centric Media Network

A thesis submitted to the
Nanyang Technological University

by

Yichao JIN

in fulfillment of the requirement
for the degree of Doctor of Philosophy
of Computer Engineering

December 2015

Abstract

Nowadays, the emergence of cloud computing is transforming the way of operating various IT infrastructures, including video distribution architecture, with an objective to improve their efficiency. Specifically, the cloud-based paradigm enables the video service operators to dynamically leverage virtualized resources, such as bandwidth, computation and storage, so that the user requests can be served in an on-demand manner. As a result, it provides an opportunity to reduce the operational cost of delivering video contents, throughout careful scheduling and management on the usage of cloud resources.

In this thesis, we investigate the cost-efficient control and management problem in cloud centric media network, a cloud based media content distribution paradigm. First, we consider the cost optimization problem for platform services. Specifically, we focus on three cases with different combinations of those functions. The first case stands for a typical content delivery system, where the operators can only utilize the cloud caching resource, and decide how to allocate them for different contents and locations. For this case, we mathematically establish the relation between the replica number and the mean hop distances in large-scale networks, and theoretically derive the optimal replica number for each content with different popularity. The second case introduces online transcoding function to media cloud nodes, making it possible to jointly operate caching and transcoding functions for adaptive video streaming. For this case, we develop an analytical framework to balance a three-way trade-off between storage, computing and bandwidth resources, and derive a closed-form solution for cloud resource allocation. Finally, the last case jointly optimizes not only in-network caching and transcoding function but also the network routing function for each node, enabled by the emergence of Software Defined Networking (SDN) and Network Function Virtualization (NFV). For this case, we propose a two-step iterative approach to find the optimal solution, and develop an analytical framework to prove its convergence and optimality.

Then, we design and implement a multi-screen cloud social TV system as a novel cloud application on top of the cloud centric media network, and try to minimize the operational cost of such service. Specifically, multi-screen cloud social TV uses cloud clone as its key technology, offering a rich set of functions including video transcoding, session synchronizing, and video teleportation. As a result, users can seamlessly transfer ongoing sessions among heterogeneous media outlets instantly. Under this application scenario, we formulate the cost-minimization problem as a Markov Decision Process by adopting a Markov chain to model the user watching behavior across TV and smartphone. The objective is to minimize the monetary cost of operating the video teleportation service, by migrating the cloud clone to the best place, as the user shifts his device. We further propose four algorithms to address this optimization problem.

The aforementioned cost optimization policies, including the optimization processes over both platform and application services, can efficiently reduce the operational cost of distributing media contents. The proposed approaches and obtained results may provide guidelines to improve the efficiency of using cloud resources for video distribution services.

Acknowledgments

This dissertation would not have been possible without a lot of people who helped me and changed my life profoundly during my study at NTU.

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Yonggang Wen. His infectious enthusiasm, unlimited patience, and inspiring guidance have been the major driving force through my research career at NTU. Under his supervision and training, I have developed skills in logical thinking, technical writing, communication skills and leadership, which will be invaluable to my future career.

Beyond my supervisor, I would like to thank Dr. Kyle Guan, Dr. Jianfei Cai, Dr. Cedric Westphal and Dr. Haiyong Xie, who have provided many insightful comments and contributed significantly to my research work. In addition, I also would like to thank in advance the thesis examiners for their comments and suggestions to improve the quality of this thesis.

Moreover, I would thank the students and staffs in CeMNet Annexe at the School of Computer Engineering in Nanyang Technological University. The lab peers, including Weiwen Zhang, Yuan Yuan, Ran Wang, Weida Xu, Peng Sun, Guanyu Gao, Han Hu, Wenfeng Xia, Xiao Liu and Weizheng Hu, have not only provided me with their valuable suggestions for my research, but also enriched my Ph.D. life at NTU with enjoyable experiences. Besides, I would also like to thank the lab staffs for their warmhearted assistants throughout my research activities.

Finally, I cannot end without giving my special thanks to my parents and my wife, for their unselfish love and unconditioned supports.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	ix
List of Tables	xiv
1 Introduction	1
1.1 Background and Motivation	1
1.1.1 Video Consumption Is Dominating Internet Usage	1
1.1.2 Traditional Video Delivery Architecture Is Inefficient	2
1.2 Overview on Cloud Centric Media Network	3
1.3 Research Focus	5
1.4 Organization of the Chapters	6
2 Literature Review	9
2.1 Cost Efficient Content Placement	9
2.2 Cost Efficient In-Network Transcoding	11
2.3 Cost Efficient Orchestration of Caching, Transcoding and Routing	12
2.4 Cost Efficient Cloud Clone Migration for Cloud Social TV	13
2.5 Conclusion	15
3 Cost Efficient Content Placement	17
3.1 Introduction	17
3.2 Problem Modeling & Formulation	19
3.2.1 System Architecture	19
3.2.2 System Models	21

3.2.3	Problem Formulation	22
3.3	Optimal Single Content Placement	24
3.3.1	Ring Graph	25
3.3.2	Petersen Graph & Heawood Graph	27
3.3.3	Generalized Moore graph	31
3.3.4	Random Regular Graph	31
3.3.5	Deployed Networks	33
3.4	Optimal Placement for A Catalog of Content	36
3.4.1	General Solution	36
3.4.2	Optimization without Capacity Constraints	37
3.5	Performance Evaluation	39
3.5.1	Experimental Settings	39
3.5.2	Fundamental Trade-off for Single Content Placement	40
3.5.3	Fundamental Trade-off for A Catalog of Contents	43
3.5.4	Optimal Overall Monetary Cost	43
3.5.5	Optimal Replica Number Distribution	45
3.6	Algorithm Evaluation	47
3.6.1	Heuristics Design	47
3.6.2	Total Cost Comparison	48
3.6.3	Computational Complexity Comparison	50
3.7	Conclusion	51
4	Cost Efficient In-Network Transcoding	53
4.1	Introduction	53
4.2	System Models & Problem Formulation	55
4.2.1	System Architecture	57
4.2.2	Motivating Example	58
4.2.3	System Models	58
4.2.4	Problem Formulation	64
4.3	Analytical Solutions	64
4.3.1	Optimal Transcoding Configuration	64

4.3.2	Optimal Caching Space Allocation	66
4.3.3	Preliminary Theoretical Analysis	67
4.4	Performance Evaluation	69
4.4.1	Experimental Setup	69
4.4.2	Verification	70
4.4.3	Cost Savings	71
4.4.4	Impact of System Parameters	73
4.5	Conclusion	79
5	Cost Efficient Orchestration of Caching, Transcoding and Routing	80
5.1	Introduction	80
5.2	System Overview & Problem Formulation	82
5.2.1	System Architecture	82
5.2.2	System Models	84
5.2.3	Problem Formulation	88
5.3	Analysis & Algorithms	91
5.3.1	Convergence and Optimality	91
5.3.2	Algorithms to the Joint Optimal Solution	92
5.4	Performance Evaluation	94
5.4.1	Parameter Settings	94
5.4.2	Convergence Verification	96
5.4.3	Joint Optimal Solution	97
5.4.4	Performance Comparison	99
5.5	Conclusion	102
6	Cost Efficient Cloud Clone Migration for Cloud Social TV	103
6.1	Introduction	103
6.2	System Overview & Problem Formulation	105
6.2.1	System Implementation	105
6.2.2	System Models	109
6.2.3	Markov Decision Process Formulation	113
6.3	Migration Strategies for Cloud Clone	115

6.3.1	Case Studies for Cloud Clone Placement	115
6.3.2	Random Fixed Placement & Cost Upper Bound	117
6.3.3	Offline Algorithm & Cost Lower Bound	119
6.3.4	Semi-Online Algorithm with Known Transition Model	121
6.3.5	Online Algorithm with Reinforcement Learning	123
6.4	Performance Evaluation	127
6.4.1	Experimental Settings	127
6.4.2	Monetary Cost Evaluation	129
6.4.3	Optimal Migration Policies	135
6.4.4	Comparisons with Existing Methods	137
6.5	Conclusion	138
7	Summary and Future Work	139
7.1	Summary	139
7.1.1	Optimization over Platform Services	139
7.1.2	Optimization for Application Service	141
7.2	Future Work	142
7.2.1	Optimization under Alternative Pricing Models	142
7.2.2	Operation over Wireless Access Network	142
7.2.3	Large-scale Deployment and More Applications	143
A	Cost Efficient Content Placement	144
A.1	Proof of Property 3.1	144
A.2	Proof of Property 3.2	144
B	Cost Efficient In-Network Transcoding	146
B.1	Proof of Theorem 4.1	146
B.2	Proof of Theorem 4.3	147
C	Cost Efficient Orchestration of Caching, Transcoding and Routing	149
C.1	Proof of Lemma 5.1	149
C.2	Proof of Theorem 5.1	151

Abbreviation Table	152
References	153
Publications	163

List of Figures

1.1	A generic architecture of Cloud Centric Media Network.	4
1.2	Main content and organization of the thesis.	7
3.1	Architecture of cloud centric media network. Content providers use the media cloud to distribute UGCs to users as requested. Within the media cloud, the storage resources can be allocated to different edge servers to replicate contents in an on-demand manner.	20
3.2	Demonstration of optimal partition for 3 replicas in ring topology. (a) An example of ring topology with 9 nodes; (b) The root of each partition refers to the node holding the replica, and other nodes forward the content from the root to its users.	25
3.3	Demonstration of optimal partition for 3 replicas in Petersen Graph. (a) Petersen Graph; (b) The root of each partition refers to the node holding the replica, and other nodes forward the content from the root to its users.	27
3.4	Demonstration of optimal partition for 3 replicas in Heawood Graph. (a) Heawood Graph; (b) The root of each partition refers to the node holding the replica, and other nodes forward the content from the root to its users.	28
3.5	Deployed networks: (a) the U.S IP backbone network; (b) the National Science Foundation Network (NSFNET); (c) the hypothetical backbone network US64; (d) the European optical network (EON)	34
3.6	Estimated functions for deployed networks. The obtained logarithmic function is more precise than the existing estimation in all four topologies.	35

3.7	The real-world trace from a video provider. (a) The size of different contents follows Pareto distribution; (b) The popularity of different contents follows Zipf distribution.	39
3.8	Monetary cost for the 10^{th} popular content. The total cost is a convex function of the replica number n in all cases.	41
3.9	Total monetary cost for different content. The overall cost function for the content with different popularity is different.	42
3.10	Total monetary cost under different constraint combinations, decreases as the storage/bandwidth constraint increases until the other constraint becomes active.	44
3.11	The optimal distribution of content replica number follows a power-law distribution in terms of its popularity rank in all four topologies.	46
3.12	Comparisons on the optimal total monetary cost generated by different algorithms, where the bandwidth resource is unconstrained. Our approach can find the near-optimal solution that is very closed to the optimal one, with much lower complexity.	49
4.1	End-to-end view of on-demand adaptive video streaming via media cloud. The end-users consume videos in different versions for different devices. Those video segments are originally published by the origin server. They are transmitted to the users upon request via edge servers, which can strategically cache or transcode video segments to reduce the total cost of operating the adaptive streaming services within the media cloud.	55
4.2	An example scenario of on-demand adaptive video streaming services. The users are requesting three video segments in two different versions, via an edge server that is 3 hops away from the origin server.	59
4.3	Illustration of partial transcoding scheme. Each edge server caches all bitrate versions for top x popular segments, and only caches the highest bitrate version for the rests, subject to the allocated cache space.	61
4.4	Numerical results on the optimal strategy and its operational cost. (a) The combined bandwidth and transcoding cost is a convex function of the transcoding setting x ; (b) The total cost is a convex function of the allocated caching space c	71

4.5	Comparisons on the operational costs under different strategies. Comparing with existing solutions, our derived solution always generates the lowest cost.	72
4.6	Optimal strategy vs. unit transcoding cost w_{tr} . As the unit transcoding cost increases, (a) the total cost increases; (b) the optimal caching space decreases; (c) the optimal transcoding setting linearly increases.	72
4.7	Optimal strategy vs. unit bandwidth cost w_{bw} . As the unit bandwidth cost increases, (a) the total cost increases in a log scale; (b) the optimal caching space increases linearly; (c) the optimal transcoding setting decreases exponentially.	73
4.8	Optimal strategy vs. unit caching cost w_{ca} . As the unit caching cost increases, (a) the total operational cost increases in a log scale; (b) the optimal caching space decreases exponentially; (c) the optimal transcoding setting remains the same.	75
4.9	Optimal strategy vs. hop distance d_i . As the hop distance to the origin server increases, (a) the total cost increases; (b) the optimal caching space increases linearly; (c) the optimal transcoding setting decreases exponentially.	75
4.10	Optimal strategy vs. Zipf exponent α . As the Zipf exponent increases, (a) the total cost decreases exponentially; (b) the optimal caching space decreases exponentially; (c) the optimal transcoding setting increases linearly.	77
5.1	Architecture of NFV and SDN enabled media cloud. Various network functions including in-network caching and transcoding can be virtualized and dynamically operated among different nodes within the media cloud. In addition, SDN further enables a complete control over the routing matrix of the whole network.	83
5.2	Network topologies used in the simulation, including (a) synthetic 10-node ring, (b) real Internet2's layer 3 network in the U.S specifically the future Internet	95
5.3	Convergence performance of our iterative approach. All the testing cases converge to the optimal solution within 7 iterations.	97

5.4	Optimal virtualized resource allocation. More caching or transcoding resources should be allocated to the node with heavier user request rate, while the two resources are compensatory to each other.	98
5.5	Optimal network routing policy: for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path if there is only one.	99
5.6	Total networking cost vs. cost function $f(u_e) = nu_e^n$. The joint optimization solution performs better than those baselines in all simulation cases.	100
5.7	Total networking cost vs. Zipf exponent α . The joint optimization solution performs better than those baselines in all simulation cases.	101
6.1	Multi-screen cloud social TV architecture. End users with different devices are connected via residential gateways and access networks to the cloud, which offers cloud clone based virtual overlay content delivery services to distribute live or on-demand TV programs. Such service can be dynamically operated in terms of creating/halting/migration cloud clones.	107
6.2	System overview on cloud clone. It serves as the proxy for each user in the cloud, to manage all the associated devices and real-time session information.	109
6.3	User behavior on session migration by using video teleportation. (a) Real user case of multi-screen cloud social TV; (b) Device switching model as a Markov chain.	111
6.4	Three case studies on cloud clone migration.	116
6.5	An example of the state transition of offline algorithm, where all the previous and future user behaviors are deterministic.	120
6.6	Systematic illustration of the state transition for semi-online algorithm with known transition probability.	122
6.7	Systematic model of online algorithm with reinforcement learning. The learning agent receives real user behaviors and the costs of operating cloud clone are two kinds of input at each time slot, tunes the trained user model, and makes migration decision.	124
6.8	Minimal monetary cost vs. delivery path (simulated user behaviors). . . .	130

6.9	Minimal monetary cost vs. delivery path (real traces).	131
6.10	Minimal monetary cost vs. migration size (simulated user behaviors) . .	132
6.11	Minimal monetary cost vs. migration size (real traces)	132
6.12	Minimal monetary cost vs. simulated user behavior patterns	134
6.13	Examples of the migration scheduling based on Q-learning method. . . .	135
6.14	Comparisons between Q-learning algorithm and existing methods. Q- learning method outperforms the existing ones in all the cases.	137

List of Tables

2.1	Comparisons among different previous works	16
3.1	Notation Table	19
3.2	Comparison on theoretical and real diameters for different topologies . .	36
3.3	The time used to compute optimal replica number (s)	51
4.1	Notation table	56
4.2	Comparisons on different strategies in the motivation example	59
4.3	Bitrate levels in real adaptive streaming system	69
5.1	Notation table	85
5.2	Bitrate versions in real adaptive streaming system	95
6.1	Notation Table	106
6.2	Experimental Parameter Settings	129

Chapter 1

Introduction

In this chapter, we first present the background and challenges of multimedia content distribution over Internet. We then introduce Cloud Centric Media Network, which has been proposed as a novel cloud based architecture to address these challenges. On top of the cloud centric media network, we illustrate the objective and the research interests. Finally, we point out the key contributions of the thesis.

1.1 Background and Motivation

1.1.1 Video Consumption Is Dominating Internet Usage

Recently, owing to the ubiquitous penetration of high speed Internet access and the increasing popularity of multimedia applications, more and more video contents are created, distributed, and consumed everyday. Specifically, according to Cisco's VNI report [1], video streaming traffic contributed 66% to the total Internet traffic in 2013. In addition, this report further predicts that this number will account for 80-90 percent by 2018, completely dominating the Internet usage. This means, in every single second, there will be nearly a million minutes of video contents crossing the global network by that time. More importantly, these numbers are still expecting to continue growing afterwards, fueling by the advancement of media technology and the strong demands of the market.

The exploding growth of video distribution and consumption poses significant stress to the existing video delivery architecture. In particular, first, unlike plain text and images, video contents have much bigger size, requiring much more storage and bandwidth resources to be stored and transmitted. Second, the unique features of video distribution, such as adaptive video streaming, ask for additional computing resources within the network. Last but not least, different multimedia applications in the user-end come up with a wide variety of video consumption styles, exposing high dynamics to the networking architecture. As a result, it is necessary to thoroughly re-examine whether the current video distribution framework is still capable of appropriately handling with these issues.

1.1.2 Traditional Video Delivery Architecture Is Inefficient

Unfortunately, the traditional video delivery architecture is not efficient enough to distribute the huge amount of video contents. In particular, it suffers from a list of problems as followings.

- Resources cannot be dynamically utilized in response to the changes of user request patterns. For example, although caching contents in the content distribution network can save about 30% of operational cost [2], existing CDN solutions are mostly built upon dedicated hardware, which have to be designed for peak time traffic. As a result, there is no way to utilize the spare resources, when the number of user requests goes down. In this case, those resources are simply wasted, leaving significant spaces for further improvement.
- Existing solution did not take the characteristics of adaptive video streaming services into account. Adaptive video streaming has become the de-facto standard to distribute video contents to heterogeneous end-devices, where the video will be pre-transcoded into multiple representations in the origin server. Although this

solution is very successful in reducing bandwidth consumption and improving end user experiences [3], it also diminishes the effectiveness of caching function in the traditional content delivery solution. In other words, to get the same cache-hit ratio using a single encoding, the cache size must scale up to include all possible representations.

- It is impossible to capture a global view of networking status under the existing Internet scheme, leading to the lacks of orchestration over routing and resource allocation. In particular, appropriate routing scheme can lead to significant operational cost saving [4]. However, existing network elements (e.g., routers and switches) are working in a highly distributed manner. Each node only exchanges the information with its neighbors, without a centralized controller or mechanism to manage a global view. As a result, it could be inefficient to achieve the global optimization jointly over dynamic routing and resource allocation.
- There is no way to tune the network operation models for novel multimedia applications. The emergence of new applications always comes up with novel features which requires intensive in-network resources and careful scheduling on them. For example, to efficiently operate the inter-screen video transportation in multi-screen applications, one must dynamically change the operational strategy when users switch their end-devices (e.g., between a large screen and a small screen). Nevertheless, giving the aforementioned drawbacks of the existing mechanism, it is very difficult to elastically adjust the operational schemes.

1.2 Overview on Cloud Centric Media Network

Cloud computing has been emerged as a promising technology to transform a large part of the IT industry. It allows the service operators to leverage virtualized resources, including

Figure 1.1: A generic architecture of Cloud Centric Media Network.

bandwidth, computation and storage in an on-demand manner. More specifically, by virtualization technology, resources can be dynamically allocated from a shared pool to different virtual machines to meet the user demand.

Owing to the elastic nature of cloud computing, we introduce cloud computing to media content distribution services, proposing a novel Cloud Centric Media Network to address those issues of traditional architecture. As shown in Figure 1.1, it is built over a hybrid media cloud infrastructure to transmit media contents from content providers to end-users. The media cloud consists of a layered service model, including,

- *Infrastructure Layer* consists of a set of interconnected data centers, forming an un-configured content distribution overlay with an underlying network infrastructure. Each media data center exposes a pool of IT resources, including computing, storage, and bandwidth to media service operators via resource virtualization techniques. For example, physical servers, enabled by supervisors (e.g., VMware Server, Xen, Microsoft Hyper-V, etc), can be abstracted into a pool of virtual machines (VM). Each VM is an instantiation of a logical server, which runs its own guest

operating system and media applications (e.g., streaming engines or transcoding engines) powered by networking function virtualization (NFV) [5]. Besides, the whole network can also be centralized managed via emerging SDN (or open-flow) technology [6].

- *Platform Layer* offers a solution stack as a service, through a proprietary inter-cloud messaging bus protocol. Specifically, this layer aims to provide a wide variety of tools, libraries, and interfaces to elastically orchestrate multiple resources in accordance to different applications. As a result, the offerings would facilitate the deployment of the media cloud on resource allocation, virtual machine management, and media service discovery, etc.
- *Application Layer* instantiates various novel multimedia applications as a service, where the software workflow, required resource, and the major associated data are hosted in the cloud, whereas the end-users only need a thin client.

1.3 Research Focus

The objective of this thesis is to efficiently orchestrate multiple resources in Cloud Centric Media Networks for different multimedia applications. Specifically, we aim to develop theoretical frameworks and propose corresponding algorithms, to optimally allocate virtualized resources (i.e., storage, computing, and bandwidth resources) in various combinations, so that the total cost of operating Cloud Centric Media Network can be minimized. In particular, we focus on the following scenarios.

- We first consider an existing content delivery network scenario, which caches multiple copies of video contents at the edge of Cloud Centric Media Network. In this case, we focus on the problem of deciding the number of the replicas for each content and their locations.

- In addition to cache function, we introduce transcoding function to deal with the distribution of adaptive video streaming over Cloud Centric Media Network. Under this scenario, we propose a partial in-network transcoding scheme, aiming to jointly optimize the virtualized caching and transcoding functions at each cloud node.
- We further adopt the concept of both NFV (Network Function Virtualization) and SDN (Software Defined Network), which are promising technologies in future Internet. As a result, the operator can not only adjust the caching and transcoding functions at each node, but also adopt multi-path routing scheme to further improve the performance and cost of operating Cloud Centric Media Network.
- Finally, we develop multi-screen cloud social TV as a proof-of-concept application of Cloud Centric Media Network, and propose the optimal cloud clone migration scheme to efficiently operate it. This may provide a state-of-art example of using Cloud Centric Media Network to facilitate the emergence of novel multimedia applications.

1.4 Organization of the Chapters

Figure 1.2 shows the main content and organization of the thesis. Specifically, we can divide the main content into two parts, including theory and application. The first half of the thesis (Chapter 3, 4 and 5) considers the cost optimization in the layer of platform service. Specifically, in Chapter 3, we investigate the optimization problem that the Cloud Centric Media Network operators can only decide the content placement scheme. Next, in Chapter 4, we introduce online transcoding function to media cloud nodes, making it possible to jointly operate caching and transcoding functions to minimize the total cost. Finally, in Chapter 5, we present the joint optimization over not only in-network caching and transcoding function but also the network routing function for each

Figure 1.2: Main content and organization of the thesis.

node. The second half (Chapter 6) considers the cost optimization problem in the layer of application service. Specifically, we focus on the optimization for our proof-of-concept application (i.e., cloud social TV) by strategically migrating cloud clones to different locations.

In summary, the rest of this thesis is organized as follows:

- **Chapter 2** introduces related works on cost optimization in the context of media cloud computing.
- **Chapter 3** aims to minimize the monetary cost of operating Cloud Centric Media Network by strategically decide the number of replicas and their locations for each content.
- **Chapter 4** aims to minimize the monetary cost of operating Cloud Centric Media Network by additionally introducing in-network function, and strategically coordinating it with in-network caching.

- **Chapter 5** aims to minimize the monetary cost of operating Cloud Centric Media Network by jointly orchestrating multiple networking functions, including caching, transcoding and routing.
- **Chapter 6** aims to minimize the monetary cost of operating cloud social TV as a proof-of-concept application in Cloud Centric Media Network, by optimally scheduling cloud clone migration.
- **Chapter 7** summarizes the thesis and discusses the further work for cloud centric media network.

The next chapter provides literature review of the research in this thesis.

Chapter 2

Literature Review

In this chapter, we provide a literature review of efficiently operating video distribution network with different decision variables in different scenarios. We first give an overview of the platform services (i.e., content placement, in-network transcoding, and service orchestration) for traditional video delivery services. We then review existing works of cloud social TV applications, and its optimization technologies.

2.1 Cost Efficient Content Placement

The optimal content placement problem, due to its complexity, was addressed by using heuristics in most existing works. Qiu *et al.* [7] focused on selecting M replicas among N different potential locations to minimize the bandwidth cost, and investigated the performances of several different algorithms. Kangasharju *et al.* [8] studied the object replication strategies on CDN nodes with finite storage capacity by developing four natural heuristics. Karlsson [9] formulated the minimal replication cost problem as an integer programming problem, and proposed a heuristic method for choosing a replica placement. Tang *et al.* [10] investigated this problem by using several heuristic algorithms, for QoS-aware content distribution. Differed from these works, we formulate a convex optimization problem to theoretically derive the optimal policy and obtain analytical

insights. In [11], the authors introduced a distributed collaborative caching mechanism with focus on exploring the constraints of existing IPTV structure.

Some other researches formulated the content placement problem in specific or arbitrary network topologies as a classic graph theory problem, aiming to derive an analytical solution. For specific network topologies, the analytical solutions on optimal content placement for ring based and tree based CDN were discussed by [12] and [13] respectively. For arbitrary network topologies, this problem could be attributed to the k -center problem [14]. Baev *et al.* [15] proposed an approximation algorithm by using the integer linear programming, while considering the storage constraint to place content in arbitrary networks. In [15], the content placement problem in P2P system was formulated as a clustered k -center problem. Laoutaris *et al.* [17] considered both k -center problem and facility location problem for optimal content placement in large-scale networks. However, they mainly focused on how to select the optimal locations for a fixed number of replica, whereas we aim to analytically find the relationship between the replica number and the mean hop distance from any of those replicas to the users. In addition, these works either solved an unconstrained problem [7, 12], or missed one of the resource constraints [8, 17], whereas we address this optimization problem by jointly considering the storage and bandwidth capacity constraint.

However, none of these pervious researches can be directly applied to the monetary cost effective content placement problem in CCMN. Our work in Chapter 3 differs from them in several aspects. First, we analytically find the relationship between the replica number and the mean hop distance from any of those replicas to the users, whereas most existing works [7, 8, 16] mainly focused on how to select the optimal locations for a fixed number of replica. Second, we formulate the content placement problem into a convex optimization problem with a design objective to minimize the monetary cost. Our approach analytically finds the optimal content placement policy on more realistic network topologies (e.g., mesh network), whereas the existing works [7–10] usually adopted

heuristic methods. Third, we address this optimization problem by jointly considering the storage and bandwidth capacity constraints, whereas the previous works either solved an unconstrained problem [7, 12], or missed one of the resource constraints [8, 17].

2.2 Cost Efficient In-Network Transcoding

Media cloud, as an emerging paradigm to efficiently distribute video streaming services, has attracted significant attention. Zhu *et al.* [18] proposed a media-edge cloud architecture, in which storage and computing resources are allocated at the edge of media cloud, to perform user-centric video processing and QoS (Quality of Service) adaptation for different end-devices. Díaz-Sánchez *et al.* [19] designed the media cloud as an open middleware for WAN (Wide Area Network) scale content management and delivery. Wang *et al.* [20] presented Cloud-Assisted Live Media Streaming (CALMS) as a generic cloud-based framework to facilitate migrating live media streaming from the origin server to the media cloud. Wang *et al.* [21] discussed a optimal cloud-assisted adaptive mobile video streaming scheme, that different video versions are elastically pre-fetched and stored within the media cloud. More recently, Wen *et al.* [22] summarized these works on video streaming over media cloud, together with a line of key design principles, into a unified layered service framework.

The orchestration on multiple resources, as one of the key design considerations in media cloud, also had been intensively investigated in many other application scenarios. Qazi *et al.* [23] introduced resource management module to balance the computing resource consumption at each middlebox, in the context of software defined networking. Grandl *et al.* [24] orchestrated multiple resources via a heuristic packing approach for cluster scheduler, to optimize the job completion time. Zhou *et al.* [25] developed a blind resource scheduling algorithm for mobile media cloud, by formulating a finite time horizon optimization problem and statically routing user requests to appropriate media

service nodes. Wu *et al.* [26] adopted on-demand cloud resource provisioning to desirably meet the dynamic and intensive resource demands of video on-demand services, by focusing on the virtual machine usage and storage space rental.

Chapter 4 clearly differs from those related researches mainly in two ways. First, we focus on multi-resource orchestration in the media cloud, with the objective to optimize the total operational cost incurred by delivering on-demand adaptive video streaming. The context and the objective of such orchestration problem are different from those existing researches. Second, we develop an analytical framework to balance the three-way trade-off between storage, computing and bandwidth resources, and derive a closed-form solution. In contrast, most previous works only focused on one or two resource components. Besides, few of them came up with closed-form solutions. To the best of our knowledge, this work is the first attempt to systematically examine the optimal multiple resource scheduling for efficient adaptive video streaming distribution over the media cloud.

2.3 Cost Efficient Orchestration of Caching, Transcoding and Routing

We had witnessed the leverage of SDN to optimize the network routing in a diversity of networking applications over the past few years. In particular, McKeown *et al.* [27] initially proposed OpenFlow and SDN to decouple network control and data forwarding for research innovation in campus networks. Lately, Egilmez *et al.* [28] introduced OpenFlow controller to dynamically re-route scalable video streaming with the objective to improve the PSNR (Peak Signal-to-Noise Ratio). Agarwal *et al.* [29] formulated a dynamic routing problem based on traffic engineering to minimize the maximum utilization of the links in the network. Keller *et al.* [30] used SDN to efficiently manage the consistency for live virtual machine migration in cloud computing environment. Arefin *et al.* [31]

achieved a SDN-based cross-layer multi-stream multi-site protocol for 3D teleimmersion to improve the application interactivity and resource utilization. Li *et al.* [32] explored SDN to schedule the exclusive data flow in the data center network in an energy-aware manner.

At the same time, a number of works focused on improving the performance or efficiency of video distribution, by carefully scheduling network functions (e.g., caching and transcoding). Although none of them were specific to the NFV framework, their solutions were closely related to the NFV-based resource allocation. Specifically, Dai *et al.* [33] developed a hierarchical caching framework coupling with dynamic request routing to improve the caching and bandwidth utilization. Li *et al.* [34] investigated the optimal in-network caching space provision for each individual router, to optimize the overall networking cost. Wang *et al.* [35] jointly studied video transcoding and delivery for adaptive video streaming, aiming at reducing both computing and storage costs.

Chapter 5 provides some unique contributions in following aspects. First, we investigate the cost-efficient video distribution in SDN and NFV enabled media cloud, where the application scenario is obviously different from the one of those previous works. Second, we focus on the joint optimization problem of NFV-based virtualized resource allocation and SDN-based routing, whereas most existing works addressed only one of them. Finally, we propose a two-step iterative approach to find the optimal solution, and develop an analytical framework to prove its convergence and optimality. To the best of our knowledge, this work is the first attempt to systematically introduce SDN and NFV to orchestrate adaptive video streaming services in cloud computing environment.

2.4 Cost Efficient Cloud Clone Migration for Cloud Social TV

Recently, an increasing number of novel systems are proposed to provide multi-screen experiences with different enabling technologies. Deep shot [36] was proposed as a general

framework to transfer ongoing sessions across multiple screens. *Lu et al.* [37] presented a virtualized screen solution, which used a cloud clone scheme to complete screen rendering in the cloud. In this way, the screen display can be delivered as a series of images to the thin clients. *Wu et al.* [38] described their design on a cloud based mobile social TV system. Its key module was a virtual machine based surrogate in the cloud, which provided video transcoding services and segmenting the streaming traffic for each user. *Wang et al.* [39] adopted a similar scheme to instantiate a private agent in the cloud for each user, providing adaptive video streaming and social networking services.

Adaptive video streaming, due to its high practical values to efficiently distribute video contents to different end-users, has become an active research topic. *Bernaschi et al.* [40] presented a transcoding-based adaptive streaming scheme for heterogeneous networks. They implemented a testbed to show their solution is practical and efficient in real scenarios. *Shen et al.* [41] proposed a transcoding-enabled caching scheme for content distribution networks, where the content adaptation is performed at the network intermediaries. In [42], the authors discussed the techniques for media and streaming strategy adaptation, and indicated placing transcoding service in proxy servers as one of the typical streaming systems.

Similar cost minimization problems on cloud based systems have been addressed by several previous researches. *Armbrust et al.* [43] listed the data transfer cost as one of the top obstacles and opportunities for cloud computing. It indicated that, an appropriate virtual machine placement policy is important to reduce the operational cost. *Jaime et al.* [44] examined the energy tradeoff between video transportation and processing for view video streaming, and obtain the optimal video processing location with the lowest energy cost. *I. Zhovnirofsky et al.* [45] described a performance enhancement (PE) scheme, where each application is supported by a PE. Each PE locates at one server, and it can be turned on or off on demand. In this way, this mechanism aims to improve the quality of service and reduce the cost.

Chapter 6 differs from these researches in several aspects. First, we built CCMN as a novel framework to integrate a variety of media services, and elastically provide them to users. Second, our cloud clone offers the richest set of functions, including video transcoding, session synchronizing, and video teleportation. As a result, users can seamlessly transfer ongoing sessions among heterogeneous media outlets instantly. Third, none of those works aimed to improve the system by optimizing the operational cost, while we focus on a cost minimization problem by finding the optimal location of cloud clone migration. Finally, those works did not take multi-screen experiences into account, and they mostly aimed at an optimal fixed location for each virtual machine. Whereas we need an optimal dynamic cloud clone migration scheme with online scheduling.

2.5 Conclusion

In this chapter, we presented the literature survey on efficient video distribution scheduling with different decision variables in different scenarios. To better understand the background of this thesis, we summarize both the contribution and the limitation of these previous works into Table 2.1. Motivated by those open issues presented in the table, we will present our research details on how to addressing them in the following chapters.

Table 2.1: Comparisons among different previous works

Research Area	References	Approaches	Open Issues
Content placement	[7], [8], [9], [10]	Heuristic algorithms	No analytic solutions
	[12], [13], [14], [15], [15], [17]	Graph theory	Unrealistic topology, not for real network with multi resource constraints
Content transcoding and caching	[18], [19], [20], [21]	Pre-fetching and pre-transcoding	No orchestration on different resources
	[23], [24], [25], [26]	Multi-resources allocation	No closed-form solution for the optimal decision
NFV and SDN solution	[27], [28], [29], [30], [31], [32]	SDN-enabled packet routing	No joint optimization over both SDN-enabled routing and NFV-enabled function scheduling
	[33], [34], [35]	NFV-enabled network function scheduling	
Operation tuning for cloud-based multi-screen applications	[36], [37], [38], [39]	Multiscreen system implementation	Lack of operation model tuning
	[43], [44], [45]	Cloud-enabled operation model tuning	Not applicable for cloud-based multi-screen system

Chapter 3

Cost Efficient Content Placement

3.1 Introduction

Content placement problem is one of the critical issues to minimize the monetary cost incurred by using cloud resources to orchestrate elastic content delivery services in Cloud Centric Media Network. In particular, building CDN services in the cloud has been touted as a novel and efficient approach to reduce the operational cost [46]. However, a valid cloud CDN solution requires a careful evaluation on the trade-off between the bandwidth and storage resources. On one hand, placing a large number of content replicas at different geo-locations would increase storage-related cost [47]. But it will bring the content closer to the end users, thus reducing the delay and saving the bandwidth usage. On the other hand, the planners can choose to deploy fewer replicas to save the storage resources, albeit at the expense of increased delays and bandwidth costs. Therefore, a cost-effective content placement policy that balances the trade-off between storage and bandwidth, needs to be thoroughly studied.

In this chapter, we formulate this content management problem as a constrained optimization problem. The objective is to minimize the monetary investment associated with the usage of bandwidth, storage and computation resources. This problem is solved in two steps. First, for a single content, we map it into the k -center problem [14] to explicitly derive the optimal placement policy for a given number of replica, on deployed

networks. Second, for a catalog of contents, the problem is formulated as a convex optimization problem. By proposing an algorithm to solve this problem, we obtain the optimal replica number for each content title.

The contributions of this chapter are multi-fold:

- We mathematically present the bandwidth cost and the storage cost as two competing factors for content placement problem in the context of Cloud Centric Media Network. It introduces a fundamental trade-off to the design of cost effective content placement strategy.
- Our solution on single content problem suggests, assuming a uniform request pattern, the mean hop distance H between users and their closest replica, is a logarithmic function of the reciprocal of replica number n (i.e., $H \sim \log(N/n)$) in a set of deployed networks with N nodes. This logarithmic relationship represents the fundamental trade-off between storage and bandwidth resources, offering an analytical framework to evaluate the cost associated with different placement strategies.
- The numerical verifications on real traces suggest, under feasible bandwidth and storage constraints, the optimal replica number follows a power-law distribution to its popularity in a set of deployed networks. This offers guidelines for the real-word operations in cloud CDN environment.

The reminder of the chapter is organized as follows. In Section 6.2, we present the system models and the problem formulation. In Section 3.3, we characterize the k -center problem for single content placement. In Section 3.4, we solve the convex optimization problem for a catalog of contents. In Section 6.4, we verify the obtained solutions using real traces. In Section 3.6, we compare the performance of our approach with a set of heuristic algorithms. In Section 7.2, we conclude this chapter.

Table 3.1: Notation Table

Symbol	Definition
n_k	The number of replica for content k
B_k	The size of content k
R_k	The downloaded time of content k over t period
G	The topology of underlying networks
c_{tr}	Per hop cost for transmitting per bit data
c_{st}	Per bit cost for storing content
Δ	The node degree of a graph
M	The total number of unique contents to be delivered
N	The total number of datacenters
H	The mean hop distance between users and replicas
S_{tot}	The total storage capacity constraint
T_{tot}	The total bandwidth capacity constraint

3.2 Problem Modeling & Formulation

In this section, we first describe the system architecture to provide necessary background. Then, we present key system models for Cloud Centric Media Network. Finally, using these models, we formulate the content placement problem as a constrained optimization problem. For clarity in the discussion, we summarize the important notations in table 6.1.

3.2.1 System Architecture

Figure 3.1 shows a reference architecture of traditional cloud CDN scenario, which consists of three parts, including cloud service provider, content service providers, and a list of content consumers. The cloud providers own a list of virtual machines (VMs), which are carved out in geographical distributed data centers. Those VMs work collaboratively to build an elastic CDN overlay on top of its underlying physical networks. The content service providers usually offer users a platform to publish their own generated contents. Typically, those contents are stored and processed in origin servers, with limited physical resources. Due to the enormous growth of UGCs, CDN services would be necessary to

Figure 3.1: Architecture of cloud centric media network. Content providers use the media cloud to distribute UGCs to users as requested. Within the media cloud, the storage resources can be allocated to different edge servers to replicate contents in an on-demand manner.

maintain the video distribution at a certain QoS level. However, it is not easy for them, specially small companies, to establish their own CDNs, or sign long-term contract with large CDN companies. To address this gap, in Cloud Centric Media Network, they are encouraged to use cloud resources, to achieve content replication and media streaming services, on a pay-per-use model. As a result, the content consumers from different regions globally can be served at a lower price.

How to minimize the monetary cost incurred by using cloud resources to efficiently serve user requests, is one of the critical design objectives for Cloud Centric Media Network. Indeed, the monetary cost highly depends on the content placement policy, which decides the number of replica for each content and their locations. Specifically, because the storage cost is usually a little cheaper than the bandwidth cost, it is advantageous to replicate content to different places, so that the average distance between users and contents, and the bandwidth cost can be reduced. However, if too many replicas are

placed and their locations are not properly chosen, especially for those large amount of unpopular contents, it results in significant storage usage and limited reduction on bandwidth usage. Therefore, an optimal placement policy should be in place, to balance the trade-off between bandwidth and storage cost.

3.2.2 System Models

In this subsection, we present three system models, including a topology model, a content model and a user request model for the problem formulation.

3.2.2.1 Topology Model

The underlying physical infrastructure can be viewed as a connected graph G by modeling N data centers as the nodes and the network connectivities as the edges. Each node is associated with storage and computation resources, and each link is associated with bandwidth resources. All those resources are finite in Cloud Centric Media Network. This work will use four well-known deployed backbone topologies as shown in section 3.3.5 as examples.

3.2.2.2 Content Model

The content model defines the basic properties of a catalog of M unique contents to be delivered in a time window t . For a specific content k , where $k = 1, \dots, M$ is sorted by from the most popular one to the least popular one, both its size B_k and its popularity r_k are random variables.

In particular, the content size follows bounded Pareto distribution [48] with the cumulative distribution function as,

$$F(x) = \frac{1 - B_L^\alpha x^{-\alpha}}{1 - (\frac{B_L}{B_U})^\alpha}, B_L \leq x \leq B_U, \quad (3.1)$$

where B_L and B_U are the smallest and largest content size, and α is the shape parameter having $\alpha > 0$. The larger α is, the smaller the percentage of large files are.

The popularity r_k can be measured by the download times R_k in a period t . It follows Zipf-like distribution [49, 50] as,

$$r_k = \frac{R_k}{R_{tot}} = \frac{k^{-\beta}}{\sum_{s=1}^M s^{-\beta}}, \quad (3.2)$$

where $R_{tot} = \sum_{k=1}^M R_k$ is the total amount of requests for all the contents, and β is the shape parameter of Zipf distribution. The large β indicates the relatively small percentage of very popular ones. The two distributions will be verified by real traces in section 6.4.

3.2.2.3 User Request Model

Each site in Cloud Centric Media Network is also a service entry point attached with an access network. Each user request is served by the nearest node which holds a replica of the requested content. We assume a uniform traffic pattern, that the traffic load from each entry point is almost the same in a given time window t . This assumption could be reasonable, because when deploying CDN sites, a fundamental principle is to balance the traffic among them, so that the performance can be optimized [51]. In other words, more sites are available for the locations with more user requests, resulting in a uniform traffic pattern. For example, in the US IP backbone network and U.S 64 backbone network [52], the density of nodes beyond the west and east coast of the U.S. is much higher than the one in the middle area.

3.2.3 Problem Formulation

The objective is to minimize the monetary cost associated with the usage of bandwidth, storage and computation resources, which depends on the content placement strategy. Specifically, the storage cost depends on the content size and the replica number. The bandwidth cost depends on the hop distance from users to replicas. And the computation cost is incurred by processing each user request, which is independent with the replica

number. As a result, in this chapter, we only focus on the storage and bandwidth cost as follows.

Storage cost for content k over a time duration t is a function of its number of replicas n_k as,

$$C_{st}^k(n_k) = c_{st} B_k n_k t, \quad (3.3)$$

where B_k is the content size, c_{st} is the per-bit cost.

Bandwidth cost for content k with B_k size and R_k download times over the same time duration t is a function of its replica number n_k and the underlying topology G , as,

$$C_{tr}^k(n_k, G) = R_k B_k c_{tr} H(n_k, G), \quad (3.4)$$

where c_{tr} is per hop cost for transmitting per bit data, $H(n_k, G)$ is the mean hop distance between an user and the closest replica of content k .

Based on the system models and our assumptions, we formulate the content placement problem as a constrained optimization problem, with an objective to minimize the combined storage and networking cost as,

$$\min_{\mathbf{n}} \quad f(\mathbf{n}) = c_{st} \mathbf{B}^T \mathbf{n} t + c_{tr} \mathbf{B}^T \mathbf{H} \mathbf{R} \quad (3.5)$$

$$\text{s.t.} \quad g_i(\mathbf{n}) = 1 - n_i \leq 0, \quad \text{for } i = 1, \dots, M, \quad (3.6)$$

$$g_{i+M}(\mathbf{n}) = n_i - N \leq 0, \quad \text{for } i = 1, \dots, M, \quad (3.7)$$

$$g_{2M+1}(\mathbf{n}) = \mathbf{B}^T \mathbf{n} - S_{tot} \leq 0, \quad (3.8)$$

$$g_{2M+2}(\mathbf{n}) = \mathbf{B}^T \mathbf{H} \mathbf{R} - T_{tot} \leq 0, \quad (3.9)$$

where M is the number of unique contents, N is the number of data centers, $\mathbf{H} = \text{diag}(H(n_1, G), \dots, H(n_M, G))$, $\mathbf{n} = (n_1, \dots, n_M)$, $\mathbf{B} = (B_1, \dots, B_M)^T$, and $\mathbf{R} = (R_1, \dots, R_M)^T$.

The constraint (3.6) indicates that there must be at least one version of each content stored by the origin server within the media cloud, and (3.7) indicates that the replica

number cannot exceed the number of all cloud nodes. The constraints (3.8) and (3.9) capture the total storage and bandwidth capacity limitations respectively. Note that, in this formulation, we relax the integer constraint of n_k (i.e., n_k can be decimal, satisfying $n_k \in [1, N]$), for a lower bound solution. This simplification can be reasonable because the video contents are usually sliced into a set of small segments, allowing the cloud nodes to cache a fraction of a content. Besides, we also do not consider the initial cost of placing the replica from the origin server to the cloud node, because for each content, such cost is only incurred for the first time.

We will adopt a two-step method. First, for a single content k , we will find the function of optimal mean hop distance H in terms of its replica number n_k in section 3.3. Second, we will use the obtained result to address the optimization problem for a catalog of content in section 3.4.

3.3 Optimal Single Content Placement

In this section, we cast the single content placement as a variant of k -center problem [14] as follows. Given a network topology $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of data centers that $N = |\mathbf{V}|$, and $\mathbf{E}$ is the set of undirected links between them. If n ($n \leq N$) replicas of a content item are cached at a set of V_R different nodes ($n = |V_R|$), other $N-n$ nodes have to access a content copy from the nearest hosting node via one or more hops. We aim to find the relationship between the replica number n and the average hop distance H for an user. Here we formally define H as,

$$H = \frac{1}{N} \sum_{i \in V} \min_{j \in V_R} h_{ij}, \quad (3.10)$$

where h_{ij} is the shortest path from node $i \in V$ to node $j \in V_R$.

Due to the NP-completeness of this problem [14], the complexity of directly solving it can be prohibitive, especially when the size of network is large. We thus look into analytical functional form that relates H and n for random regular graphs, which are good

Figure 3.2: Demonstration of optimal partition for 3 replicas in ring topology. (a) An example of ring topology with 9 nodes; (b) The root of each partition refers to the node holding the replica, and other nodes forward the content from the root to its users.

models for the dynamic network topology in Cloud Centric Media Network. We first study a few examples of generalized Moore Graph¹, such as ring topology, the Petersen graph, and the Heawood Graph. The symmetry and well-defined structures of these graphs enable us not only to derive analytical form of $H(n)$ but also to gain understandings of the efficient topological structures that allow for efficient content delivery. Using these properties, we further derive a lower and an upper bound of H for a random regular graph with N nodes and a degree of Δ . Finally, we provide close estimates for irregular deployed networks, for which, the exact analytical solutions are difficult to obtain. The generated results serve as a foundation to solve the optimization problem in section 3.4.

3.3.1 Ring Graph

We first study ring graph, which can be treated as a trivial case of Generalized Moore Graphs. It can be observed that for any given number of replicas in ring topology, the optimal replica placement is to maximize the hop distances among those replicas. Thus,

¹Generalized Moore Graph is a Δ -degree regular graph, where each node has a Δ -ary spanning tree that is full at each level, except probably the last.

for a placement of n replicas on a ring graph of N nodes, we have,

$$N = \lfloor \frac{N}{n} \rfloor n + r = r_0 n + r, \quad (3.11)$$

where r is the reminder of dividing N by n . The parameter r_0 indicates that there are $n - r$ replicas, each serving the download requests from a group of r_0 nodes. And each of the remaining r replicas serve $r_0 + 1$ nodes. Figure 3.2 gives an example, where $N = 9$, $n = 3$, $r = 0$, and $r_0 = 3$.

We adopt the divide and conquer approach to derive the solution. In particular, we separately consider two cases (i.e., N is odd and even).

When N is odd, a ring topology is a trivial case of Moore Graph² with node degree $\Delta = 2$, that,

$$\begin{aligned} H(n) &= \frac{1}{N} [n(\frac{r_0 - 1}{2} + 1)\frac{r_0 - 1}{2} + r\frac{r_0 + 1}{2}] \\ &= \frac{r_0 + 1}{2} [1 - \frac{n}{N} \frac{r_0 + 1}{2}], \quad r_0 \text{ is odd.} \end{aligned} \quad (3.12)$$

The first term denotes the total hops for all n replicas serving r_0 nodes, and the second term denotes the hop distances from the additional one node served by r of the replicas.

When N is even, a ring topology is a generalized Moore graph. In this case, $H(n)$ can be derived in the same way as,

$$H(n) = \frac{r_0}{2} (1 - \frac{n}{N} \frac{r_0}{2}), \quad r_0 \text{ is even.} \quad (3.13)$$

By combining Eq. (3.12) and (3.13), we have,

$$H(n) = \lfloor \frac{r_0 + 1}{2} \rfloor (1 - \frac{n}{N} \lfloor \frac{r_0 + 1}{2} \rfloor). \quad (3.14)$$

This approach (i.e., Eq. (3.11)) and results (i.e., Eq. (3.14)) provide guidelines for our further analysis.

²Moore Graph is a Generalized Moore Graph where each node has a full Δ -ary routing spanning tree at all the levels.

Figure 3.3: Demonstration of optimal partition for 3 replicas in Petersen Graph. (a) Petersen Graph; (b) The root of each partition refers to the node holding the replica, and other nodes forward the content from the root to its users.

3.3.2 Petersen Graph & Heawood Graph

We further study the properties of content placement problem in Generalized Moore Graph of degree $\Delta = 3$. In particular we focus on Petersen graph (an example of Moore graph with $N = 10$), and Heawood graph (an example of Generalized Moore Graph with $N = 14$).

3.3.2.1 Numerical Results

To obtain the numerical results for a given replica number n in the two examples, we exhaustively calculate the average shortest distance H from n selected centers to all other nodes for all the possible combinations. After all the iterations are finished, the solution with the smallest H is then selected as the optimal result.

Figure 3.3 and Figure 3.4 present the optimal replica placement solutions when $n = 3$ for Petersen graph and Heawood graph, respectively. The key observation here is that, the minimum hop routing spanning trees are rooted at each of replicas.

From the observation, the original replica placement problem can be transformed into a graph partition problem. That is, for a given number of replica n , and a Generalized Moore Graph $\mathcal{G}(\Delta, N)$ (where $N\Delta$ must be even), how to optimally partition $\mathcal{G}(\Delta, N)$

Figure 3.4: Demonstration of optimal partition for 3 replicas in Heawood Graph. (a) Heawood Graph; (b) The root of each partition refers to the node holding the replica, and other nodes forward the content from the root to its users.

into n sub-trees $S_1, S_2, \dots, S_n$, where $\sum_{i=1}^n |S_i| = N$, and $S_i \cap S_j = \emptyset$ for $i \neq j$, so that the aggregated hop distance, from the roots to all other nodes in all sub-trees can be minimized. Then the average hop distance can be calculated by dividing this aggregated hop distance by total node number N . Mathematically, by assuming A_i as the aggregated hop distance from all the nodes to the root in subtree S_i , we can obtain the mean hop distance H between any user and his nearest replica as,

$$H = \sum_{i=1}^n A_i / N. \quad (3.15)$$

3.3.2.2 A Lower Bound of Aggregated Hop Distance

To analytically obtain a lower bound of aggregated hop distances when partitioning a Generalized Moore Graph, we first derive two important properties, which are invariant under all cases.

Property 3.1 *Each partition forms a Δ -ary spanning tree that is full at each level, except probably the last. (i.e., each sub-tree with its height at h , must have a root at its 0-th level, and $\Delta(\Delta - 1)^{k-1}$ nodes at its k -th level, $k = 1, \dots, h - 2$).*

Proof: See Appendix A.1 for a completed proof. ■

Property 3.2 *The node number of each sub-tree is either $\lfloor N/n \rfloor$ or $\lceil N/n \rceil$ (i.e., a uniform partition).*

Proof: See Appendix A.2 for a completed proof. ■

Following those properties, we find that the optimal solution should contain r sub-trees, each serving r_0+1 nodes (i.e., $|S_1|=|S_2|=...=|S_r|=r_0+1$). There are also $n-r$ nodes serving r_0 nodes (i.e., $|S_{r+1}|=|S_{r+2}|=...=|S_n|=r_0$), where r is the reminder of dividing N by n , and $r_0 = \lfloor N/n \rfloor$. For instance, Figure 3.3(b) illustrates the optimal graph partitions for Petersen graph when $n=3$. Both sub-tree S_2 and S_3 have $r_0 = \lfloor 10/3 \rfloor = 3$ nodes, and sub-tree S_1 has $r_0 + 1 = 4$ nodes. Note that the lower bound may not be reached, because the node at the border of two partitions may have the same shortest distance to more than one replica (e.g., in Figure 3.4(b), the hop distances from node 3 to replica 8 and 4 are both 1).

3.3.2.3 Analytical Solution

We derive the analytical lower bound for Petersen graph and Heawood graph by considering two parts. First, we calculate the aggregated hop distances A_o from all n sub-trees, each is with r_0 nodes, and all its levels are full. For the second part, we compute the extra distance from the additional node in the last level of those r sub-trees.

To proceed the analysis, we first investigate the optimal height h of each sub-tree, where $\Delta \geq 3$, is the least integer that satisfies the following condition,

$$\Delta \frac{(\Delta - 1)^h - 1}{\Delta - 2} \geq r_0 - 1. \quad (3.16)$$

The expression on the left side is the total number of nodes for a full tree with a height of h . The expression on the right side is the number of the leaves (excluding the root) in the subtree S_{r+1} to S_n .

From Eq. (3.16), we have h as a function of Δ and r_0 as,

$$h = \lceil \log_{\Delta-1} \frac{\Delta r_0 - 2r_0 + 2}{\Delta} \rceil. \quad (3.17)$$

Since the last level of those sub-trees may not be necessary full, we have to find the node count o in the last level at each sub-tree as,

$$o = r_0 - 1 - \Delta \frac{(\Delta - 1)^{h-1} - 1}{\Delta - 2}. \quad (3.18)$$

Thus, the aggregated hop distance from each of the n trees with r_0 nodes for both the Petersen and the Heawood graph, A_o , can be expressed as,

$$A_o(h, o) = 2^{h-1}(3h - 6) + 3 + ho, \quad (3.19)$$

where the first term refers to the aggregated hop distance from the nodes excluding the first and the last level. The second term, '3' refers to the distance from all the nodes in the first level, and the last term refers to the distance from all the nodes in the last level.

Then we compute the aggregated hop distance φ from the additional node from each of the r trees, as,

$$\varphi(h, o) = h + \delta(\Delta(\Delta - 1)^{h-1} - o), \quad (3.20)$$

where $\delta(x)$ is an indicator function, indicating whether those trees are full before adding the node, as,

$$\delta(x) = \begin{cases} 1, & x = 0 \\ 0, & \text{otherwise.} \end{cases} \quad (3.21)$$

Finally, the lower bound of the average hop distance $H(n)$ as a function of n in the Petersen graph can be solved as,

$$H(n) \geq \frac{1}{10}(nA_o(h, o) + r\varphi(h, o)). \quad (3.22)$$

Similarly, the lower bound for Heawood graph follows exactly the same format as that of Petersen graph as,

$$H(n) \geq \frac{1}{14}(nA_o(h, o) + r\varphi(h, o)). \quad (3.23)$$

It can be seen that, the only difference between Eq. (3.22) and Eq. (3.23) is the first parameter. This inspires us to extend the analysis to arbitrary Generalized Moore Graph by adopting the same rationale.

3.3.3 Generalized Moore graph

We first compute the aggregated hop distance A_o for each of the n sub-trees. Note that each sub-tree has r_0 nodes,

$$A_o(h, o) = \Delta \sum_{k=0}^{h-2} (k+1)(\Delta-1)^k + ho, \quad (3.24)$$

where the first term is the aggregated hop distance from all the nodes excluding the last level. There are Δ nodes in the first level, and each of those node has $\Delta-1$ children. Therefore, it leads to the result, that there are $\Delta(\Delta-1)^k$ nodes locating $k+1$ hops away from the root, for $k = 0, 1, \dots, h-2$. The second term is the sum of total hop distance to the root from the nodes on the last level. We note that in this term o still has the same expression as in equation (3.18).

Thus, we have a lower bound for Generalized Moore Graph $\mathcal{G}_M(\Delta, N)$ with N nodes and degree Δ , as,

$$H(n, \mathcal{G}_M(\Delta, N)) \geq \frac{1}{N}(nA_o(h, o) + r\varphi(h, o)). \quad (3.25)$$

This result will be further used to derive an upper bound and a lower bound in random regular graph.

3.3.4 Random Regular Graph

In random regular graph, each node still has the same degree Δ , but not necessarily the same connectivity pattern. To extend the analysis on content placement problem in such graph, we first get a lower and an upper bound for H . Then we derive an approximation that captures the fundamental scaling of H as a function of N , Δ , and n .

3.3.4.1 Lower Bound

There is a lower bound, when all the levels of each sub-tree are full except the last. This case is exactly the same as the analysis for Generalized Moore Graph. Therefore, this lower bound can be expressed as,

$$H(n, \mathcal{G}(\Delta, N))_l = \frac{1}{N}(nA_o(h, o) + r\varphi(h, o)). \quad (3.26)$$

To further simplify the functional form of $H(n, \mathcal{G}(\Delta, N))_l$, we make the simplification that N is dividable by n (i.e., $r_0 = N/n$ and $r=0$). We also assume that $\Delta \approx \Delta + 1$. With these simplifications, we have $h \approx \log_{\Delta} r_0$, $o \approx 0$, and $A_o(h, o) \approx \sum_{k=1}^{h-1} k\Delta^k = \frac{((h-1)\Delta - h)\Delta^h + \Delta}{(\Delta-1)^2} > \frac{r_o(h(\Delta-1) - \Delta)}{(\Delta-1)^2}$. Finally, Eq. (3.26) can be reduced to the following form,

$$H(n, \mathcal{G}(\Delta, N))_l \approx c_1 \log_{\Delta} \frac{N}{n} + c_2, \quad (3.27)$$

where $c_1 \approx 1/(\Delta - 1)$ and $c_2 \approx \Delta/(\Delta - 1)^2$ are independent of n and N .

3.3.4.2 Upper Bound

We use the upper bound for random regular graph obtained from [53], to get the height upper bound h_u of each sub-tree after partitioning as,

$$h_u \leq \log_{\Delta} (2 + \epsilon)r_0 \log r_0 < 2 \log_{\Delta} c_3 r_0, \quad (3.28)$$

where c_3 is a constant as $\sqrt{2 + \epsilon}$, and ϵ is a small number.

As a result, it is safe to draw the conclusion that, when each sub-tree with the height at $h_u = 2 \log_{\Delta} c_3 r_0$ is full at all the levels, it holds an upper bound for the aggregated hop distance (note in this case, the node number of each sub-tree may be larger than r_0). By using the same approach on deriving the lower bound, we have an upper bound as,

$$H(n, \mathcal{G}(\Delta, N))_u \approx (\Delta c_3)^2 \log_{\Delta} \frac{c_3 N}{n} + c_4, \quad (3.29)$$

where c_4 is a constant which is independent with n and N .

3.3.4.3 Approximation Result

By combining Eq. (3.27) and (3.29), we have the approximation form as,

$$H(n_k, \mathcal{G}(\Delta, N)) \approx C_1 \log_{\Delta} \frac{C_2 N}{n_k}, \quad (3.30)$$

where C_1 and C_2 are topology-specific coefficients. Compared with existing approximation form in power-law function (i.e., $H = C_1(N/n)^{C_2}$) [54, 55], our solution can further capture the topological characteristics via Δ in logarithmic function. As a result, it is expected to be able to more accurately represent the relationship between H and n , for different network topologies.

The result suggests that, assuming a uniform user request pattern, the mean hop distance between users and their closest replica is a logarithmic function of the reciprocal of replica number in random regular graph. As to be shown in the following session, this convex form allows us to analytically solve content placement or similar resource allocation problem.

3.3.5 Deployed Networks

We study four representative deployed networks [52] as shown in figure 3.5, which roughly share the same basic properties with the random regular graph. We make this argument by evaluating the graph diameter (i.e., the maximum distance among all pairs of nodes in a graph G), which is one of the most important metrics [53]. Specifically, we evaluate the analytical lower bound (i.e., $a_l \approx \lfloor \log_{\Delta-1} N \rfloor + 1$) and upper bound (i.e., $a_u \approx \lceil \log_{\Delta-1} 2n \ln n \rceil + 1$) [53] of random regular graphs, which have the same degree (i.e., average node degree) and size with those in deployed networks, and compare the analytical results on random regular graph with the deployed network as shown in table 3.2. It can be seen that, the diameters of those deployed graphs are between the lower and upper bound of their counterparts in random regular graph forms.

Figure 3.5: Deployed networks: (a) the U.S IP backbone network; (b) the National Science Foundation Network (NSFNET); (c) the hypothetical backbone network US64; (d) the European optical network (EON)

Figure 3.6: Estimated functions for deployed networks. The obtained logarithmic function is more precise than the existing estimation in all four topologies.

To evaluate the performance of our solution in deployed networks, we compare the analytical results generated from our logarithmic function with the ones that use a power-law function (i.e., $H(n_k) \approx C_1(\frac{N}{n})^{C_2}$) [54, 55]. Specifically, by using numerical algorithms, we first get the exact optimal placement of n replicas and the associated $H(n)$, for $n = 1, 2, \dots, N$. Next we fit these results with both Eq. (3.30), and the power-law function from [54, 55]. The accuracy of the curve-fitting is evaluated by Mean Squared Error (MSE), denoted as,

$$\sigma^2 = \frac{1}{N} \sum_{i=0}^N (\tilde{x}_i - x_i)^2, \quad (3.31)$$

where $\tilde{x}_i$ is the estimation, and x_i is the numerical result.

Figure 3.6 illustrates this comparison. In all the cases, our obtained logarithmic

Table 3.2: Comparison on theoretical and real diameters for different topologies

	U.S IP	NSF	U.S 64	EON
Lower Bound	4	4	10	3
Diameter	6	4	16	4
Upper Bound	7	8	16	6

function is able to represent the real condition more precisely with much lower MSE. As a result, we use this function to estimate H , and drive the design of our algorithm to solve the content placement problem for a catalog of content in the next section.

3.4 Optimal Placement for A Catalog of Content

By substituting Eq. (3.30) into the origin problem (i.e., Eqs. (3.5)-(3.9)), we get a convex optimization problem. Specifically, the constraint (3.9) is convex, because each item, $H(n_k) = C_1 \log_{\Delta}(C_2 N/n_k)$, for $k = 1, \dots, M$, in matrix $\mathbf{H}$, is convex. Constraints (3.6)-(3.8) are all linear functions of replica number. And the objective function is also convex, which is the sum of a linear function and a convex function.

Due to the convexity of this problem, this section first adopts a feasible direction algorithm to find the optimal solution in general condition. By using the Karush-Kuhn-Tucker (KKT) condition, we also present the optimal solution for the problem without capacity constraints to gain some fundamental insights.

3.4.1 General Solution

In this subsection, we apply the Topkis-Veinott's feasible direction method [56] to solve the convex optimization problem. Algorithm 3.1 describes the details of this method.

The optimality criteria of this algorithm is to iteratively move from one feasible point to another improved one. Specifically, the algorithm starts with one random feasible point $\mathbf{n}_1$, and in each iteration, finds the reduced gradient direction $\mathbf{d}_k$ by solving a one-dimensional optimization problem to determine how far to proceed along this direction.

As a result, it can be proved that the new point $\mathbf{n}_{k+1} = \mathbf{n}_k + \lambda_k \mathbf{d}_k$ must be better or equal than the previous point $\mathbf{n}_k$. This process is repeated until the optimal solution is found.

Our algorithm also assures the convergence into the optimal point. To prove this argument, first, we show the generated solution is a Fritz John point (i.e., a feasible point that is local optimal) by using the proof from [56]. Specifically, it has been proved that in the feasible direction algorithm, $\mathbf{n}_k$ is a Fritz John point if and only $z_k = 0$, which is exactly the termination criterion of our algorithm. Second, we prove the obtained Fritz John point is also a KKT point (i.e., the feasible point that is global optimal) by checking those additional restrictions of KKT conditions. In particular, it has been proved that [57], if the objective function and all the inequality constraints are continuously differentiable convex functions, and the Lagrangian multipliers on the gradient of the objective function are nonzero, then the Fritz John point is also a KKT point. Because our problem meets those restrictions, the global optimality of the solution is guaranteed.

The complexity of this algorithm is $O(M)$. This statement can be proved as follows. First, it has been proved that [56], the feasible direction method can always find the optimal solution in finite iteration rounds C . For each iteration, the algorithm takes $2M+2$ steps to find the reduced gradient direction by solving the one-dimensional optimization problem with $2M+2$ constraints. As a result, the total steps are $C(2M+2)$, and the complexity is $O(M)$.

3.4.2 Optimization without Capacity Constraints

Algorithm 3.1 provides a way to solve the optimization problem under constraints, but it is difficult to generate an analytical result by this algorithm. As a result, we consider the optimization problem without capacity constraints, and obtain analytical solutions for fundamental insights.

Algorithm 3.1 The Feasible Direction Algorithm

Input: The objective function $f(\mathbf{n})$

The constrains $g_i(\mathbf{n})$, $i = 1, 2, \dots, 2n + 2$

One feasible solution $\mathbf{n}_1$ such that all $g_i(\mathbf{n}_1) \leq 0$

Output: The optimal solution $\mathbf{n}^*$

- 1: Initialize $k=1$;
 - 2: Find a feasible descent direction $\mathbf{d}_k$ by solving a one-shot problem as,

$$\begin{aligned} & \min z_k \\ \text{s.t. } & \nabla f(\mathbf{n}_k)^t \mathbf{d} - z_k \leq 0 \\ & \nabla g_i(\mathbf{n}_k)^t \mathbf{d} - z_k \leq -g_i(\mathbf{n}_k), \text{ for } i = 1, \dots, 2M + 2 \\ & -1 \leq d_j \leq 1, \text{ for } j = 1, \dots, 2M + 2 \end{aligned}$$

if $z_k = 0$ **then** go to Step 4
else go to Step 3
 - 3: Find the step λ_k along direction $\mathbf{d}_k$ by solving a one-shot problem as,

$$\begin{aligned} & \min f(\mathbf{n}_k + \lambda \mathbf{d}_k) \\ \text{s.t. } & 0 \leq \lambda \leq \lambda_{max} \\ & \lambda_{max} = \sup\{\lambda : g_i(\mathbf{n}_k + \lambda \mathbf{d}_k), \text{ for } i = 1, \dots, 2M + 2\} \end{aligned}$$

Update $\mathbf{n}_{k+1} = \mathbf{n}_k + \lambda_k \mathbf{d}_k$, $k = k + 1$
 go to Step 2
 - 4: Finish, return $\mathbf{n}_k$ as $\mathbf{n}^*$
-

By removing the constraints (3.8) and (3.9), the optimization problem becomes an optimization process over a box (i.e., $1 \leq n_i \leq N$, $i = 1, \dots, M$). Therefore, we can turn it into an unconstrained problem, by ensuring and $\nabla f(n_k) \leq 0$ for $n_k = N$, and $\nabla f(n_k) \geq 0$ for $n_k = 1$. By considering the KKT first order conditions, we could find the optimal replica number for each content by letting $\nabla f(n_k) = 0$. Thus we have,

$$\nabla f(n_k) = c_{st} B_k - \frac{c_{tr} B_k R_k C_1}{n_k \ln \Delta} = 0. \quad (3.32)$$

By solving Eq. (3.32), and jointly considering the constraints (3.6) and (3.7), we have the optimal replica number for a catalog of content as,

$$n_k^* = \max\{1, \min(N, \frac{c_{tr} A R_k}{c_{st} t \ln d})\}. \quad (3.33)$$

This result suggests that, under feasible constraints (i.e., $n_k \in (1, N)$), unconstrained

Figure 3.7: The real-world trace from a video provider. (a) The size of different contents follows Pareto distribution; (b) The popularity of different contents follows Zipf distribution.

storage and bandwidth resources), the optimal replica number n_k^* of content k is proportional to its downloaded times R_k in those deployed networks.

Moreover, by substituting the optimal replica number into the origin problem, we can get the optimal storage space and the optimal network bandwidth as,

$$T_{tot}^* = c_{tr} \mathbf{B}^T \mathbf{H}^* \mathbf{R}, \quad (3.34)$$

$$S_{tot}^* = \mathbf{B}^T \mathbf{n}^*, \quad (3.35)$$

where $\mathbf{H}^* = \text{diag}(C_1 \log_{\Delta} \frac{C_2 N}{n_1^*}, \dots, C_1 \log_{\Delta} \frac{C_2 N}{n_M^*})$, and $\mathbf{n}^* = (n_1^*, \dots, n_M^*)$. In practice, any other resource allocation schemes, that are different with T_{tot}^* and S_{tot}^* , will lead to either over- or under-provision.

3.5 Performance Evaluation

This section uses real-world traces to evaluate the obtained monetary cost efficient content placement strategy.

3.5.1 Experimental Settings

The real-world trace was captured from one of the leading video providers. It contains the request history log of 50 contents over one week. Figure 3.7 shows the distributions

of both the size and the popularity in terms of the downloaded times of this real data.

It can be seen that, the real trace agrees with the content size distribution model and the user request distribution model as discussed in section 6.2. It verifies our assumptions on these two models. In particular, the bounded Pareto distribution with its parameter $\alpha=0.78$, $B_U=150$ MB and $B_L=1$ MB matches well with the real content size distribution, where the estimation MSE is $\sigma^2 = 0.038$. And the Zipf distribution with its parameter $\beta=0.86$ and $R_{tot}=2.9 * 10^4$ represents the real popularity distribution of each content, where the estimation RMSE (root of MSE) is $\sigma = 1.69$.

We adopt pay-per-use CDN pricing model from GoGird [58], one leading cloud service provider, as an example to drive our experiment. Specifically, storage price is $c_{st} = \$0.6/GB$ per month and bandwidth price is $c_{tr} = \$0.25/GB$. Note, similar pricing models are widely used by other famous cloud service providers, such as Microsoft Azure ($c_{st} = \$0.19/GB$ per month and $c_{tr} = \$0.19/GB$) [59], and Rackspace ($c_{st} = \$0.18/GB$ per month and $c_{tr} = \$0.10/GB$) [60]. Besides, we apply those deployed networks in section 3.3 to deliver the traces, and produce the optimal solutions by our methods.

We implement the simulator by programming over 1000 lines C# codes. This simulator takes the aforementioned parameters, four network topologies, Equation (3.30), and Algorithm 3.1 as the input to generate the optimal replica number for each content and the corresponding operational cost.

3.5.2 Fundamental Trade-off for Single Content Placement

Figure 3.8 uses the 10th popular content in the trace as an example to show the fundamental trade-off between storage and bandwidth cost for single content placement. It can be seen, in all the four deployed networks, the storage cost is in proportional to the replica number, and the bandwidth cost follows a logarithmic function of the reciprocal of replica number. The trade-off here is that, as the replica number grows, the storage cost

Figure 3.8: Monetary cost for the 10^{th} popular content. The total cost is a convex function of the replica number n in all cases.

Figure 3.9: Total monetary cost for different content. The overall cost function for the content with different popularity is different.

increases and the bandwidth cost decreases, while the total cost is convex. Consequently, how to balance this trade-off to find the optimal replica number for each content under resource constraints, drives this work.

Besides, figure 3.8 also indicates the optimal solution depends on its underlying topology G . It can be seen, for the same content to be delivered over different networks, the minimal total cost can be achieved by different replica number. For example, if we do not consider the resource constraints, the optimal replica number for the 10th popular content is 6 in U.S IP backbone network, 6 in NSF network, 8 in U.S 64 network, and 5 in EON.

3.5.3 Fundamental Trade-off for A Catalog of Contents

Figure 3.9 illustrates the total monetary cost in function of replica number for different contents with different popularity (i.e., the 5th, the 10th, the 20th, the 30th, and the 40th popular content). It can be observed, as the content popularity decreases, the optimal replica number decreases accordingly. This indicates the trade-off for a catalog of content depends on the content popularity. For example, in U.S IP backbone network, if we do not consider the resource constraints, the optimal replica number is 12 for the 5th popular content, 6 for the 10th popular content, 3 for the 20th popular content, 2 for the 30th popular content, and 1 for the 40th popular content.

Figure 3.9 also indicates, the total monetary cost depends on not only content popularity but also content size. For example, in NSF network, when replica number of the 5th popular content is less than 6, its overall monetary cost is the highest among all the contents, mainly because the enormous bandwidth cost incurred by insufficient replica number and the large amount of downloaded times. When the replica number of this content is more than 10, its overall cost is lower than the cost of the 20th popular content, which attracts much less downloaded times (i.e., 14.38 times/hr) than the 5th popular content does (i.e., 74.23 times/hr). This can be attributed to the relative small size of the 5th popular content (i.e., 5.87 MB) compared to the size of the 20th popular content (i.e., 9.39 MB), and the consequent small storage cost.

3.5.4 Optimal Overall Monetary Cost

Figure 3.10 presents the optimal overall monetary cost under various combinations of storage and bandwidth constraints. We carefully select the range of these two parameters to emulate all possible conditions (i.e., one of the constraint is active, both are active, and both are inactive). The objective is to understand the impact of different constraints.

Figure 3.10: Total monetary cost under different constraint combinations, decreases as the storage/bandwidth constraint increases until the other constraint becomes active.

For a given T_{tot} , there are two phases as the growth of S_{tot} , depending on whether the storage constraint is active. Specifically, in the first phase, the limited storage capacity constrains the optimal solution. Thus, the total monetary cost decreases as the storage space increases (e.g., $T_{tot} = 300$ MB/s and $S_{tot} \leq 2.2$ GB in U.S IP backbone), while in the second phase, the storage capacity is no longer the dominating factor, and the optimal cost stays the same (e.g., $T_{tot} = 300$ MB/s and $S_{tot} \geq 2.2$ GB in U.S IP backbone). The threshold to separate these two phases decrease as the bandwidth capacity increases, until the bandwidth constraint also becomes inactive (e.g., in U.S IP backbone, the threshold is 2.2 GB when $T_{tot} \geq 300$ MB/s, and 2.4 GB when $T_{tot} = 250$ MB/s). It is also noticeable that, if both the storage and bandwidth capacity are insufficient (e.g., $S_{tot} \leq 1.5$ GB and $T_{tot} = 350$ MB/s in U.S IP backbone network), there could be no feasible solution. This set of observations again verifies the trade-off between these two kinds of resources.

3.5.5 Optimal Replica Number Distribution

Figure 3.11 plots the optimal replica number of a catalog of content in different conditions (i.e., unconstrained, constrained by storage resource, and constrained by bandwidth resource).

First, we find that, all the curves are roughly linear in a log-log graph. This suggests the optimal replica number for each content follows a power-law distribution to its popularity rank, regardless of whether the solution is constrained by resource limitations. Note, this insight only applies to $n_k \in (1, N)$.

Second, we notice that, when the bandwidth constraint is active, the optimal replica number is always the highest; while the storage constraint is active, the optimal replica number is the lowest. This still can be attributed to the trade-off between the two resources. Specifically, when bandwidth resource is limited, we have to place more replicas to reduce the bandwidth cost. And when storage space is limited, less replicas are allowed

Figure 3.11: The optimal distribution of content replica number follows a power-law distribution in terms of its popularity rank in all four topologies.

to be placed. The optimal criteria of balancing these competition factors among different contents, is ensured by the reduced gradient direction method in our algorithm.

3.6 Algorithm Evaluation

To further evaluate the performance and optimality of our proposed algorithm, we also adopt three natural heuristics (i.e., random algorithm, popularity based algorithm and near-optimal algorithm) inspired by [7, 8] to solve the same problem. We then compare the results generated by those algorithms with our approach, in terms of the obtained total monetary cost and the computational complexity.

3.6.1 Heuristics Design

Since the design of those heuristics could be very complex if the problem is constrained by both bandwidth and storage resources, we only consider the storage constraint here. The detailed descriptions on these three heuristics are as follows.

Random Algorithm [7] assigns replicas of each content to different nodes randomly, subject to the total storage constraints. Specifically, we first allocate one replica to all the contents, to meet the lower bound of replica number and the storage constraint (if the storage resource is still insufficient in this case, then there is no feasible solution). Then, for each content k , we randomly pick additional n_k replicas ($0 < n_k < \min\{N, \lfloor S_{free}/B_k \rfloor\}$), where S_{free} is the remaining amount of free storage space. This process iteratively runs, until all the contents have been allocated corresponding replica number, or the storage resource runs out. To improve the performance, we run this algorithm for 10 times, and pick assignment that yields the lowest cost in our experiment. Note this algorithm does not need any information on the content popularity and underlying topology.

Popularity Based Algorithm [8] allows the most popular contents have the upper bound replica number N , subject to the storage constraint, while the rest contents have only 1 replicas. Specifically, we still first assign one replica to every content, then iteratively assign $\min\{N, \lfloor S_{free}/B_k \rfloor\}$ replicas to each content from the most popular one to the least popular one, until all the contents have been assigned, or the storage resource runs out. Note this approach uses content popularity information (which may not in a proper way), but it does not require the network topology information to calculate the hop distance between replica and clients.

Near-optimal Algorithm [7] iteratively calculates the total monetary cost generated by all the possible replica number for every single content k from the most popular one to the least popular one. When the storage space is sufficient, it picks the optimal replica number n_k^* with the lowest cost. Otherwise if storage space is not enough that $B_k * n_k^* > S_{free}$, it picks the sub-optimal replica number $n_k^o = \lfloor S_{free}/B_k \rfloor$. Similarly, it starts by assigning one replica to each content, and ends when all the contents have been allocated proper replica number, or the storage resource runs out. Note this algorithm uses both content popularity and underlying topology information.

3.6.2 Total Cost Comparison

Figure 3.12 compares the total monetary cost generated by four different approaches based the same data set when the bandwidth resource is infinite. It can be seen, random algorithm is consistently the worst for different storage capability, in U.S IP backbone network, NSF network and EON network. Even in U.S 64 network, the result generated by random algorithm is still no better than the one yielded by popularity based algorithm. The second worst algorithm is popularity based algorithm. Compared to the results generated by near-optimal algorithm and our approach, popularity based algorithm yields the cost that is 15%-30% higher. Near-optimal method and our proposed approach

Figure 3.12: Comparisons on the optimal total monetary cost generated by different algorithms, where the bandwidth resource is unconstrained. Our approach can find the near-optimal solution that is very closed to the optimal one, with much lower complexity.

perform almost the same (the difference is within 1%). They both generate the lowest total cost among all the four algorithms.

The observations can be understood as follows. Random algorithm is the simplest approach which just randomly picks the replica number for each content regardless of any context information (e.g., content popularity and topology). As a result, it performs the worst, and the generated results fluctuate for different rounds. Popularity based algorithm takes content popularity into consideration, but it fails to utilize it in an effective way. For example, in U.S 64 network, when the storage capacity is less than 2.85 GB, the total cost increases as the storage space grows. This is because the last assigned $\lfloor S_{free}/B_k \rfloor > n_k^*$, that the reduced bandwidth cost cannot overcome the increased storage cost. Near-optimal algorithm finds the optimal solution based on the popularity and underlying topology information at each step. It has been proved such method can yield results very closed to the optimal ones [7]. Our approach uses the analytical relationship between replica number and mean hop distance for deployed networks. It also successfully obtains the near optimal solution.

3.6.3 Computational Complexity Comparison

Table 3.3 shows the time consumed by different algorithms in different network topologies. We run those simulations implemented by C sharp codes on a desktop PC with 2.4GHz dual-core CPU and 2 GB memory. Each simulation is executed for 10 times, and the average results are put on the table (except for near-optimal algorithm in U.S 64 case, which costs too much time). It can be observed that, both random and popularity-based heuristics carry light computing loads. While the complexity of our approach is slightly higher than them within a factor of 2.5-17, and the near-optimal algorithm costs significantly more time (within a factor of 80-50000).

To explain this observation, we analytically find the complexity of those four algorithms. First, both random and popularity algorithm assign replica to each content for at

Table 3.3: The time used to compute optimal replica number (s)

	U.S IP	NSF	U.S 64	EON
Random	0.005	0.001	0.20	0.003
Popularity	0.007	0.001	0.22	0.004
Near-optimal	15.09	0.08	10600	0.33
Our Approach	0.042	0.003	3.73	0.010

most one time. As a result, their complexity are both $O(M)$. Second, we have shown our solution also has a similar reasonable complexity at $O(M)$ in subsection 3.4.1. Finally, near-optimal algorithm has to calculate all the possible combinations for each content with each possible replica number, to determine the optimal assignment with the lowest cost. Even we use near-optimal method introduced by [7] to find the optimal solution for a given replica number n_k , the complexity of this algorithm can be still as high as $O(M^2N^2)$. Thus it is not practical when both M and N are large.

3.7 Conclusion

In this chapter, we investigated the content placement problem, in which content service providers use cloud resources in pay-per-use model to orchestrate content delivery. The objective is to minimize the combined storage and bandwidth cost. We formulated this problem as a constrained optimization problem, under resource constraints, to balance the fundamental trade-off between storage and bandwidth cost. To address this problem, this chapter mainly covered three aspects,

- First, for the placement of single content, we translated it into the k -center problem in graph theory, and derived an approximation for the optimal mean hop distance between users and contents in function of replica number. Upon the assumption of uniform user request pattern, we showed a logarithmical relationship between the mean hop distance H and the reciprocal of replica number n , (i.e., $H \sim \log(N/n)$), in random regular graph and some deployed networks, with the size of N -nodes.

- Second, for the placement of a catalog of content, we got a convex optimization problem, and solved it by applying feasible direction algorithm. We also analytically found the optimal replica number for each content without resource constraints. The results may suggest the optimal replica number follows a power-law to its popularity in those deployed networks, under feasible constraints.
- Finally, we used real traces to verify our monetary cost effective content placement strategy. We numerically showed the fundamental trade-off between storage cost and bandwidth cost in content placement problem. We also verified the power-law distribution on replica number for a catalog of contents under different combinations of resource constraints. Moreover, the optimality and the efficiency of our approach are evaluated, by comparing with three alternative heuristic algorithms.

Chapter 4

Cost Efficient In-Network Transcoding

4.1 Introduction

Nowadays, the increasing popularity of adaptive video streaming services poses significant challenges to existing video distribution approaches. First, the large-scale distribution of video contents calls for tremendous bandwidth resources, especially during peak period. This can be hardly sustained by the traditional static mechanism of network resource provision [51]. Second, in the context of on-demand adaptive streaming, each video content has multiple versions to fit the network condition and the usage of different end-devices. Thus, it is necessary to have computing and storage resources to transcode video contents and cache them at some intermediate nodes within the network [61]. Finally, the lack of coordination on those resources, leads to poor cost efficiency and limited flexibility.

To achieve this target in cloud centric media network, we need to carefully evaluate a three-way trade-off between the storage, computing, and bandwidth cost. On one hand, sufficient caching spaces should be allocated to bring those popular video contents to a closer place to users, so that significant bandwidth cost can be saved. Moreover, as the in-network transcoding becomes a promising function in today's industry [62, 63],

it is possible to reduce the operational cost by keeping only the highest bitrate, and transcoding other versions on the fly. On the other hand, the generated caching and transcoding cost could also overwhelm their benefits, if not appropriately scheduled. Therefore, a cost-efficient multi-resource scheduling scheme should be in place, to balance the trade-off and minimize the total operational cost.

In this chapter, we investigate the optimization problem on multi-resource allocation, specifically for on-demand adaptive video streaming services. The objective is to minimize the overall operational cost associated with the usage of storage, computing and bandwidth resources. A two-step approach is adopted to analytically solve this problem. First, given a fixed caching space at each edge server, we formulate a convex optimization problem, and derive a closed-form expression of the optimal transcoding configuration, with respect to the local caching size. Second, we use the obtained results to further formulate another convex optimization problem, and analytically get the optimal amount of caching space that should be allocated at each edge server.

The contributions of this chapter are manyfold, including:

- We mathematically build a convex optimization problem to examine the trade-off between caching, transcoding and bandwidth cost, when using media cloud to distribute adaptive video streaming services.
- We derive the closed-form of optimal transcoding setting and caching space for each edge server. The obtained solutions provide an analytical framework for evaluating the cost associated with different application scenarios.
- Through extensive simulations, we find our strategy achieves significant cost savings compared with existing schemes. Moreover, the optimal solution and its benefit can be affected by various parameters including, the unit cost, the hop distance to the origin server, the request patterns, and the settings of different bitrate version.

Figure 4.1: End-to-end view of on-demand adaptive video streaming via media cloud. The end-users consume videos in different versions for different devices. Those video segments are originally published by the origin server. They are transmitted to the users upon request via edge servers, which can strategically cache or transcode video segments to reduce the total cost of operating the adaptive streaming services within the media cloud.

The rest of this chapter is organized as follows. Section 4.2 discusses the system architecture, models and the problem formulation. Section 4.3 theoretically analyzes the optimal strategy. Section 4.4 numerically evaluates the performance of our solution. Finally, Section 7.2 summarizes this work.

4.2 System Models & Problem Formulation

In this section, we first describe the system architecture to provide necessary background. Then, we illustrate the fundamental trade-off via a motivation example. Following that, we discuss four system models. Finally, using those models, we formulate a constrained optimization problem for multi-resource allocation. For clarity in the following discussion, we summarize those key notations as shown table 6.1.

Table 4.1: Notation table

Symbol	Definition
$G(V, E)$	Media cloud topology with node set V and edge set E .
d_i	Hop distance from node i to the origin server.
r	Number of available bitrate versions.
m	Number of different video segments.
s_{ij}	i -th popular segment with the j -th highest bitrate.
b_{ij}	Segment size of s_{ij} .
b_s	Summation of all bitrates for a segment.
b_h	The highest bitrate version of a segment.
b_m	Mean size of all bitrate versions for a segment.
$p(s_{ij})$	Prob. of requesting s_{ij} .
p_j	Prob. of requesting the j -th bitrate version.
α	Zipf exponent.
c_i	Allocated cache space at node i .
x_i	Number of all-bitrate versions cached by node i .
h_i	Number of different segments can be held by node i .
w_{ca}	Unit cost of allocating caching space to edge servers.
w_{bw}	Unit cost of bandwidth consumption per hop.
w_{tr}	Unit cost of online transcoding a segment.
C_i^{ca}	Storage cost to serve per request at node i .
C_i^{bw}	Bandwidth cost to serve per request from node i .
C_i^{tr}	Transcoding cost to serve per request from node i .
p_i^{hit}	Prob. of serving requests locally at node i .
p_i^{tr}	Prob. of local transcoding ratio at node i .
p_i^{miss}	Prob. of local cache miss ratio at node i .

4.2.1 System Architecture

Figure 6.1 shows a systematic end-to-end view of delivering on-demand adaptive streaming over media cloud. It mainly consists of three parts, including an origin server, a media cloud, and users with different devices. Specifically, all video segments with all available bitrate versions are originally published by an origin server [64]. They are delivered to those end-devices in different formats with various bitrates upon user requests, throughout a media cloud. Within the media cloud, each edge server is a service entry point attached with an access network. The cloud service providers dynamically instantiate virtualized caching and transcoding resources from a resource pool, and decide how to allocate them on each edge server. As a result, some user requests can be served directly by edge servers based on in-network caching or online transcoding [65, 66]. This significantly saves the bandwidth cost of delivering adaptive streaming from the origin server.

To achieve cost-efficient on-demand adaptive streaming services, it is critical to minimize the total operational cost incurred by orchestrating multiple virtualized resources for media cloud service provider. Indeed, such operational cost is decided by the orchestration on caching, transcoding and bandwidth resources. In particular, on the one hand, significant bandwidth cost can be saved, by allocating more cache spaces to an edge server to make the contents much closer to end-users. Moreover, by moving some transcoding resources to the edge server, the local cache hit ratio and the bandwidth consumption can be further improved. On the other hand, certain operational costs will be incurred by using either caching or transcoding resources. Such cost could overwhelm the savings on the bandwidth consumption. As a result, there is an opportunity to reduce the total operational cost by examining the three-way trade-off between the caching cost, the transcoding cost and the bandwidth cost.

4.2.2 Motivating Example

For the sake of clarifying the problem, we present an illustrative example as shown in Figure 6.4. In particular, on the server side, we assume the origin server hosts three different video segments (i.e., a , b , and c), where each segment has two different versions. The size of version 1 is 2MB, and the one of version 2 is 1MB. On the end-user side, we assume there are three requests on both a_1 and a_2 , two requests on b_1 and b_2 , and one request on c_1 and c_2 . These users are connected to the media cloud via an edge server that is 3 hops away from the origin server. In the middle, the unit bandwidth cost is 1 per MB per hop, the unit transcoding cost is 2 per MB, and the unit caching cost is 5 per MB. Under this setting, we try to investigate how different configurations on the edge server affect its total operational cost.

Table 4.2 shows a cost comparison on six different strategies with different caching and transcoding configurations, based on the motivation example scenario. This table clearly demonstrates the fundamental three-way trade-off. Specifically, first, via the comparison between the first column and the rests, we show that the total cost can be reduced by allocating caching and transcoding resources to the edge server. Second, by comparing the second, third, and fourth column, we show that the allocation of either too many or too little caching resources may increase the total cost. Finally, by comparing the third column with the last column, we show that the total cost can be further reduced by introducing the transcoding resources. As a result, the objective of this chapter is to find out the optimal solution that minimizes the total cost.

4.2.3 System Models

4.2.3.1 Topology Model

We model the topology of media cloud as a undirected graph $G = (V, E)$, where V denotes the set of edge servers, $n = |V|$ denotes the total number edge servers, and E denotes

Figure 4.2: An example scenario of on-demand adaptive video streaming services. The users are requesting three video segments in two different versions, via an edge server that is 3 hops away from the origin server.

Table 4.2: Comparisons on different strategies in the motivation example

	w/o ca w/o tr	2MB ca w/o tr	5MB ca w/o tr	8MB ca w/o tr	2MB ca with tr	5MB ca with tr
C^{bw}	54	36	15	3	27	9
C^{ca}	0	10	25	40	10	25
C^{tr}	0	0	0	0	6	4
C^{tot}	54	46	40	43	43	38

the set of network links between those servers. For simplicity, this work considers there is only one origin content source $s \in V$, with d_i denoting the hop distance from the node i to s . This can be easily extended to the case of multiple different sources, by separately generating the solution for each of them using the identical method.

This topology $G(V, E)$ is built on top of multiple resources. Specifically, each node $v \in V$ is associated with both caching and transcoding resources, and each link $e \in E$ is associated with bandwidth resources. The consumption on different resource component will incur a corresponding amount of cost, in a pay-per-use manner.

4.2.3.2 Content Model

We consider there are in total m different video segments to be distributed, where each segment has r different bitrate versions ranging from the highest bitrate b_1 to the smallest bitrate b_r . We assume the length of each segment is the same on average, and all those

r bitrates are frequently accessed (i.e., we do not consider the versions that are seldom used). As we focus on the adaptive streaming service, r is always larger than 1. As such, we have overall $m*r$ different versions of all video segments. Each segment is denoted as s_{ij} , where $i = 1, \dots, m$, refers to its popularity rank, and $j = 1, \dots, r$, refers to its bitrate rank.

We assume the aggregated user requests towards those m different segments follow the Zipf-like distribution [67, 68]. We also adopt the assumption from [34, 69], that the user requests independently arrive at each edge server in the media cloud, following the independent reference model [70]. Thus, the probability of requesting the j -th bitrate version of the i -th popular segment s_{ij} at an edge server is,

$$p(s_{ij}) = \frac{p_j/i^\alpha}{\sum_{k=1}^m (1/k^\alpha)} = \frac{p_j/i^\alpha}{H_{m,\alpha}}, i = 1, \dots, m, \quad (4.1)$$

where $H_{M,\alpha} = \sum_{l=1}^m (1/l^\alpha)$ is the m -th generalized harmonic number, and α is the Zipf exponent characterizing the distribution, where α must be positive. A large α indicates more requests on popular segments and less requests on unpopular ones. Typically, α is between 0.5 and 1.5. We adopt the assumption from [34] that, α can be closed to one, but in general not exactly equal to one.

4.2.3.3 Request Processing Model

We adopt the partial in-network transcoding model from [71] as shown in Figure 4.3. Under this scheme, each edge server strategically caches all bitrate versions for a few top popular video segments, and only keeps the highest bitrate versions for some other less popular segments, constrained by the allocated cache space. Mathematically, we introduce a parameter $x_i \in [0, c_i/b_s]$, where c_i is the allocated cache space for node i , and b_s is the total size of all bitrate versions of one video segment. Under this configuration, node i caches all bitrate versions of the x_i most popular segments, and only the highest

Figure 4.3: Illustration of partial transcoding scheme. Each edge server caches all bitrate versions for top x popular segments, and only caches the highest bitrate version for the rests, subject to the allocated cache space.

version of the segments, whose popularity rank is between x_i and h_i . In this case, h_i can be calculated by,

$$h_i = x_i + (c_i - x_i b_s) / b_h, \quad (4.2)$$

where b_h is the size of the highest bitrate version. Note that we assume that the caching space at each node c_i is always much smaller than the total size of all video segments, and the transcoding setting x_i cannot be larger than the total amount of video segments (i.e., $c_i < m b_s$).

In this way, there would be three possible states to serve each user request at node i , as follows.

- There is an *Exact Cache Hit*, if the requested video segment is available in the local cache space. In this case, this request can be directly served, without incurring any extra expense on bandwidth or computing resources.
- There is a *Transcoding Hit*, if the popularity rank of the requested video segment is in the range of x_i and h_i , and it is not the highest bitrate version. In this case, this request still can be served locally, based on real-time transcoding, which involves the usage of computing resources at the edge server. In practice, when the local transcoding hit occurs, it will evaluate whether the local transcoding cost is higher

than retrieving the requested segment directly from the origin server, and pick the operation with lower cost.

- There is a *Local Cache Miss*, if the requested video segment is not among the top h_i ones. In this case, we adopt a simple yet practical mechanism that, all those cache misses will be directly forwarded to the origin server. This costs a certain amount of bandwidth resources to transmit the required segment from the origin server to the edge.

Therefore, we have the local cache hit ratio, which includes both exact cache hit and transcoding hit, as,

$$p_i^{hit} = \sum_{k=1}^{h_i} \sum_{j=1}^r p(s_{kj}) = \frac{H_{h_i, \alpha}}{H_{m, \alpha}}, \quad (4.3)$$

and the probability of local transcoding hit as,

$$p_i^{tr} = \sum_{k=x_i+1}^{h_i} \sum_{j=2}^r p(s_{kj}) = \frac{(1 - p_1)(H_{h_i, \alpha} - H_{x_i, \alpha})}{H_{m, \alpha}}. \quad (4.4)$$

Besides, we can also easily have the local cache miss ratio as,

$$p_i^{miss} = 1 - p_i^{hit}. \quad (4.5)$$

4.2.3.4 Cost Model

Based on the request processing model, there would be three different costs from the view of cloud service provider, including a caching cost, transcoding cost, and a bandwidth cost. In particular, the caching cost is charged by allocating cache spaces to each edge server. The transcoding cost is incurred when the higher quality version is transcoded into a lower bitrate version. And the bandwidth cost is incurred by transmitting the requested streaming data from the source (i.e., either the edge server holding a copy in the cache or the origin server) to the user along its delivery path. Such transmission cost usually comes from high-end routers and fibre cost [43].

The expectation of the caching cost for holding popular video segments at node i is linear to the allocated space as,

$$\mathbb{E}[C_i^{ca}] = w_{ca}c_i, \quad (4.6)$$

where w_{ca} refers to the unit cost of allocating caching resources to keep the copies of video segments at edge servers.

The bandwidth cost within the media cloud only incurs by serving those cache misses. As a result, its expectation serve an user request from node i is in proportion to the amount of transmitted traffic for cache misses. Mathematically, it can be calculated by summing up the bandwidth cost on each link along with its distribution path as,

$$\mathbb{E}[C_i^{bw}] = w_{bw}p_i^{miss}d_ib_m, \quad (4.7)$$

where w_{bw} is the unit cost to transmit video streaming for one hop, d_i is the traversed hop distance of serving the cache miss, and $b_m = \sum_{i=1}^r b_ip_i$ is the average size of a video segment among all bitrate versions.

The expectation of the transcoding cost by serving a request originated from node i is given by,

$$\mathbb{E}[C_i^{tr}] = \min\{w_{tr}b'_m, w_{bw}d_ib'_m\}p_i^{tr}, \quad (4.8)$$

where w_{tr} is the unit cost to transcode video segments, p_i^{tr} is the transcoding ratio at node i with respect to local transcoding configuration x_i , and $b'_m = \sum_{i=2}^r b_ip_i$ refers to the average size of requested segments that need to be transcoded locally. Here we adopt the transcoding cost model from [35, 72], that the cost is proportional to the output bitrate. Without loss of generality, this case may also stand for the bandwidth cost of retrieving the requested segment directly from the origin server, if local transcoding cost is higher.

4.2.4 Problem Formulation

Using the system models, we formulate a constrained optimization problem, with the objective to minimize the total operational cost to serve one request at each node, as,

$$\min_{x_i, c_i} \quad \mathbb{E}[C_i^{tr}] + \mathbb{E}[C_i^{bw}] + \mathbb{E}[C_i^{ca}], \quad (4.9)$$

$$s.t. \quad 0 \leq x_i \leq \frac{c_i}{b_s}, \quad (4.10)$$

where there are two decision variables, including the in-network transcoding configuration parameter x_i , and the allocated cache space c_i . The constraint (4.10) captures the relationship between them. Note that, in our solution, all cache misses are directly forwarded to the origin server. This implies the decisions on one node is independent with the ones on any other nodes. As a result, the optimal solutions for every node, also lead to the network-wide cost minimization.

4.3 Analytical Solutions

This section adopts a two-step approach to analytically solve the optimization problem. First, for a given cache space c_i , we derive the optimal transcoding configuration x_i^* with respect to c_i . Second, we substitute x_i^* into the original objective function (5.9), and further obtain a closed-form of the optimal allocated cache space c_i^* . Finally, we use the obtained results to theoretically quantify the performance gain.

4.3.1 Optimal Transcoding Configuration

We first investigate the sub-problem that, given a fixed cache space c_i , attempts to minimize the combined bandwidth and transcoding cost. Specifically, we still formulate it as a constrained optimization problem, which is given by,

$$\min_{x_i} \quad \mathbb{E}[C_i^{tr}] + \mathbb{E}[C_i^{bw}], \quad (4.11)$$

$$s.t. \quad 0 \leq x_i \leq \frac{c_i}{b_s}. \quad (4.12)$$

To analytically solve this problem, we check the convexity of both the objective function (4.11), and the constraint (4.12) with respect to the decision variable x_i . In particular, we focus on the existence of their first order derivative and the positivity of their second derivative. In this way, we mathematically prove the following theorem.

Theorem 4.1 *The combined bandwidth and transcoding cost minimization problem is a convex optimization problem in terms of the local configuration setting x_i at each node.*

Proof: The key idea is to examine the convexity of the objective function, i.e., Eq. (4.11), by checking the existence of its first-order derivative and the positivity of its second-order derivative. Besides, we also show that all the constraints, i.e., Eq. (4.12), are affine. See Appendix B.1 for a completed proof. ■

As a result, theorem 4.1 allows us to apply the convex optimization method to reach the closed-form solution, as stated in the following theorem.

Theorem 4.2 *The optimal transcoding configuration in terms of the allocated local cache space c_i at node i is,*

$$x_i^* = \min\left\{\frac{c_i}{b_h \eta^{1/\alpha} + b_s - b_h}, \frac{c_i}{b_s}\right\}, \quad (4.13)$$

where we introduce $\eta = (\frac{a_2}{a_1} - 1)^{\frac{b_s - b_h}{b_h}}$, $a_1 = (1 - p_1) \min\{w_{tr}b'_m, w_{bw}d_i b'_m\}$, and $a_2 = w_{bw}d_i b_m$ as constants to simplify the expression.

Proof: Let the first-order derivative of the objective function (4.11) equal to zero, we have,

$$x_i^{-\alpha} = \eta h_i^{-\alpha}, \quad (4.14)$$

By solving this equation, we further have the optimal transcoding configuration strategy $x_i^* = \frac{c_i}{b_h \eta^{1/\alpha} + b_s - b_h}$ with respect to c_i . In addition, it is also necessary to make sure the generated solution does not violate the constraint (4.12). In summary, we obtain the final solution as Eq. (4.13). ■

4.3.2 Optimal Caching Space Allocation

We further derive the optimal caching space allocation, by substituting x_i^* into the origin optimization problem as,

$$\min_{c_i} \quad \mathbb{E}[C_i^{tr}] + \mathbb{E}[C_i^{bw}] + \mathbb{E}[C_i^{ca}], \quad (4.15)$$

$$s.t. \quad x_i = x_i^*, \quad (4.16)$$

To obtain the analytical solution of this problem, we still start with the proof on its convexity as stated below.

Theorem 4.3 *The original optimization problem is a convex optimization problem in terms of either the local configuration setting x_i or the allocated caching space c_i at each node.*

Proof: The proof follows the same rationale as used in Theorem 4.1. See Appendix B.2 for a completed proof. ■

Similarly, we analytically obtain the optimal caching space at each node as shown in the following theorem,

Theorem 4.4 *By letting the first-order derivative of Eq. (4.15) equal to zero, and solving the equation, we have the optimal caching space c_i^* at node i as,*

$$c_i^* = \left(\frac{a_1 \gamma_1^{1-\alpha} + (a_2 - a_1) \gamma_2^{1-\alpha}}{\gamma_3} \right)^{1/\alpha}, \quad (4.17)$$

where $\gamma_1 = \min\{(b_h \eta)^{1/\alpha} + b_s - b_h)^{-1}, b_s^{-1}\}$, $\gamma_2 = \frac{1-(b_s-b_h)\gamma_1}{b_h}$, and $\gamma_3 = \frac{w_{ca}(m^{1-\alpha}-1)}{1-\alpha}$, where a_1, a_2 are exactly the same as used in Eq. (4.13).

Note that, this optimal caching space c_i^* is independent of another decision variable x_i^* . As a result, in practice, we can first numerically obtain c_i^* for each node according to Eq. (4.17) and those known system parameters, then we can further derive x_i^* by substituting the obtained c_i^* into Eq. (4.13).

4.3.3 Preliminary Theoretical Analysis

In this subsection, we analytically investigate how those system parameters affect the optimal solution and its corresponding operational cost saving. Those preliminary insights are based on the obtained closed form of the optimal local transcoding setting x_i^* and allocated caching space c_i^* .

4.3.3.1 Baselines

For comparison purpose, we study our proposed strategy together with a few existing solutions. For the local transcoding setting, we consider two kinds of baselines, including the traditional all-rate caching scheme [51, 73] (i.e., different bitrate versions of the same segment are viewed as different segment, that $x_i = h_i = c_i/b_s$, and the pure transcoding scheme [74] (i.e., each edge server only caches the highest bitrate version of each segment, and generates other representations via online transcoding, that $x_i = 0$ and c_i/b_h). For the cache space allocation, we choose a static caching space allocation scheme [75], where the caching space is able to keep a certain amount of segments at each node.

4.3.3.2 Impact of Different Parameter

We examine the impact of various system parameters from three different aspects. These insights will be further evaluated throughout extensive simulations in detail in Section 4.4.

First, by checking Eq. (4.8), we find that, when the local transcoding cost is higher than retrieving the requested segments from the origin server (i.e., $w_{tr} \geq w_{bw}d_i$), there is no benefit to only cache the highest version for a given cache space. In this case, the optimal solution degrades to the traditional all-rate caching scheme. Otherwise, the operational cost can be reduced by adopting our partial caching and transcoding scheme. And the cost saving increases as $w_{bw}d_i - w_{tr}$ increases. This implies that, our solution

works better for those nodes far away from the origin server (i.e., d_i is large), or when the bandwidth cost is relatively expensive (i.e., w_{bw}/w_{tr} is large).

Second, by checking Eq. (4.13), we report three observations. First, the optimal transcoding setting x_i^* is linear to the allocated caching space c_i at each node i (i.e., x_i^*/c_i is constant). It is because x_i^*/c_i is the key to balance the trade-off between the transcoding and bandwidth cost. When x_i^*/c_i increases, the bandwidth cost increases whereas the transcoding cost decreases. We can show that it is still a convex optimization problem with respect to x_i/c_i , by substituting x_i with $c_i \frac{x_i}{c_i}$ in the original problem (i.e., Eq. (4.11) and Eq. (4.12)), and following the same rationale as used in the proof of Theorem 4.1. Therefore, there must exist a constant optimal $(x_i/c_i)^*$. This observation provides the mathematical basis to further derive the optimal caching space at each node. Note that, such linear function is only valid when the allocated cache size is smaller than the total size of all video segments in all possible versions (i.e., $c_i < mb_s$). Second, as α increases, x_i^* also increases until the local transcoding strategy degrades to the all-rate caching (i.e., x_i^* increases when $\eta^{1/\alpha} > 1$, and remains c_i/b_s when $\eta^{1/\alpha} \leq 1$). This implies that, the transcoding setting leans to the pure-transcoding scheme when the number of requests towards different segments tends to be uniform (i.e., α tends to zero), whereas it leans to the all-rate caching scheme when only a few popular contents attract most requests (i.e., α is large). Finally, we notice that, this expression does not consist of w_{ca} , indicating the optimal transcoding configuration is independent of the unit caching cost. Indeed, the first step aims to balance the trade-off only between the transcoding and bandwidth cost, whereas the optimal caching space allocation step deals with the trade-off between the caching cost and the combined transcoding and bandwidth cost.

Third, by examining (4.17), we notice that, the optimal allocated caching space c_i^* is in proportion to $w_{ca}^{-1/\alpha}$. This insight clearly quantifies the relationship between the optimal caching space and the unit caching cost.

Table 4.3: Bitrate levels in real adaptive streaming system

Video Type	Setting 1	Setting 2	Setting 3
320x240	0.1 Mbps	—	—
480x360	0.4 Mbps	0.4 Mbps	—
720x480	0.9 Mbps	0.9 Mbps	0.9 Mbps
720x576	1.5 Mbps	1.5 Mbps	1.5 Mbps
1280x720	2.5 Mbps	2.5 Mbps	2.5 Mbps
1920x1080	3.0 Mbps	3.0 Mbps	3.0 Mbps

4.4 Performance Evaluation

In this section, we evaluate the performance of our proposal via extensive simulations, based on the settings that reflect real application scenarios. Our objective is not only to verify the trade-off between three different kinds of cost, but also to understand the impact of various system parameters on the optimal decision. Those insights could ultimately provide operational guidelines for real deployment.

4.4.1 Experimental Setup

For the content model, we assume there are $m = 500,000$ different segments in total, and the length of each segment is normalized into one second. We choose three different bitrate settings from a real adaptive streaming system [76] as shown in Table 5.2. In setting 1, $r = 6$, $b_h = 3.0Mb$, $b_s = 8.4Mb$. Setting 2 does not support the lowest bitrate version in setting 1, where $r = 5$, $b_s = 7.9Mb$. And setting 3 further removes the second lowest bitrate version, where $r = 4$, $b_s = 7.5Mb$.

For the user request model, we artificially generate the requests towards those m different segments by following a Zipf distribution, and towards r different bitrate versions by following a uniform distribution. The request arrivals are assumed to follow Poisson process, and the mean arrival rate is set as one per second. Besides, we assume each edge server initially has nothing in its local cache, and it uses the LFU (Least Frequently Used) replacement algorithm to cache those requested segments. Thus, we also set a

warm-up period for each node, that those served requests are three times of the cached segments (i.e., $3rh_i$). After the warm-up period, we further simulate 7200 request arrivals at each round. Finally, we report the average value of repeating the simulation under each parameter setting for 10 rounds.

For other system parameters, we normalize the unit cost of each resource component as $w_{bw} = 3$ per MB, $w_{tr} = 1$ per MB, and $w_{ca} = 0.01$ per GB per second. Besides, we also use $d_i = 6$ and $\alpha = 0.9$ as an example to perform our simulation. These settings will be adopted in the following experiments, unless stated otherwise.

We implement about 300 lines C# codes as the simulator. This simulator takes the aforementioned parameters, and Equations (4.13), (4.17) as the input to generate the optimal caching and transcoding resource allocation and corresponding operational cost.

4.4.2 Verification

In this subsection, we first verify our analytical solution on the optimal strategy (i.e., x_i^* and c_i^*) by comparing it with numerical results. Here, we use setting 1 as the content bitrate version for the evaluation.

Figure 4.4 demonstrates the numerical solutions on local transcoding setting and caching space, where the results are generated based on an exhaustive manner. In particular, Figure 4.4(a) shows the relationship between the local transcoding strategy (in terms of the fraction of all-rate caching, $x_i b_s / c_i$) and its corresponding operational cost, where the local cache space $c_i = 32GB$ is fixed. And Figure 4.4(b) presents the operational cost of all three resource components, in terms of the allocated caching space (i.e., c_i), where for each specific value, we always set an optimal local transcoding configuration (i.e., $x_i = x_i^*$).

This set of experiments confirm our derived analytical solutions. Specifically, first, we find the numerical solutions (e.g., $x^{*'} = 17400$ in Figure 4.4(a) and $c^{*'} = 97GB$ in Figure

Figure 4.4: Numerical results on the optimal strategy and its operational cost. (a) The combined bandwidth and transcoding cost is a convex function of the transcoding setting x ; (b) The total cost is a convex function of the allocated caching space c .

4.4(b)) fits well with our analytical results (e.g., $x^*=17398$ and $c^*=98.9GB$ calculated by Eq. (4.13) and Eq. (4.17)). Second, from Figure 4.4(a), we observe there is a clear trade-off between the transcoding and bandwidth costs. As x increases, the transcoding cost decreases and the bandwidth cost increases. This verifies our analysis in Section 6.2. Third, Figure 4.4(b) demonstrates the trade-off between the storage cost and the combined bandwidth plus transcoding cost. As c increases, the storage cost increases linearly, but the combined bandwidth and transcoding cost decreases. This verifies our discussions in Section 4.2.1. Finally, we find the total cost in both figures, is a convex function in terms of x_i and c_i , respectively. This is consistent with our theoretical proof on Theorem 4.1 and 4.3.

4.4.3 Cost Savings

In this subsection, we numerically compare our proposed approach with those baselines, in terms of the total operational cost, and the cost savings. We still use setting 1 as the example of content bitrate versions.

Figure 4.5 presents the comparisons on the operational costs under different strategies. Specifically, we consider four different baseline cases, including 1) optimal transcoding

Figure 4.5: Comparisons on the operational costs under different strategies. Comparing with existing solutions, our derived solution always generates the lowest cost.

Figure 4.6: Optimal strategy vs. unit transcoding cost w_{tr} . As the unit transcoding cost increases, (a) the total cost increases; (b) the optimal caching space decreases; (c) the optimal transcoding setting linearly increases.

setting with fixed caching space at 32GB; 2) optimal transcoding setting with fixed caching space at 64GB; 3) optimal caching space allocation setting with pure transcoding scheme (i.e., $x_i = 0$); and 4) optimal caching space allocation setting with all-rate caching scheme (i.e., $x_i = c_i^*/b_s$).

We report the observations from this set of experiments as follows. First, for all edge nodes in the media cloud, although they are with different hop distances to the origin server, our strategy always comes up with the lowest overall operational cost, among all schemes. Moreover, the cost savings are significant, which can be up to 50%. This is again consistent with the optimality of our analytical solution. Second, we find the

Figure 4.7: Optimal strategy vs. unit bandwidth cost w_{bw} . As the unit bandwidth cost increases, (a) the total cost increases in a log scale; (b) the optimal caching space increases linearly; (c) the optimal transcoding setting decreases exponentially.

cost of all strategies goes up, as d_i increases. This growth can be attributed to the increased bandwidth cost to serve a cache miss. Finally, as the hop distance grows, the cost saving compared with all-rate caching increases, whereas the one compared with pure transcoding scheme decreases. To understand this phenomenon, we look at the local transcoding model as shown in Figure 4.3. Specifically, all-rate caching scheme has the lowest cache hit ratio that maximizes the amount of cache missed requests, while pure transcoding scheme operates in an opposite way to minimize those cache misses. Therefore, the gap between the costs of these two schemes gets smaller. Note that, those nodes within 3 hops away from the origin server, will not perform local transcoding, because the transcoding cost is higher than the bandwidth cost of retrieving the content directly from the origin server. Thus, the cost saving compared with pure transcoding scheme increases for $d_i \leq 3$. This confirms our discussion in Section 4.3.3.

4.4.4 Impact of System Parameters

In this subsection, we evaluate the optimal strategy (i.e., the optimal local transcoding configuration and the optimal cache space allocation) and the corresponding operational cost, under different experimental settings. Our objective is to ultimately understand the impacts of various system parameters. In this set of simulations, we still adopt the

same settings as discussed in Section 4.4.1. The difference is that, we will directly use the optimal analytical solution (i.e., x^* and c^*), which are calculated based on Eq. (4.13) and Eq. (4.17).

4.4.4.1 Impact of bitrate versions

We first examine the impact of different bitrate versions, which are summarized from all those simulations (i.e., Figure 4.6 to Figure 4.10). First, setting 3 generates highest operational cost, while setting 1 has the lowest one. This is because the benefit of in-network transcoding gets diminished, as the number of bitrate version decreases. This is also reflected by the transcoding cost model as shown in Eq. (4.8). Second, setting 3 asks for the largest optimal caching space, while setting 1 requires the lowest space. We explain this phenomenon by checking Eq. (4.17). Specifically, as b_s increases, both γ_1 and γ_2 grow, leading to the growth of c^* . Finally, the curves for different bitrate settings are almost overlapped with each other, indicating the number and the size of different bitrate versions has very limited influence on the optimal transcoding setting, in terms of the fraction of all-rate caching (i.e., $b_s x^*/c^*$). This can be traced back to Eq. (4.13), that small changes on b_s will not greatly affect the local transcoding configuration.

4.4.4.2 Unit transcoding cost w_{tr}

Figure 4.6 investigates the impacts made by the unit transcoding cost w_{tr} , by changing its value from 1 to 5 (i.e., $1 < w_{tr} < 5$).

This experiment reveals a list of insights. First, from Figure 4.6(a), it is intuitive that the total operational cost goes up, as the unit transcoding cost increases. Second, from Figure 4.6(b), the optimal cache space allocation decreases, as the unit transcoding cost increases. In this case, the in-network transcoding service becomes more expensive. Therefore, instead of allocating extra caching space to only keep the highest bitrate to improve the transcoding hit ratio, retrieving those requests directly from the origin

Figure 4.8: Optimal strategy vs. unit caching cost w_{ca} . As the unit caching cost increases, (a) the total operational cost increases in a log scale; (b) the optimal caching space decreases exponentially; (c) the optimal transcoding setting remains the same.

Figure 4.9: Optimal strategy vs. hop distance d_i . As the hop distance to the origin server increases, (a) the total cost increases; (b) the optimal caching space increases linearly; (c) the optimal transcoding setting decreases exponentially.

server would yield much less cost. Finally, from Figure 4.6(c), the optimal transcoding setting leans to the all-rate caching scheme, as the unit transcoding cost w_{tr} increases. This is because the higher the transcoding cost is, the less benefits can be generated by performing real-time transcoding on edge servers. In this case, it would be better to increase the exact cache hit ratio, but decrease the transcoding hit ratio at the same time.

4.4.4.3 Unit bandwidth cost w_{bw}

Figure 4.7 presents the impacts from the unit bandwidth cost w_{bw} , by selecting its value range from 0.2 to 3 (i.e., $0.2 < w_{bw} < 3$).

This experiment suggests us to allocate more caching and transcoding resources to serve more requests at edge servers rather than forwarding them to the origin server, when the bandwidth cost is expensive. Specifically, from Figure 4.7(a), the optimal total operational cost increases in a log scale, as w_{bw} increases, implying less cache misses are happening. This observation can be verified by Figure 4.7(b) and 4.7(c) that, as the unit bandwidth cost increases, the the optimal caching space increases linearly, and the optimal fraction of all-rate caching decreases exponentially. This indicates that, more requests are being served locally, but a significant amount of responses are generated based on real-time transcoding.

4.4.4.4 Unit storage cost w_{ca}

Figure 4.8 shows how the unit caching cost w_{ca} influent our optimal scheme and the system performance. In this experiment, w_{ca} ranges from 0.002 to 0.05 (i.e., $0.002 < w_{ca} < 0.05$).

We report the following observations. First, from Figure 4.8(a), the optimal total cost increases in a log scale, as w_{ca} increases. This is because more requests will be sent to the origin server, when the caching cost is expensive. Second, from Figure 4.8(b), the

Figure 4.10: Optimal strategy vs. Zipf exponent α . As the Zipf exponent increases, (a) the total cost decreases exponentially; (b) the optimal caching space decreases exponentially; (c) the optimal transcoding setting increases linearly.

optimal allocated caching space decreases exponentially, as w_{ca} increases. This result indicates more cache space should be allocated, when the unit cost is cheap. And the power-law decreasing trend confirms our analysis in Section 4.3.3. Third, from Figure 4.8(c), the optimal transcoding setting keeps unchanged for different w_{ca} . Indeed, the transcoding setting only deals with the trade-off between the bandwidth and transcoding cost. Therefore, it is not affected by the unit caching cost. Finally, the insights from Figure 4.6, Figure 4.7, and Figure 4.8, again verify the three-way trade-off between the caching, transcoding, and bandwidth cost.

4.4.4.5 Hop distance to the origin server d_i

Figure 4.9 demonstrates the optimal strategy and the corresponding performance for the edge servers, which are with different hop distances from the origin server. Here, we investigate the nodes within 10 hops of the origin server (i.e., $1 < d_i \leq 10$).

We have a few observations from this experiment. First, from Figure 4.9(a), the total operational cost increases as the hop distance goes up, due to the increment on the bandwidth cost to serve those cache misses. Second, from Figure 4.9(b), the optimal cache space allocation should be in proportion to the hop distance. Finally, from Figure 4.9(c), we find that, the optimal fraction of all-rate caching decreases exponentially, as d_i

increases. These two observations suggest us to allocate more cache space to those edge servers far away from the origin server, and configure them to perform more transcoding tasks. Note that, when $d_i \leq 3$, our optimal transcoding strategy degrades to all-rate caching scheme, because the real-time in-network transcoding cost is higher than the bandwidth cost of serving the requests by the origin server. These insights verify our analysis in Section 4.3.3.

4.4.4.6 User request pattern α

Figure 4.10 presents the impacts from the user request pattern, in terms of the Zipf exponent α . According to the findings from [77], that α is around 1 in practice, we change its value from 0.8 to 1.3.

This experiment indicates our strategy works better for a large α , where there are relatively more requests on very popular segments, and fewer requests on those unpopular ones. Specifically, from Figure 4.10(a), we find the total cost decreases exponentially, as α increases. This is because more requests can be served directly by edge servers, based on both the exact cache hit and the transcoding cache hit. Moreover, from Figure 4.10(b), the optimal cache space also decreases exponentially, because each edge server only needs to keep those very popular ones for a large α (i.e., only a small amount of segments are very popular, while the rests are not popular at all). Finally, from Figure 4.10(c), the optimal transcoding setting leans to the all-rate caching for large α . This can still be attributed to the trade-off between the caching and transcoding cost. On one hand, in-network transcoding function increases the local cache hit ratio at an expense of transcoding cost. On the other hand, such transcoding cost may overwhelm the caching cost of directly keeping all bitrate versions, when α is large. As a result, in this case, it is better to allocate more cache space for all-rate caching.

4.5 Conclusion

In this chapter, we studied the optimal transcoding and caching allocation scheme in media cloud, to minimize the total operational cost of delivering on-demand adaptive video streaming. In particular, we examined a three-way trade-off between the caching, transcoding, and bandwidth cost. To balance this trade-off, we formulated an optimization problem to find the optimal strategy. By using the convex optimization framework, we adopt a two-step method to analytically derive the closed form of the optimal transcoding setting and caching space, respectively, for every edge server. The obtained solution was verified throughout extensive simulations. The results showed that, our approach achieves significant cost savings, compared with a few existing methods. In addition, we also found that, the optimal strategy and its benefits can be affected by the unit cost of different resource, the hop distance to the origin server, the users' request pattern, and the settings of different bitrate version for one segment. These insights provide operational guidelines to orchestrate cost efficient adaptive video streaming services over media cloud.

Chapter 5

Cost Efficient Orchestration of Caching, Transcoding and Routing

5.1 Introduction

Software-defined networking (SDN) and network function virtualization (NFV) [5] have emerged as the most promising candidates for improving network function and protocol programmability and dynamic adjustment of network resources [78]. They introduce a new dimension to efficiently orchestrate content distribution services.. Specifically, NFV implements network functions (e.g., caching and transcoding) in software, that can be run on virtual machines and migrated among various locations in the network. This enables the elastic management of networking and computing resources. Combined with SDN, NFV provides an opportunity to address those challenges of efficient video distribution. Nevertheless, this solution space has not been well investigated thus far.

In this chapter, we aim to improve the cost efficiency of video distribution over cloud centric media network, by jointly considering the NFV-enabled storage and computing resource allocation, and the SDN-enabled routing. In particular, we consider the orchestration of routing, caching and transcoding functions. At each node, the caching function brings the contents to a closer place to users, saving the bandwidth cost. Its benefit for distributing adaptive video can be further increased by keeping only the highest bitrate,

and transcoding other versions on the fly upon request. And at each link, the routing function schedules the traffic load, balancing the traffic network-wide. Therefore, the media cloud operator needs to decide which items to cache or transcode, and how to coordinate these functions with appropriate routing policy in the network.

Our approach iteratively proceeds in two phases. First, in the resource allocation phase, we study how to optimally allocate a fixed amount of virtualized in-network storage and computing resources over the whole network, with an objective to maximize overall cache hits. However, this phase may fail to take congestion into consideration, resulting in tremendous networking cost. As a result, we need an additional network routing phase, to find the optimal routing matrix that balances the traffic flow over each link, for a given resource placement strategy. Its objective is to minimize the total networking cost, based on the traffic engineering model. Finally, we show that, the iterative repeat of these two phases converges to the joint optimal resource allocation and routing.

The contributions of this chapter are multi-fold, including:

- We systematically build a set of models to describe the adaptive video distribution system over NFV and SDN enabled media cloud. Based on them, we formulate an optimization problem, to jointly optimize the virtualized resource allocation and the routing control.
- We propose a two-step iterative approach, to find the joint optimal configuration on the virtualized resource allocation and its corresponding routing policy. In addition, we also present the analytical proof on the convergence and optimality of our proposed approach.
- Through extensive simulations, we show that our approach quickly converges to its optimum. It achieves significant cost savings, compared with the optimal solution over either resource allocation or routing alone.

- We obtain operational guidelines by examining the joint optimal solution. From the resource allocation aspect, more resource should be allocated to the node with heavier request rate. From the routing aspect, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path alone if there is only one.

The rest of this chapter is organized as follows. Section 6.2 presents the system overview, system models, and the problem formulation. Section 6.3 shows theoretical analysis for the problem, and proposes algorithms to derive the optimal solution. Section 6.4 numerically evaluates the performance based on extensive simulations. Finally, Section 7.2 summarizes this work.

5.2 System Overview & Problem Formulation

In this section, we first give an overview of the adaptive video distribution system over media cloud, to provide necessary background. Then, we introduce our system models and formulate an optimization problem.

5.2.1 System Architecture

Figure 6.1 presents a systematic end-to-end view of delivering adaptive video streaming over an NFV and SDN enabled media cloud. The system consists of three parts,

- On the server side, the origin server publishes all video segments with all candidate formats, which can be delivered to end-users in an on-demand manner.
- On the user side, they connect via access networks to the media cloud from different regions. Due to the diversity of end-devices, the requested video streaming typically uses various bitrates and formats.

Figure 5.1: Architecture of NFV and SDN enabled media cloud. Various network functions including in-network caching and transcoding can be virtualized and dynamically operated among different nodes within the media cloud. In addition, SDN further enables a complete control over the routing matrix of the whole network.

- In the middle, some switches act as backbone service entry points, serving the aggregated user requests from the access network. The cloud service provider can rent virtualized caching and transcoding resources from a resource pool, and dynamically attach them into any set of those switch nodes as virtual appliances based on the NFV concept. In addition, with the SDN technology, the cloud service provider has a full view of the underlying traffic load, and complete control over the routing matrix in a centralized manner.

One critical design objective of such system is to minimize the networking cost incurred by delivering adaptive video streaming. Indeed, such cost highly depends on the joint orchestration of caching and transcoding functions. Specifically, on the one hand, in-network caching keeps popular contents at the edge of network, saving the networking cost. And the transcoding function transcodes the cached contents into different formats with different bitrates, further improving the cache hit ratio. On the other hand, given limited in-network storage and computing resources, the cost saving could be quite small,

if they are not appropriately scheduled. As a result, there is an opportunity to optimize the operational cost of delivering adaptive video streaming, by carefully allocating these two resources, and operating the network routing accordingly.

5.2.2 System Models

This subsection illustrates our system models to mathematically capture key features of the system. For clarity in the discussion, we summarize important notations in table 6.1. Note, we only present the meaning of some small letters in the table, and we will use their boldface form to denote vectors or matrix (e.g., $\mathbf{x}$ with x_i as its i -th component). We also refer letter t as the iteration number (e.g., $\mathbf{R}(t)$) in our iterative approach.

5.2.2.1 Network Model

We model the media cloud network as an undirected graph $G=(V, E)$, where V denotes the set of switches, and E is the set of network links among those switches. Note, in this work, we consider there is only one origin server, which can be accessed via a switch. But this network model can be easily extended to multiple origin servers by connecting them each with a direct link with infinite capacity to a “single super-server”. Moreover, we assume this core network is operated by a single network administrator.

This topology $G(V, E)$ is built upon a set of resources and constraints. Specifically, we assume that, the cloud service provider rents a total amount of caching (i.e., c_{tot} in terms of the cache space) and transcoding (i.e., o_{tot} in terms of virtualized CPU cycles, which are in turn in proportion to the bitrate of transcoded contents) resources over the whole network. Those resources can be dynamically allocated to any set of nodes, running as virtual appliances. Besides, each link $e \in E$ is with limited bandwidth capacity d_e . Moreover, the cloud service provider can choose one or more node-server paths to transmit data between an intermediate node and the origin server, and decide the traffic load at each path subject to the link capacity, by installing the routing matrix $\mathbf{R}$ at the

Table 5.1: Notation table

Symbol	Definition
$G(V, E)$	Network topology with node set V and edge set E .
d_e	Bandwidth capacity at edge e .
c_{tot}	Total amount of caching resource in terms of the size.
o_{tot}	Total amount of transcoding resource in terms of the rate.
R_{lv}	Fraction of traffic on node v 's l^{th} path.
I_{lv}^e	Indicates if node v 's l^{th} path using link e .
m	Total number of segments published by origin server.
r	Total number of frequently accessed bitrate versions.
s_{ij}	j^{th} bitrate version of chunk i .
b_j	Segment size of the j^{th} bitrate version.
b_m	Mean size of all bitrate versions.
λ_v	Aggregated user request arrival rate at node v .
$P(s_{ij})$	Prob. of requesting s_{ij} .
p_j	Prob. of requesting the j^{th} bitrate version.
α	Zipf exponent.
x_v	# different segments can be served by node v .
y_v	# different segments are served by exact cache at node v .
c_v	Required cache space at node v .
o_v	Required transcoding rate at node v .
$f(u_e)$	Convex cost function with respect to link utilization u_e .
p_{hit}^v	Total cache hit ratio at node v .
p_{tr}^v	Transcoding hit ratio at node v .
p_{miss}^v	Cache miss ratio at node v .

SDN controller. For each node-server path, we use $I_{lv}^e \in \{0, 1\}$ to indicate if node v 's l -th path traverses link e . This information can be calculated offline for a known network topology.

5.2.2.2 Content Model

To provide adaptive video streaming services, we assume that all content objects are chunked into m different segments with fixed length, and each segment has r different bitrate versions in different formats, that are frequently accessed. We denote $\mathbf{s} = \{s_{11}, \dots, s_{ij}, \dots, s_{mr}\}$ as the set of all segments with all bitrate versions, where an element s_{ij} refers to the j -th bitrate version of chunk i , that $i = 1, \dots, m$ and $j = 1, \dots, r$. We also denote $\mathbf{b} = \{b_1, \dots, b_r\}$ as the set of ordered segment bitrate, where b_1 and b_r refer to the lowest and the highest bitrate, respectively. As we focus on the adaptive streaming service, r is always larger than 1.

We assume the aggregated user requests at each entry switch towards those m different segments follow Zipf distribution [68, 79]. In addition, we adopt the independent reference model [69, 70], that the request arrival pattern at each entry point is independent, following Poisson process with the arrival rate as λ_v at node v . In this way, we have the probability of requesting s_{ij} at each node as,

$$P(s_{ij}) = \frac{1/i^\alpha}{p_j \sum_{k=1}^m (1/k^\alpha)} = \frac{1/i^\alpha}{p_j H_{m,\alpha}}, i = 1, \dots, m, \quad (5.1)$$

where p_j is the probability of requesting the j -th bitrate of a segment, and $H_{m,\alpha} = \sum_{k=1}^m (1/k^\alpha)$ is the m -th generalized harmonic number, and α is the exponent parameter of the Zipf distribution, that $\alpha > 0$. Typically, α is between 0.5 and 1.5 in practice [34]. A larger α indicates more requests for the popular content and fewer request of the unpopular content.

5.2.2.3 Request Processing Model

We adopt the partial in-network transcoding model from [71], that each node caches all bitrate versions for a few top popular segments, and only the highest version for some segments that are less popular, constrained by the allocated caching space. As a result, there are three possible conditions to serve requests at each node.

- An *exact cache hit* occurs, when the node keeps the exact copy of the requested content in its local cache space. In this case, the node can directly serve the user.
- A *transcoding hit* occurs, when the node has the highest bitrate version that can be used to transcode into the requested version in real-time. In this case, the node is still able to serve the user locally, involving the usage of the computing resource.
- A *local cache miss* occurs, when the node holds neither the exact content nor a transcodable version. In this case, we assume all cache misses will be directly forwarded to the origin server.

Note that, because of the tremendous amount of accessible contents over Internet, those caching resources can never keep all of them within the network [34,51]. As a result, it is necessary to optimize the occurrence of these above conditions by intelligently allocating virtualized resources.

We model these conditions by focusing on the number of segments that can be cached or transcoded at each node. Specifically, we denote x_v as the number of segments, that can be directly served by node v . We further assume node v caches all the bitrate versions of the top y_v popular segments (where $y_v < x_v$), and only the highest bitrate version of the $x_v - y_v$ segments, with its popularity rank between y_v+1 and x_v . Thus, we have the required caching space at node v as,

$$c_v = y_v \sum_{i=1}^r b_i + (x_v - y_v)b_r, \quad (5.2)$$

and the required transcoding rate at node v is,

$$o_v = \lambda_v p_{tr}^v \sum_{j=1}^{r-1} p_j b_j, \quad (5.3)$$

where $\sum_{j=1}^{r-1} p_j b_j$ denotes the mean size of those transcoded versions of one segment, (note, we assume the transcoding resource consumption is in proportion to the output bitrate [80]), and p_{tr}^v refers to the transcoding ratio at node v as,

$$p_{tr}^v = \sum_{i=y_v+1}^{x_v} \sum_{j=1}^{r-1} P(s_{ij}) = \frac{(1 - p_r)(H_{x_v, \alpha} - H_{y_v, \alpha})}{H_{m, \alpha}}. \quad (5.4)$$

Similarly, we have the overall cache hit ratio, including both exact cache hit and transcoding hit, at node v as,

$$p_{hit}^v = \sum_{i=1}^{x_v} \sum_{j=1}^r P(s_{ij}) = \frac{H_{x_v, \alpha}}{H_{m, \alpha}}, \quad (5.5)$$

and the local cache miss ratio at node v as,

$$p_{miss}^v = 1 - p_{hit}^v. \quad (5.6)$$

5.2.3 Problem Formulation

We formulate an optimal scheduling problem, by jointly considering the network routing policy, and the allocation of in-network caching and transcoding functions.

5.2.3.1 Network Routing

We consider the network routing phase that closely reflects the practical operations in IP backbone networks [81–83]. In particular, we define a routing matrix R_{ev} to capture the fraction of flow going to node v that traverses each link e . This routing matrix can be configured in accordance with the known underlying network topology, the placement of in-network caching and transcoding resources, and observed traffic loads. The objective of such configuration is to minimize the networking cost.

Mathematically, we capture this traffic-engineering practice, by building the optimization problem over routing matrix $\mathbf{R}$, for the given caching and transcoding allocation,

$$\min_{\mathbf{R}} \quad \sum_{e \in E} f\left(\sum_{v \in V} \sum_l R_{lv} \lambda_v b_m p_{miss}^v I_{lv}^e / d_e\right), \quad (5.7)$$

$$s.t. \quad \sum_l R_{lv} = 1, \quad R_{lv} \in [0, 1], \quad (5.8)$$

where $b_m = \sum_{j=1}^r p_j b_j$, denotes the mean segment size of all candidate bitrate versions, and $f(u_e)$ is the cost function that is strictly increasing and convex, with respect to the link utilization $u_e = \sum_{v \in V} R_{ev} \lambda_v b_m p_{miss}^v / d_e$, [84, 85]. In practice, f can be an exponential function or a piecewise-linear function that has the similar shape, to penalize the link congestion. Besides, the product of multiplying $\lambda_v b_m p_{miss}^v$ can be understood as the incoming transmission rate to serve cache misses of node v , and $I_{lv}^e \in \{0, 1\}$ indicates if node v 's l -th path uses link e . Finally, the constraint (5.8) indicates the summation of traffic load over all possible paths must be one. By minimizing the total networking cost, this optimization problem implicitly achieves the load balance, by shifting flows from heavily utilized links to those less utilized ones.

5.2.3.2 Virtualized Resource Allocation

The resource allocation phase aims to maximize the overall cache hits for a given routing matrix, so that more requests can be served at the edges, reducing the total traffic load within the media cloud. Specifically, we have the following cache hit maximization problem over $\mathbf{x}$ and $\mathbf{y}$ as,

$$\max_{\mathbf{x}, \mathbf{y}} \quad g_0 = \sum_{v \in V} \lambda_v p_{hit}^v, \quad (5.9)$$

$$s.t. \quad g_1 = \sum_{v \in V} c_v \leq c_{tot}, \quad (5.10)$$

$$g_2 = \sum_{v \in V} o_v \leq o_{tot}, \quad (5.11)$$

$$\forall e \in E \quad g_i = \sum_{v \in V} \sum_l R_{lv} \lambda_v b_m p_{miss}^v I_{lv}^e \leq d_e, \quad (5.12)$$

where Eq. (5.10) (i.e., g_1) captures the total caching space constraint, Eq. (5.11) (i.e., g_2) captures the total transcoding resource constraint, and Eq. (5.12) (i.e., g_i , $i = 3, \dots, |E|+2$) captures the bandwidth constraint at each link. By solving this optimization problem, we aim to update the allocation of in-network caching and transcoding functions with a lower cost.

5.2.3.3 Joint Problem

By jointly investigating the network routing and resource allocation problem, we propose a two-step approach in a feed-back loop.

First, at time $t + 1$, the optimal network routing step computes new routing matrix based on the previous resource placement scheme $\mathbf{x}$ and $\mathbf{y}$ at time t as,

$$\mathbf{R}(t + 1) = \arg \min_R \sum_{e \in E} f\left(\sum_{v \in V} \sum_l R_{lv} \lambda_v b_m p_{miss}^v I_{lv}^e / d_e\right), \quad (5.13)$$

under the constraint (5.8).

Then the resource allocation step updates the allocation of in-network caching and transcoding functions in terms of $\mathbf{x}$ and $\mathbf{y}$, based on the obtained routing matrix $\mathbf{R}(t + 1)$ at the previous iteration, by following,

$$\mathbf{x}(t + 1), \mathbf{y}(t + 1) = \arg \max_{\mathbf{x}, \mathbf{y}} \sum_{v \in V} \lambda_v p_{hit}^v, \quad (5.14)$$

$$s.t. \quad \text{Constraint (5.10), (5.11), and (5.12)}. \quad (5.15)$$

The iterations of Eq. (5.13) and Eq. (5.14) repeat over time, with resource allocations adapting to the new routes, and the routing matrix adapting to the updated traffic rates. This process ends when the decision variables $\mathbf{x}$ and $\mathbf{y}$ have converged. The convergence and the optimality of our solution will be verified in the next section. Note, this work only focuses on an one-shot optimization problem for a specific request pattern λ_v and $P(s_{ij})^v$. As the user request patterns change over time, this approach still works by periodically repeating the iteration.

5.3 Analysis & Algorithms

In this section, we first analytically prove our iterative approach converges to a provably stable and optimal solution. Then we present two algorithms to solve this joint problem.

5.3.1 Convergence and Optimality

Our proof consists of three steps. First, we show that either the network routing configuration or the resource allocation phase actually solves a convex optimization problem, respectively. Second, we construct an alternative optimization form, with its Gauss-Seidel method equivalent to the original joint problem. Finally, we provide a sufficient condition to guarantee the convergence to its optimality.

5.3.1.1 Convexity of resource allocation and network routing

First, we check Eq. (5.9) to Eq. (5.12) with respect to $\mathbf{x}$ and $\mathbf{y}$, focusing on the existence of their first order derivative and the positivity/negativity of their second derivative. By doing so, we have the following lemma.

Lemma 5.1 *Each one-shot optimal virtualized resource allocation problem to Eq (5.12) is jointly convex in $\mathbf{y}$ and $\mathbf{z}$, where $z_v = x_v^{1-\alpha}$ is an auxiliary variable.*

Proof: See Appendix C.1 for a completed proof. ■

Besides, by checking Eq. (5.7) and constraint (5.8), we find that, the optimal network routing problem is also a convex optimization problem over $\mathbf{R}$, because the networking cost function f is strictly convex in $\mathbf{R}$ by definition, and the equality constraint is affine in terms of $\mathbf{R}$.

5.3.1.2 Equivalent optimization form

Second, we modify our joint problem, by combining two objectives (i.e., minimizing overall networking cost, and maximizing overall local cache hit ratio) into one function

as,

$$h(\mathbf{R}, \mathbf{y}, \mathbf{z}) = \sum_{e \in E} f\left(\sum_{v \in V} \sum_l \frac{R_{lv} \lambda_v b_m p_{miss}^v I_{lv}^e}{d_e}\right) - \sum_{v \in V} \lambda_v p_{hit}^v.$$

This modified problem is equivalent to a successive optimization over $\mathbf{R}$, then $\mathbf{y}$ and $\mathbf{z}$. Its objective is to make the network routing update to correspond exactly with the optimal resource allocation scheme over $\mathbf{y}$ and $\mathbf{z}$, in Gauss-Seidel algorithm [86]. In this way, we need to prove the solution of this modified problem converges to the optimum of,

$$\arg \min_{\mathbf{R}, \mathbf{y}, \mathbf{z}} \quad h(\mathbf{R}, \mathbf{y}, \mathbf{z}), \quad (5.16)$$

$$s.t. \quad Eq. (5.8), (5.10), (5.11), \text{ and } (5.12). \quad (5.17)$$

5.3.1.3 Convergence of the joint problem

Based on Lemma 5.1 and the modified optimization problem, we complete the convergence proof by obtaining the following theorem.

Theorem 5.1 *The joint optimal resource allocation and network routing problem converges to its optimum.*

Proof: See Appendix C.2 for a completed proof. ■

5.3.2 Algorithms to the Joint Optimal Solution

We propose two alternative approaches to separately reach the optimal solution of the network routing and resource allocation policy at each iteration. Specifically, we adopt KKT (Karush-Kuhn-Tucker) conditions to the optimal network routing, and subgradient method to the optimal resource allocation. These two processes iteratively repeat until the obtained solutions converge. In this way, the results (i.e., $\mathbf{R}$, $\mathbf{x}$, and $\mathbf{y}$) must converge to the optimum of the joint problem, as we have already proved its convergence and optimality.

5.3.2.1 Algorithm to optimal network routing

We use KKT condition to solve this one-shot optimal network routing configuration problem for a given resource allocation. Specifically, the Lagrange function is given by,

$$L(\mathbf{R}, \gamma) = \sum_{e \in E} f\left(\sum_{v \in V} \sum_l R_{lv} C\right) + \gamma\left(\sum_l R_{lv} - 1\right), \quad (5.18)$$

where $C = \lambda_v b_m p_{miss}^v I_{lv}^e / d_e$ denotes the constant part in function f , and γ is the KKT multiplier.

Thus, setting the gradient $\nabla L(\mathbf{R}, \gamma) = 0$ yields the following equations for the optimal solution,

$$\frac{\partial L(\mathbf{R}, \gamma)}{\partial R_{lv}} = \sum_{e \in E} C \frac{\partial f(R_{lv} C)}{\partial R_{lv}} + \gamma = 0, \quad (5.19)$$

$$\frac{\partial L(\mathbf{R}, \gamma)}{\partial \gamma} = \sum_l R_{lv} - 1 = 0, \quad (5.20)$$

$$R_{lv} \mu_{lv} = 0, \quad \forall l \in L, \quad \forall v \in V, \quad (5.21)$$

where Eq. (5.21) is the complementary slackness condition with μ_{lv} as the KKT multipliers, capturing the positive constraint of R_{lv} . In this way, we have $2|L||V| + 1$ equations with $2|L||V| + 1$ unknown variables. As a result, we use fsolver in Matlab to solve these equations, and obtain the optimal solution.

5.3.2.2 Algorithm to optimal resource allocation

We adopt subgradient method as presented by Algorithm 5.1, to find the optimal virtualized resource allocation at each node for a given routing matrix. In particular, the algorithm begins with an initial feasible solution $\mathbf{y}^{(0)}$ and $\mathbf{z}^{(0)}$. At each iteration, it takes a step β_k along with its subgradient $g^{(k)}$ of the objective or one of the constraint functions. The optimal criteria of this process is that, if the current point is feasible, it uses an objective subgradient; otherwise, the algorithm chooses a subgradient of any violated constraint. We repeat the iteration, until it gets converged. The proof on the convergence

and optimality of applying subgradient method to solve constrained convex problem, can be found in [87].

Algorithm 5.1 Subgradient method

Input: Routing matrix $\mathbf{R}(t+1)$

Initial resource allocation $\mathbf{y}^{(0)}$ and $\mathbf{z}^{(0)}$

Output: Optimal resource allocation $\mathbf{y}(t+1)$ and $\mathbf{z}(t+1)$

- 1: initiate $k = 0$
 - 2: **do**
 - 3: **if** $\forall i = 1, \dots, |E| + 2, g_i(\mathbf{z}) < 0$, **then** $g^{(k)} = \nabla g_0(\mathbf{z})$
 - 4: **else** $g^{(k)} = \nabla g_j(\mathbf{y}, \mathbf{z})$, for the j that $g_j(\mathbf{y}, \mathbf{z}) > 0$
 - 5: update $\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} - \beta_k g^{(k)}$
 - 6: update $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \beta_k g^{(k)}$
 - 7: update $k = k + 1$
 - 8: **while** $|z_v^{(k)} - z_v^{(k-1)}| > \epsilon, \exists v \in V$ **or** $g_i(\mathbf{y}, \mathbf{z}) > 0, \exists i = 1, \dots, |E| + 2$
 - 9: return $\mathbf{y}^{(k)}$ and $\mathbf{x}^{(k)} = \mathbf{z}^{(k)1/(1-\alpha)}$ as the optimal solution
-

The complexity of Algorithm 5.1 is $O((|V|+|E|)/\epsilon^2)$. Specifically, at each iteration, we need to check $|E| + 2$ constraints, and correspondingly update $\mathbf{y}$ and $\mathbf{z}$, which contain $|V|$ elements respectively. And the subgradient method takes $O(1/\epsilon^2)$ iterations to converge [87].

5.4 Performance Evaluation

In this section, we evaluate the performance of our proposal via extensive simulations, based on the settings that reflect real application scenarios. The objective is not only to verify the convergence and optimality of our proposal, but also to understand the impact of various system parameters, ultimately obtaining operational guidelines for real deployment.

5.4.1 Parameter Settings

For the network topology model, we use two examples as shown in Figure 5.2, including a simple and symmetric 10-node ring topology, and the real Internet2's layer 3 network

Figure 5.2: Network topologies used in the simulation, including (a) synthetic 10-node ring, (b) real Internet2’s layer 3 network in the U.S specifically the future Internet

Table 5.2: Bitrate versions in real adaptive streaming system

Type	240p	360p	480p	720p	1080p
Bitrate	0.2Mbps	0.5Mbps	1.2Mbps	2.0Mbps	3.0Mbps

in the U.S [88] which was built specifically for research and testing purpose for the future Internet. To facilitate our discussions, all nodes and links are indexed by the number on them. In the ring topology, each source-destination pair has exactly two paths (i.e., clockwise and counter-clockwise). And in Internet2, we choose four paths that have minimum hops among all possible paths for every source-destination pair. As such, we readily have I_{lv}^e for both topologies. In addition, we assume the link capacity follows a truncated Gaussian distribution bounded below by zero [85], where $d_e \sim \mathcal{N}(1, 0.15)$ Gbps. Besides, we replace the networking cost function by $f(u_e) = nu_e^n$, as suggested by some traffic engineering studies [85].

For the content model, we assume there are in total $m = 20,000$ different segments, where each segment has $r = 5$ different bitrate versions for different devices, and the length of each segment is normalized into one second. We use real bitrate data from a real adaptive streaming system [76], as shown in Table 5.2. The popularity of different

segments is artificially generated based on Zipf distribution, and the popularity of different bitrate follows a truncated Gaussian distribution as $p_j \sim \mathcal{N}(0.2, 0.02)$. At each node, we assume the Zipf distribution (i.e., α) is roughly the same, and the request rate also follows a truncated Gaussian distribution [85] as $\lambda_v \sim \mathcal{N}(200, 20)$ requests per second. Finally, we assume the total caching resource is 10% of all segments, and the total transcoding resource can transcode 1% segments in real-time.

We will adopt those settings, and use $f(u_e) = 3u_e^3$, $\alpha = 0.9$ in following experiments, unless otherwise stated.

The simulator is implemented in about 200 lines of Matlab code. It takes the aforementioned parameters as input, and iteratively runs fsolver for optimal network routing phase, and Algorithm (5.1) for optimal resource allocation phase.

5.4.2 Convergence Verification

In this subsection, we evaluate the convergence of our iterative approach, by simulating 100 random configurations and obtaining the mean convergence rate. In particular, In each simulation round, we randomly pick a node as the origin server for both topologies, and start with a uniform caching and transcoding resource allocation over all nodes. The convergence of each simulation round is determined by whether the difference between the newly updated value and the previous one is smaller than $\epsilon = 0.001$.

Figure 5.3 verifies the convergence of our iterative approach, where we count the completion of both allocation and routing phase as one iteration. According to Figure 5.3(a), our proposal successfully converges to the joint optimum within 7 iterations, for all simulation rounds. For both topologies, more than 65% cases get converged within 4 iterations. As an example, Figure 5.3(b) shows how optimal resource allocation at each node changes over iterations in Internet2, where the origin server is set at node 10. It is clear that, after only 3 iterations, our solution quickly converges to its optimum.

Figure 5.3: Convergence performance of our iterative approach. All the testing cases converge to the optimal solution within 7 iterations.

Besides, we notice that, the ring topology needs less iterations than Internet2 on average. This is because when applying Gauss-Seidel type iterative approach, the solution at each iteration follows conjugate directions of its decision variables. Since the ring topology has less possible paths for each node in the routing phase, its system matrix is smaller than the one of Internet2. This leads to the faster convergence for ring topology.

5.4.3 Joint Optimal Solution

This subsection demonstrates the obtained optimal resource allocation and routing strategy, through a few examples. Here, we set node 10 as the origin server in both topologies.

5.4.3.1 Optimal resource allocation scheme

Figure 5.4 illustrates the optimal virtualized resource allocation at each node for both topologies, where the y-axis denotes the fraction of the allocated resource or request rate at each node over the total amount from all nodes (i.e., $\lambda_v / \sum_{v \in V} \lambda_v$, and $x_v / \sum_{v \in V} x_v$). We report two observations and their corresponding insights from those data, as follows.

First, we find that, more resources should be allocated to the node with heavier user request rate. For example, in Figure 5.4(a), node 2 has the lowest request rate, and

Figure 5.4: Optimal virtualized resource allocation. More caching or transcoding resources should be allocated to the node with heavier user request rate, while the two resources are compensatory to each other.

the lowest resource amount. In contrast, node 7 is with the highest request rate, and the highest resource amount. We explain this phenomenon based on the load balance nature of our optimization problem. In particular, on the one hand, given the identical popularity distribution at each node, more requests leads to more local cache misses. On the other hand, by placing more resources, we can reduce those cache misses. As a result, to balance the cache miss traffic at each node, we need to coordinate the resource allocation with the request arrival rate.

Second, the total cache hits can be unchanged, if we increase the total amount of caching resource but reduce the computing resource, and vice versa. Indeed, the in-network transcoding function was initially proposed to increase the local cache hit ratio, in addition to the limited caching space at each node [74]. In practice, this implies, the operator of media cloud can elastically choose different combinations of total caching and computing resource, to maintain the same cost efficiency level.

5.4.3.2 Optimal network routing policy

Figure 5.5 demonstrates the optimal routing matrix of one source destination pair (i.e., the node-server pair) in both topologies. This experiment focuses on the impact of

Figure 5.5: Optimal network routing policy: for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path if there is only one.

underlying topology, and uses the deterministic uniform model by setting the standard deviation of the request arrival rate at each node, and the link capacity at each edge as zero.

The results reveal that, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path if there is only one. For instance, in Figure 5.5(a), all nodes except node 5 have only one shortest path, and they only use it to forward cache misses. Whereas node 5 splits its traffic evenly across two paths, both of which are 5 hops away from the origin server. This also applies to Figure 5.5(b), where node 4, 6, and 9 only use their 1-hop path, whereas node 3 splits the traffic over 4 different paths, which are all 3 hops away from the origin server. The reason for this phenomenon is intuitive that, using the shortest path to transmit data minimizes the traversed hops, and splitting the traffic across multiple shortest paths balances the traffic load at each link.

5.4.4 Performance Comparison

This subsection compares our joint optimal solution with two baselines, where only the optimal resource allocation or the optimal routing is considered. Specifically, the first

Figure 5.6: Total networking cost vs. cost function $f(u_e) = nu_e^n$. The joint optimization solution performs better than those baselines in all simulation cases.

baseline optimizes the virtualized resource allocation, where every node always retrieves cache misses by using only one path with the shortest hop distance to the origin server. While the second baseline optimizes the network routing, where caching and transcoding resources are always uniformly distributed over all nodes. We evaluate their performance with various system parameters, to obtain some operational guidelines. Similarly, we run 10 random configurations for each setting, and only report the average value. Note that, since solutions from either topology almost have the same shape, we only present the results from Internet2, in following discussions.

5.4.4.1 Total cost vs. networking cost function

Figure 6.14 shows the relationship between the total networking cost and the cost function. Specifically, we change the exponent parameter of cost function to simulate different levels of congestion penalty. We present both absolute cost of all three strategies, and the cost saving of our proposal compared with baselines.

The results reveal several insights. First, we find our joint optimization solution performs better than those baselines in all simulation cases, including both Figure 6.14 and Figure 5.7. This verifies the optimality of our method. Second, the total cost in the ring topology is higher than the one in Internet2. This is because that, when

Figure 5.7: Total networking cost vs. Zipf exponent α . The joint optimization solution performs better than those baselines in all simulation cases.

balancing the same total amount of traffic over more edges, the congestion penalty can be successfully released. Third, in Figure 6.14(a), the cost of all the cases goes down, as n increases. We explain this observation, by checking the first derivative of function $f(n)$ as $f(n)' = u_e^n \log u_e$. Since $u_e \in [0, 1]$, there must exist $f(n)' \leq 0$. Thus, we prove that, $f(n)$ is a monotonic decreasing function with respect to n . Finally, according to Figure 6.14(b), as n increases, the cost saving to the optimal routing reduces, while the saving to the optimal allocation increases. The reason is that, when the congestion penalty is heavy (i.e., large n), the routing phase dominates the joint problem. Thus, in this case, the joint optimal solution leans to the routing optimization phase.

5.4.4.2 Total cost vs. popularity distribution

Figure 5.7 plots the total networking cost as a function of Zipf exponent α . In particular, by changing the value α , we are able to simulate different user request patterns (i.e., larger α indicates larger fraction of requests on those very popular segments), and examine its impact on the total cost.

We obtain the following observations from this experiment. First, in Figure 5.7(a), as α increases (i.e., more requests on popular segments), both the joint and the routing scheme yield less cost, while the cost of single optimal resource allocation method grows.

This still can be understood by examining its first derivative with respect to α as $f(\alpha)' = \sum_{v \in V} C \lambda_v (m(x_v^\alpha - x_v) \log m - x_v(m^\alpha - m) \log x_v)$, where C is a positive constant. By substituting $m = 20000$, and $x_v = 220, \forall v \in V$ from the optimal routing baseline case into $f(\alpha)'$, we find $f(\alpha)' > 0$ for $\alpha > 0, \alpha \neq 1$. On the other hand, in optimal allocation baseline and our joint method, x_v is set corresponding to λ_v . Therefore, for some nodes with large λ_v and x_v , there is $(m(x_v^\alpha - x_v) \log m - x_v(m^\alpha - m) \log x_v) < 0$. This leads to their summation $f(\alpha)' < 0$. Second, from Figure 5.7(b), we find the cost saving to the single optimal routing scheme decreases, as α increases. This implies that, when the popularity distribution has small tail, the optimal resource allocation phase dominates the joint problem.

5.5 Conclusion

In this chapter, we jointly investigated the NFV-based resource allocation and SDN-based routing for cost-efficient video distribution over future Internet. Specifically, we propose a two-step iterative approach. First, in the resource allocation phase, we maximize total cache hits by optimally allocating storage and computing resources for a giving routing policy. Second, in the routing phase, we minimize the networking cost by optimally configuring the routing matrix for a given resource placement. Finally, we analytically prove their iterative repeat converges to the joint optimum. Through extensive simulations, we verified the convergence and optimality of our approach. Besides, by examining those numerical results, we got some operational guidelines. From the resource allocation aspect, more resource should be allocated to the node with heavier request rate. From the routing aspect, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path alone if there is only one.

Chapter 6

Cost Efficient Cloud Clone Migration for Cloud Social TV

6.1 Introduction

Lately, TV experience has been dramatically transformed, with the emergence of multi-screen social TV [89, 90]. First, a traditional “laid-back” video watching experience is combined with “lean-forward” social interactions among peer viewers, resulting in a user-centric viewing environment [89]. Second, multi-screen social TV offers ubiquitous and unified services that are available at anytime, anywhere, on any device at an affordable cost, with personalized experiences [90]. Finally, with the latest multi-screen or second-screen technology, users can transfer the ongoing sessions from one device to another, without any service interruption. Nonetheless, given its highly regarded value, large-scale deployment of social TV has been limited, if not totally absent.

In response to this trend, we design and implement a multi-screen social TV system [91, 92] over the Cloud-Centric Media Network, as a proof-of-concept application. It offers *video teleportation* as its salient feature. In particular, one can easily migrate video session back and forward among different devices, with intuitive human-computer interactions. As a result, a seamless multi-screen social experience is achieved.

The enabling technology for *video teleportation* is to instantiate a virtual machine

in the cloud as a cloud clone for each user¹. Specifically, each cloud clone represents one user, serving as his proxy in the cloud, to manage all the associated devices and session information. In addition, the cloud clone also provides video transcoding [93] and advertisement insertion function to the original video streaming, achieving a personalized multi-screen social TV experience for end-users in a scalable and flexible manner.

One critical design objective is to minimize the monetary cost on operating *video teleportation*, potentially making this service affordable to the general public. A possible deployment scenario is to rent cloud resources from a vendor-neutral provider. In this case, we need to intelligently manage the resource rental cost. In particular, the transmission cost is determined by the transmitted content size and its delivery path length. The former one depends on the video source and the way the user consumes contents (e.g., TV or smartphone). The latter one depends on the cloud clone location along the content delivery path. As the user shifts sessions from one device to another, the retargeted content size changes, and the cloud clone might dynamically migrate to a better location to save the cost. Therefore, an optimal cloud clone migration policy should be in place to balance the trade-off between the migration cost and the transmission cost savings.

In this chapter, the contributions are multi-fold, including:

- We formulate the cost-minimization problem as a Markov Decision Process by adopting a Markov chain to model the user watching behavior across TV and smartphone. The objective is to minimize the monetary cost of operating the video teleportation service, by migrating the cloud clone to the best place, as the user shifts his device.
- Under this framework, we first consider a random fixed placement policy and an offline policy as an upper and a lower bound for the optimal cost. Then we follow

¹We will use virtual machine and cloud clone interchangeably in this chapter.

up with a semi-online algorithm, where only the state transition probability is known in advance. Finally, we propose the Q-learning method, which learns the user behavior from the history and makes online decisions at each time slot. This approach is more practical in real system.

- Extensive experiments based on both simulated data and real user behaviors traces are conducted to evaluate those proposed strategies. Our numerical results suggest that, up to 25% monetary cost compared with the random fixed placement can be saved by using Q-learning method in typical scenarios. The cost saving can be affected by the length of content delivery path, the data size of VM migration, and the user behavior pattern. In addition, we show that, the Q-learning based online algorithm, after a short period of warm-up phase, can achieve almost the same performance as the one which knows the user behavior model.

The rest of the chapter is organized as follows. Section 6.2 presents the system architecture and formulates the problem. Section 6.3 proposes four alternative methods to derive the optimal policy. Section 6.4 evaluates the performances of different approaches. Finally, Section 7.2 concludes this chapter.

6.2 System Overview & Problem Formulation

This section first introduces the system implementations. Then we focus on three system models. Finally, we formulate the minimum-cost cloud clone migration problem as a Markov Decision Process. For clarity and ease of reference in the discussion, we summarize the important notations in table 6.1.

6.2.1 System Implementation

In this subsection, we first present the system architecture and its key application (i.e., video teleportation). Then we discuss the detailed design of cloud clone.

Table 6.1: Notation Table

Symbol	Definition
p_t	Prob. of using TV in both the current and the next time slot
p_p	Prob. of using uses phone in both the current and the next time slot
L	Content delivery hops from source to user
B_o	Expected original content size
B_t	Expected retargeted content size (B_t^t for TV, B_t^p for phone)
$\bar{B}_t(x)$	Mean expected retargeted content size during a period x
c_{tr}	Per hop price to transmit per GB data
$C_{tr}(k)$	Transmission cost when cloud clone locates at k hops away to the source
V_{mig}	Migrated data volume
c_m	Additional price to complete each migration
$C_{mig}(k, k')$	Migration cost from node k to node k'
$\mathcal{S}$	System state set
$\mathcal{U}$	User behavior set
$\mathcal{A}$	Cloud clone action set
$\mathcal{P}_{a_t}(s, s')$	State transition prob. from state s to s' after action a_t at time t
$\mathcal{R}_{a_t}(s, s')$	Total monetary cost when state s changes into state s' by fulfilling action a_t at time t
T	Total period that the user is interacting with multi-screen social TV
$\pi(s_t)$	Optimal action policy at time slot t
$Q(s, a)$	Q-value of performing action a at state s
α	Learning rate in Q-learning
γ	Discount rate in Q-learning
θ	Q-value convergence threshold

Figure 6.1: Multi-screen cloud social TV architecture. End users with different devices are connected via residential gateways and access networks to the cloud, which offers cloud clone based virtual overlay content delivery services to distribute live or on-demand TV programs. Such service can be dynamically operated in terms of creating/halting/migration cloud clones.

6.2.1.1 System Architecture

In Figure 6.1, we present a systematic end-to-end view of our cloud multi-screen social TV system. Specifically, the system is built upon Cloud Centric Media Network, which provides on-demand media services, including content distribution, media processing and content adaption. End users with different devices are connected via residential gateways and access networks to the cloud. They can request a live or on-demand TV program from an IPTV source through the cloud via virtual overlay content delivery network. Under this framework, the cloud resources can be dynamically operated in different servers (i.e., cloud nodes) on top of the overlay network, based on a pay-per-use pricing model. Such cloud based CDN paradigm has also been offered by a list of cloud service providers [94,95], to operate media services in a more efficient manner, than the traditional CDN solution [46].

6.2.1.2 Key Feature

Our system offers a highly-touted multi-screen experience via *video teleportation*. With this feature, one user can simultaneously operate multiple devices (e.g., TV and smart-phone) and freely migrate video session back and forward from one device to another without interruption. As a result, the users can always stay connected to TV programs and social interactions.

6.2.1.3 Cloud Clone

The enabling technology for video teleportation is to deploy a *cloud clone* as shown in Figure 6.2 for each user. Specifically, every user is represented by a virtual machine (VM), which serves as his proxy in the cloud to manage all the associated devices and real-time session information (e.g., ongoing programs, information about the active device, the updated viewing history, and most recent video segments). It elastically turns on once its represented user is online, and turns off once the user gets offline. In addition, the VM also dynamically transcodes the original stream from the video source into the one with appropriate format, resolution, and bitrate as its user shifts the devices. Moreover, the VM also offers other functionalities, such as ad insertion to support personalized multi-screen experience. We call such VM as *cloud clone*. And we assume each cloud clone can be operated at any cloud server along any overlay content delivery path.

The location of cloud clone plays an important role on the cost of operating video teleportation. Specifically, the retargeted stream size changes as the user moves around or shift sessions from one device to another. It leads to the changes of the transmission cost, which further depends on the cloud clone location along the delivery path from the source to the user. Thus, there is an opportunity to reduce the operational cost by migrating the cloud clone to its best place.

Figure 6.2: System overview on cloud clone. It serves as the proxy for each user in the cloud, to manage all the associated devices and real-time session information.

6.2.2 System Models

In this subsection, we present three system models to drive the problem formulation on minimizing the operational cost on support the video teleportation feature.

6.2.2.1 Content Processing Model

In this chapter, we assume the cloud clone performs two content processing functions, including advertisement insertion and video transcoding. Through these processing procedures, a content of size B_o will be changed to B_t in the following two cases.

Case 1. $B_o \leq B_t$: When the user is consuming the content on a bigger screen (e.g., TV), B_o would be smaller than the retargeted stream size B_t . In this case, the cloud clone inserts a few personalized video advertisements, by first picking the related advertisement video based on recommendation algorithms, then combining the selected video ad overlay with the targeted quality to the original content [96]. Finally, the

combined video streaming is delivered to the end user. Such method has become a key online monetization strategy to make profit [96]. At the same time, some set-top boxes may not support the latest video format (e.g., H.264 high profile) of the content source. Thus, the cloud clone has to transcode the original content into a compatible one (e.g., H.264 baseline profile), and pick the video ad overlay in the same format. As a result, the combined content size could be bigger than the original one. Note, the transmission cost for the cloud clone to load advertising videos is ignored, since we assume the cloud has already pre-loaded all those data at each node.

Case 2. $B_o > B_t$: When the user is consuming the content on a smaller screen (e.g., smartphone), B_o would be larger than B_t . The main reason is that, the video resolution and bitrate required by such a device are significantly lower than those of the original one. In this case, the cloud clone transcodes the original content into an appropriate format with much smaller resolution and lower bitrate, to fit the output. As a result, regardless of the inserted advertisements with small resolution and low bitrate, the retargeted stream size is still much smaller than the original one.

6.2.2.2 User Behavior Model

We model user behavior across different devices as a Markovian process, which has been widely adopted to characterize a variety of user behaviors on web browsing [97], online social activities [98] and IPTV interactions [99]. Without loss of generality, the user is assumed to switch between two devices (i.e., TV and phone). Thus the model has two corresponding states. This model can be easily extended if there are more devices or states (e.g., when user is simultaneously using TV and phone, it is a new state). Since the collection of devices at home is limited, the number of states could be within a reasonable size.

Figure 6.3-(a) present a snapshot to capture a real use case that an user shifts his attention between TV and smartphone by using video teleportation. In particular, from

(a)

(b)

Figure 6.3: User behavior on session migration by using video teleportation. (a) Real user case of multi-screen cloud social TV; (b) Device switching model as a Markov chain.

the user’s perspective, he can transfer the ongoing programs from TV to smartphone by first scanning the TV screen using smartphone camera, then flipping the phone to trigger the transfer. He can also transfer the sessions back from his smartphone to TV, by simply performing a “throw” gesture. From the system operation aspect, in both cases, the workflow is coordinated by the cloud clone. More specific details can be found from our previous works [91, 92].

Figure 6.3-(b) illustrates a Markov process with two states to model such user behaviors. Specifically, the state transition matrix is completely determined by p_t (the probability in which the user uses TV in both the current and the next time slot) and p_p (the probability in which the user uses smartphone in both the current and the next time slot). Accordingly, we have $p_{tp} = 1 - p_t$, $p_{pt} = 1 - p_p$, where p_{tp} and p_{pt} are the user device switching probability between TV and smartphone.

6.2.2.3 Cost Model

In supporting the cloud clone migration, the media cloud would incur four cost components, including,

- *Transmission Cost* occurs when videos are transmitted from the source to the user. Note, when operating the media cloud, we only need to consider the cloud

network cost, while the cost incurred by access networks (e.g., broadband and wireless network cost) will not be included in this work.

- *Migration Cost* corresponds to the bandwidth cost in which the cloud clone migrates from one node to another within the media cloud.
- *Computing Cost* refers to the consumption of computational resources (e.g., CPU/GPU), when cloud clone processes user requests, transcodes the requested contents into suitable format, and inserts advertisements into the original video. This cost component is a baseline cost, which is invariant of the cloud clone location.
- *Storage Cost* is charged for keeping advertising videos and user sessions. This cost is also a baseline cost, which is constant for different cloud clone locations.

In this chapter, we only focus on the transmission and migration cost, and ignore other two baseline costs which are independent of our decision variable.

In this research, we consider an operational model in which the media service provider rents cloud resources from a vendor-neutral cloud provider (e.g., AWS, Azure). Among the three price models (i.e., on-demand, reserved and spot instances), we adopt the on-demand model, because media service is often real-time. In this case, the spot instance cannot guarantee QoS and the reserved instance could be wasted.

For a chosen content, we assume a content delivery path from the source to the end user. The path length is L , and there are overall $L + 1$ nodes including the source node, the end user node and $L - 1$ intermediate nodes. Each node along the path is indexed as k according to its hop distance to the source. Note the cloud clone can neither locate at the user side nor the content source. We assume the monetary cost incurred by transmission is proportional to the traversed hop distance, and the content size. When the cloud clone is at k , $k \in \{1, \dots, L - 1\}$, the transmission cost in a time slot is,

$$C_{tr}(k) = c_{tr}B_o k + c_{tr}B_t(L - k), \quad (6.1)$$

where c_{tr} is the per hop price to transmit per GB data, B_o is the expected original video size, and B_t is the expected size, after the cloud clone transcodes the content into a smaller one for small screen, or inserts personalized advertisements to convert it into a bigger one for big screen experience.

The monetary cost for cloud clone migration, includes both VM processing cost to initialize and complete the migration, and the transmission cost to send this VM image, which stores online user sessions (e.g., meta data of ongoing program, active device information, updated viewing and behavior history, most recent streaming segments for continuous watching experience, meta data of personalized ads, etc.) to its new destination. Therefore, the monetary charged by migrating cloud clone from one node k to another node k' is,

$$C_{mig}(k, k') = c_{tr}V_{mig}|k - k'| + c_m, \quad k \neq k', \quad (6.2)$$

where $k' \in \{1, \dots, L - 1\}$ denotes the hop distance from the cloud clone to media source after migration, V_{mig} is the migrated data volume, and c_m is the additional price to initialize and complete each migration.

6.2.3 Markov Decision Process Formulation

Using the system models, we introduce Markov decision process (MDP) to formulate the cloud clone migration problem. Specifically, the MDP formulation is a 4-tuple including the system state set, the scheduling action set, the state transition matrix and the cost function.

System States: We define a system state at time slot t as $s_t = (l_t, u_t)$ by jointly considering the location of cloud clone and active user device. $l_t \in \{1, 2, \dots, L - 1\}$ denotes the location of cloud clone at time slot t . $u_t \in \{TV, phone\}$ denotes the active device is TV or smartphone at time slot t . As a result, this system state set $\mathcal{S} = \{s_0, \dots, s_T\}$ presents

both the changes initiated by the user, and the corresponding cloud clone migrations made by the system, where T denotes the period that the user is interacting with our system.

Cloud Clone Migration Actions: The scheduler action set $\mathcal{A} = \{a_0, \dots, a_{T-1}\}$ defines the destination of the cloud clone migration at each time slot. Specifically, we model the action a_t as the migration decision at time slot t , where $a_t = k$, $k \in \{1, \dots, L-1\}$, denotes the cloud clone migrates from its current location l_t to a new place $l_{t+1} = k$. We treat $l_t = l_{t+1}$ as the case that no migration is taken at time slot t .

State Transition: The transition from state $s_t = s$ to $s_{t+1} = s'$ is determined by both the user behavior model and the cloud clone migration decision. Since the user decision is independent of the cloud clone migration, we have the transition probability $\mathcal{P}_{a_t}(s, s')$, that action a in state $s = (l, u)$ at time t will lead to state $s' = (l', u')$ at time slot $t + 1$ as,

$$\begin{aligned} \mathcal{P}_{a_t}(s, s') &= Pr_{(s_{t+1}=(l', u') | s_t=(l, u), l'=a_t)} \\ &= Pr_{(l_{t+1}=l' | l_t=l, l'=a_t)} Pr_{(u_{t+1}=u' | u_t=u)}, \end{aligned} \quad (6.3)$$

where $Pr_{(u_{t+1}=u' | u_t=u)}$ can be obtained from the transition matrix of user behavior model, and $Pr_{(l_{t+1}=l' | l_t=l, l'=a)}$ is determined by the action policy $\pi(s_t) = a_t$.

Cost Function: We define the cost function $\mathcal{R}_{a_t}(s, s')$ as the total monetary cost consumed during the period from time t to $t + 1$. This cost includes both media transmission cost and cloud clone migration cost. As a result, we have,

$$\mathcal{R}_{a_t}(s, s') = C_{tr}(k) + (1 - \delta(k - k'))C_{mig}(k, k'), \quad (6.4)$$

where $\delta(x)$ is the indicator function that,

$$\delta(x) = \begin{cases} 1, & x = 0 \\ 0, & otherwise \end{cases}. \quad (6.5)$$

As a result, $1 - \delta(k - k')$ indicates the decision on whether the migration should be taken at time t .

Optimization Objective: The goal is to find the optimal migration policy $\pi(s_t) = a_t$ to minimize the total cost. It can be formulated as an unconstrained optimization problem over a finite time horizon,

$$\min_{a_t} \sum_{t=0}^{T-1} \mathcal{R}_{a_t}(s_t, s_{t+1}). \quad (6.6)$$

6.3 Migration Strategies for Cloud Clone

In this section, we start with three simple case studies, to illustrate the problem and the fundamental trade-off between transmission cost and migration cost. Following that, we propose four alternative methods to solve this problem.

6.3.1 Case Studies for Cloud Clone Placement

Figure 6.4 illustrates three cases for optimal cloud clone placement. In these cases, we consider the content delivery path as a line topology. Node S serves as the media source, and the black node serves the cloud clone. In particular, we consider the advertisement insertion and the video transcoding function at the cloud clone, through which the original content size of B_o is converted to B_t .

6.3.1.1 Case A

In this case, as shown in Figure 6.4(a), the end user is always consuming the content on one device with a larger screen (i.e. TV) and the retargeted content size B_t^t is larger than that of the original content (i.e., $B_o < B_t^t$). It can be shown that the optimal location for the cloud clone would be the node nearest to the user along the delivery path, to minimize the transmission cost.

(a) Optimal cloud clone placement when $B_o < B_t$

(b) Optimal cloud clone placement when $B_o \geq B_t$

(c) Possible cloud clone scheduling when B_t changes

Figure 6.4: Three case studies on cloud clone migration.

6.3.1.2 Case B

In this case, as shown in Figure 6.4(b), the end user is always consuming the content on one device with a smaller screen (i.e., smartphone) and the retargeted size B_t^p is smaller than that of the original content (i.e., $B_o > B_t^p$). It can be shown that the optimal location for the cloud clone would be the node nearest to the content source along the delivery path, to minimize the transmission cost.

6.3.1.3 Case C

In this case, as shown in Figure 6.4(c), the user switches between two devices when consuming the same content. In the two phases, the optimal location for the cloud clone could change in the process to minimize the total cost.

Throughout these three cases, we observe a clear trade-off between the transmission cost and the migration cost. In particular, on the one hand, once the user switches the device, the cloud clone should immediately migrate to its best location (i.e., Case A and B), to minimize the transmission cost. On the other hand, if the switching behavior is frequent, such policy will generate significant migration cost, which may overwhelm the saving on transmission cost. As a result, we need to strategically schedule the cloud clone migration to balance this trade-off.

In this chapter, we study the cost minimization problem by examining the fundamental trade-off, according to the aforementioned Markovian user pattern. Four alternative methods will be pursued, as explained in next four subsections.

6.3.2 Random Fixed Placement & Cost Upper Bound

In this subsection, we propose a random fixed placement strategy for the cloud clone by assuming it does not migrate during the whole content consumption period T . This is the simplest strategy to this problem, which involves no dynamics. In this case, the total

cost C_{tot} in every time slot is completely decided by the content transmission cost as,

$$C_{tot}(k) = c_{tr}B_0k + c_{tr}\bar{B}_t(T)(L - k), \quad (6.7)$$

where $k \in \{1, \dots, L - 1\}$, and $\bar{B}_t(T)$ is the mean content size per time slot after processed by the cloud clone during T ,

$$\bar{B}_t(T) = (t_p B_t^p + (T - t_p) B_t^t) / T, \quad (6.8)$$

where t_p is the number of time slot that the user spends on phone during the period T .

Algorithm 6.1 Random Fixed Placement Algorithm

Input: a complete user behavior trace set $\mathcal{U}$ during T

Output: the expected cost of random fixed placement C_{tot}

- 1: initialize $C_{tot} = 0$
 - 2: calculate $\bar{B}_t(T)$ by Eq.(6.8)
 - 3: **for** $k = 1$ **to** $L - 1$ **do**
 - 4: $C_{tot} = C_{tot} + B_0k + c_{tr}\bar{B}_t(T)(L - k)$
 - 5: **end for**
 - 6: return $C_{tot} = C_{tot} / (L - 1)$ as the expected cost
-

Algorithm 6.1 describes the details of this method. The purpose of proposing such an algorithm is to get a cost upper bound, and set it as a reference to evaluate other algorithms. Because the system can not know the exact user behavior trace in advance, the scheduler can not decide which location is the optimal one at the beginning. Most probably, the system just randomly picks a location in real implementation. As a result, we aim to obtain the expected total cost of the random fixed placement strategy. We first calculate the cost for all $L - 1$ locations, then dividing the summation by $L - 1$. The results can represent a numerical upper bound for our problem.

We also derive an expected upper bound of the generated monetary cost. According to the Markovian user behavior model, we have the mean state sojourn time that an user spends on TV is,

$$\bar{T}_t = \sum_{x=1}^{\infty} x P_t^{x-1} = \frac{1}{1 - p_t}. \quad (6.9)$$

Similarly, we also have the mean state sojourn time the user spends on smartphone is,

$$\bar{T}_p = \sum_{x=1}^{\infty} x P_p^{x-1} = \frac{1}{1 - p_p}. \quad (6.10)$$

Thus, we have $\bar{B}_t$ as,

$$\bar{B}_t(T) = \frac{\bar{T}_t}{\bar{T}_t + \bar{T}_p} B_t^t + \frac{\bar{T}_p}{\bar{T}_t + \bar{T}_p} B_t^p. \quad (6.11)$$

As a result, we have the expected total monetary cost C_{fix} of the random fixed placement algorithm as,

$$\mathbb{E}[C_{fix}] = \frac{LT}{2} c_{tr}(B_o + \bar{B}_t(T)) \quad (6.12)$$

The complexity of this random fixed algorithm to take an action is $O(1)$. It just randomly picks one place among all the possible L locations, then keeps it unchanged all the time.

6.3.3 Offline Algorithm & Cost Lower Bound

In this subsection, we investigate an offline algorithm, based on dynamic programming approach, to solve the optimal migration problem. Specifically, we assume that the scheduler already has the knowledge of all user behaviors $\mathcal{U} = \{u_0, \dots, u_T\}$ in advance.

Figure 6.5 presents an example of the state transition diagram of this offline algorithm. In this case, all the previous and future user behaviors are deterministic (e.g., in Figure 6.5, $u_0 = TV, u_1 = phone, \dots, u_T = TV$), and all other states are not reachable. Thus, the transition of system states is completely determined by the decision variable a_t at each time slot as,

$$\hat{\mathcal{P}}_{a_t}(s, s') = \begin{cases} 1, & u = u_t, u' = u_{t+1}, l' = a_t; \\ 0, & otherwise \end{cases}. \quad (6.13)$$

We denote $v(s_t)$ as the minimal aggregated cost from s_t to s_T . Using value iteration in the backward induction, we have,

$$v(s_t) = \min_{a_t} \left\{ \sum_{s_{t+1}} \hat{\mathcal{P}}_{a_t}(s_t, s_{t+1}) (\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\}, \quad (6.14)$$

Figure 6.5: An example of the state transition of offline algorithm, where all the previous and future user behaviors are deterministic.

where $v(s_T) = 0$ gives an initial value. By using this iterative equation, we can find the optimal policy, given by,

$$\pi(s_t) = \arg \min_{a_t} \left\{ \sum_{s_{t+1}} \hat{\mathcal{P}}_{a_t}(s_t, s_{t+1}) (\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\}.$$

Algorithm 6.2 presents the details of this method. Specifically, it adopts the dynamic programming approach to iteratively calculate the minimal aggregated cost and the corresponding decision at each time slot. Thus, the complexity of the offline algorithm to take an action is $O(L)$.

Since the scheduler has perfect information from the user side, it follows that the monetary cost resulted from this offline algorithm provides a lower bound for the original cloud clone migration problem. We will verify this result with numerical simulations in

Algorithm 6.2 Offline Algorithm

Input: a complete user behavior trace set $\mathcal{U}$ during T

Output: optimal policy $\pi(s_t)$ at each time slot

- 1: initialize $v(s_T) = 0$
 - 2: calculate $\hat{\mathcal{P}}_{a_t}(s, s')$ from $\mathcal{U}$ by Eq.(6.13)
 - 3: **for** $t = T - 1$ **to** 1 **do**
 - 4: **for** $a_t = 1$ **to** $L - 1$ **do**
 - 5: $v'(s_t) = \sum_{s_{t+1}} \hat{\mathcal{P}}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1}))$
 - 6: **end for**
 - 7: update $v(s_t) = \min_{a_t} v'(s_t)$, $\pi(s_t) = \arg \min_{a_t} v'(s_t)$
 - 8: **end for**
 - 9: return $\Pi = \{\pi(s_1), \dots, \pi(s_{T-1})\}$ as the optimal policy
-

Section 6.4.

6.3.4 Semi-Online Algorithm with Known Transition Model

This subsection proposes an semi-online algorithm with known transition model. In this case, the scheduler only knows the user behavior transition matrix instead of the exact traces in advance, and can observe the user pattern as time evolves.

Figure 6.6 presents a system illustration of the state transition diagram of this non-learning semi-online algorithm. Specifically, it becomes a stochastic environment that, by performing action a_t at time t , the new system state can be either $s' = (a_t, TV)$ or $s' = (a_t, phone)$. This transition probability is defined by the user behavior model as,

$$\mathcal{P}_{a_t}(s, s') = \begin{cases} p_t, & u = TV, u' = TV, l' = a_t; \\ p_p, & u = phone, u' = phone, l' = a_t; \\ 1 - p_t, & u = TV, u' = phone, l' = a_t; \\ 1 - p_p, & u = phone, u' = TV, l' = a_t; \end{cases} \quad (6.15)$$

Similar to the offline method, we define $v(s_t)$ as the minimal expected aggregated cost from s_t to s_T as,

$$v(s_t) = \min_{a_t} \left\{ \sum_{s_{t+1}} \mathcal{P}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\}, \quad (6.16)$$

Figure 6.6: Systematic illustration of the state transition for semi-online algorithm with known transition probability.

Algorithm 6.3 Semi-Online Algorithm

Input: transition probability $\mathcal{P}_{a_t}(s, s')$ of user states

Output: optimal policy $\pi(s_t)$ at each time slot

- 1: initialize $v(s_T) = 0$
 - 2: **for** $t = T - 1$ **to** 1 **do**
 - 3: **for** $a_t = 1$ **to** $L - 1$ **do**
 - 4: $v'(s_t) = \sum_{s_{t+1}} \mathcal{P}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1}))$
 - 5: **end for**
 - 6: update $v(s_t) = \min_{a_t} v'(s_t)$, $\pi(s_t) = \arg \min_{a_t} v'(s_t)$
 - 7: **end for**
 - 8: return $\Pi = \{\pi(s_1), \dots, \pi(s_{T-1})\}$ as the optimal policy
-

By still using a backward Bellman equation, we can derive the optimal policy as,

$$\pi(s_t) = \arg \min_{a_t} \left\{ \sum_{s_{t+1}} \mathcal{P}_{a_t}(s_t, s_{t+1}) (\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\}.$$

Algorithm 6.3 presents the implementation details. Comparing to the offline algorithm, there are only two differences. First, the input changes from a complete user behavior trace set to the transition probability. Second, we replace the deterministic transition model $\hat{\mathcal{P}}$ with the stochastic state transit model $\mathcal{P}$. Therefore, the complexity of the semi-online algorithm to take an action is $O(L)$, which is the same as the offline algorithm.

It can be shown that the minimum cost resulted from this semi-online algorithm falls between the lower bound and upper bound derived in previous two subsection, as verified by performance evaluation in Section 6.4.

6.3.5 Online Algorithm with Reinforcement Learning

This subsection proposes an online algorithm with reinforcement learning. In this case, the scheduler knows neither the exact user behavior traces nor the states transition matrix. Instead, we set an agent to continuously learn the user behaviors and tune the transition parameters from observing the outcomes of its previous actions and past experiences.

Figure 6.7 shows the systematic model on how this interacts with the real world. Specifically, there are two inputs from the real world, including the real user behaviors and the costs incurred by applying cloud clone migration. In each discrete time slot, the learning agent receives the real states from the user side. It then tunes the trained user model and decides the next action based on current and past experiences by estimating the cost of each possible action and selecting the one which generates the minimal cost.

In real implementation, this approach consists of an experience replay phase and an online scheduling/learning phase.

Figure 6.7: Systematic model of online algorithm with reinforcement learning. The learning agent receives real user behaviors and the costs of operating cloud clone are two kinds of input at each time slot, tunes the trained user model, and makes migration decision.

6.3.5.1 Experience Replay

The objective of this experience replay phase is to obtain the optimal policy for each state by implicitly training the user behavior model based on history data. In particular, we set a warm-up stage for each new user. During this stage, we monitor this user's behaviors, but apply a fixed placement algorithm to avoid the expensive cost incurred by the state exploration. By replaying those experiences, we adopt the Temporal Difference (TD) Q-learning [100] to calculate the Q-value and update the knowledge of the internal trained models. Specifically, the Q-value represents the cost of each state-action pair combination, denoted as $Q(s, a)$. We first initialize every Q-value with an arbitrary value. Then an action a is simulated to each observed state s , and a corresponding cost $\mathcal{R}_a(s, s')$ can be obtained. Thus, the Q-value can be iteratively updated by,

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(\gamma \min_a Q(s', a) - \mathcal{R}_a(s, s')), \quad (6.17)$$

where α ($0 < \alpha \leq 1$) is the learning rate to determine how much the newly acquired information will override the old information, γ ($0 \leq \gamma \leq 1$) is the discount factor to trade off the importance of sooner and later cost.

We set T_w as the initial value of the period for the warm-up phase. The warm-up stage will terminate if it converges within this specified initial duration. Otherwise, we extend this phase until all Q-value pairs get converged. For the latter case, it has been shown that, any finite MDP problems, Q-learning will eventually converge for sure [100].

We also use an ϵ -greedy policy to explore every reachable state as much as possible, to ensure the optimality of the generated solution. Specifically, at each iteration of Q-learning phase, instead of always picking the greedily action with the learned optimal Q-value, we also select a random action with a small probability ϵ ($0 \leq \epsilon < 1$) as,

$$\pi(s_t) = \begin{cases} \arg \min_a Q(s, a), & w.p. (1 - \epsilon) \\ \text{random action from } \mathcal{A}, & w.p. \epsilon \end{cases} \quad (6.18)$$

where *w.p.* means ‘with probability’, and ϵ will reduce from its initial value by a reduction rate ϵ' per time slot, until eventually $\epsilon=0$ at the end of the exploration period.

Algorithm 6.4 Reinforcement Learning with Experience Replay

Input: user behavior history $\mathcal{U} = \{u_1, \dots, u_{T_w}\}$ Q-value convergence threshold θ

Output: $Q(s, a)$ for each state-action pair

- 1: initialize all $Q(s, a)$ arbitrarily (e.g., identically 10)
 - 2: **for** $i = 1$ **to** $L - 1$ **do**
 - 3: **for** $t = 1$ **to** $T_w - 1$ **do**
 - 4: simulate $a_t = \pi(s_t)$ according to Eq. (6.18)
 - 5: observe the new state s_{t+1} , and the cost $\mathcal{R}_a(s_t, s_{t+1})$
 - 6: update $Q(s_t, a_t)$ by Eq. (6.17)
 - 7: update $\epsilon = \epsilon - \epsilon'$
 - 8: **end for**
 - 9: **end for**
 - 10: **if** all $|Q'(s, a) - Q(s, a)| < \theta$ **then** return $Q(s, a)$
 - 11: **else** continue to collect user behavior data
 - 12: **end if**
-

Algorithm 6.4 presents the details of this experience replay phase. It implicitly studies the user behaviors by iteratively updating the Q-value. A classic ϵ -greedy exploration function is adopted to select actions at each time slot. The gathered user behavior trace will be replayed for $L - 1$ times, because there are $L - 1$ possible actions at each state.

After the replay phase, we check whether all the Q-values are converged. If so, this phase ends. Otherwise, we need to collect more sample data. In our problem, the number of states (i.e., $2L - 2$) and actions (i.e., $L - 1$) is limited. Therefore, the convergence rate of Q-value should be within an acceptable range. If the problem space becomes larger with more states and actions, we can combine a function approximator [101] into our algorithm.

Algorithm 6.5 Online Algorithm with Q-Learning

Input: current system state s_t at time slot t all learned $Q(s, a)$ from Experience Replay phase

Output: optimal policy $\pi(s_t)$ at each time slot

- 1: **for** $t = 1$ **to** $T - 1$ **do**
 - 2: apply $\pi(s_t) = \arg \min_a Q(s, a)$
 - 3: observe the new state s_{t+1} , and the cost $\mathcal{R}_a(s_t, s_{t+1})$
 - 4: update $Q(s_t, a_t)$ by Eq. (6.17)
 - 5: **end for**
 - 6: return $\Pi = \{\pi(s_1), \dots, \pi(s_{T-1})\}$ as the optimal policy
-

6.3.5.2 Online Scheduling and Learning

This phase uses the trained $Q(s, a)$ generated from the experience replay phase to make optimal decisions. Algorithm 6.5 presents the details of this algorithm. Specifically, at each time slot, the scheduler will choose the action with the minimal Q-value from all state-action pairs, as the optimal policy based on current system state. Therefore, the complexity of the Q-learning based algorithm is $O(L)$. Besides, each updated Q-value will continue to make contribution to learn the latest user behavior pattern. As a result, it forms a lifetime learning process.

This online algorithm with reinforcement learning is the most practical one comparing with others. First, in real system, only the history of user behavior is accessible while either the exact user traces in future or just the precise behavior model for each individual user is difficulty or even impossible to obtain in advance. More importantly,

this method adopts a model-free reinforcement learning technique. It indicates that even the transition probability may not be necessarily deterministic. This approach can still work when the transition probability is a random variable that its probability follows any distribution model. Second, the complexity of the algorithms keeps at a very low level for each cloud clone to take the optimal action, because the number of possible locations along the shortest content delivery path is usually small (e.g., less than 20). This makes it capable to serve thousands of users simultaneously. We will verify its performance in comparison with the ones of other approaches in Section 6.4.

6.4 Performance Evaluation

This section verifies the performance of all those algorithms based on real application scenarios.

6.4.1 Experimental Settings

6.4.1.1 Experimental Approaches

In this work, both simulated user behaviors and real traces are used to drive the evaluations.

For simulated user data, we generate 1000 periods of user behaviors according to the Markov model in section 6.2.2. Each period contains the user behaviors for $T = 2$ hrs. We set each time slot as $t = 10$ seconds, since the video teleportation operation could take around 10 seconds [91]. In this way, there are 720 records in each period, where each record shows the active device at one time slot. And there are in total 720,000 records for one simulated trace.

For real traces, we collected the user behaviors by releasing this multi-screen social TV system to over 200 students at Nanyang Technological University for an internal trial. The users can enjoy the services on a bigger screen via a web-based application portal,

and on their smartphone via a iOS/Android app provided by us. There are around 20 active users among all registered users. The total interaction time from each of them is over 50 hours. We collect behavior traces from all those active users over the last one month (i.e., 08-Feb-2014 to 07-Mar-2014), and use them as the input of our algorithm.

We implement about 1500 lines of C# code to realize all the four algorithms. It takes both simulated and real user data trace as the input, and run all the algorithms for 1000 times based on each trace. When we run the Q-learning algorithm, only the average cost from the online scheduling period will be reported. When we run other algorithms, we report the average cost based on all those 1000 times.

6.4.1.2 Parameter Settings

We adopt the price information from Amazon [102]. Specifically, the per-hop price to transmit one Gigabyte data is $c_{tr} = 0.12$ USD/GB, the price to migrate cloud clone image is $c_m = 0.02$ USD, if we use the extra small VM instance to implement cloud clone.

We use Cisco's Media Experience Engine [103] as a reference to obtain the bitrate information. Specifically, we define $b_o = 1500$ kbps as the bitrate of live streaming sources with 640x480 resolution delivered by Veoh Network (a popular Internet television provider). We set $b_t^t = 2000$ kbps as the bitrate, after the cloud clone inserts video advertisements into the original video, and transcodes it into AVC (Advanced Video Coding) SD (Standard Definition) output with 640x480 resolution for TV viewing. We set $b_t^p = 400$ kbps as the bitrate when the source is transcoded into iPhone compliant MPEG-4 output with 480x320 resolution for iPhone player. Because a time slot lasts for 10 seconds, we have $B_o = 1.875$ MB, $B_t^t = 2.5$ MB, and $B_t^p = 0.5$ MB in each time slot.

For Q-learning settings, we set the learning rate at $\alpha = 0.5$ to give equal weight to new and old knowledge, the discount factor at $\gamma = 0.99$ to take more future costs into account, and the Q-value coverage threshold at $\theta = 0.001$. We initially set $\epsilon = 0.5$, and the ϵ

Table 6.2: Experimental Parameter Settings

Parameter Definition	Settings
Length of one time slot	$t = 10$ seconds
Per-hop price to transmit per GB data	$c_{tr} = 0.12$ USD/GB
Additional price to complete migration	$c_m = 0.02$ USD
Expected original content size	$B_o = 1.875$ MB/slot
Expected content size to smartphone	$B_t^p = 0.5$ MB/slot
Expected content size to TV	$B_t^t = 2.5$ MB/slot
Q-Learning learning rate	$\alpha = 0.5$
Q-learning discount factor	$\gamma = 0.99$
Q-value convergence threshold	$\theta = 0.001$

reduction rate is $\epsilon' = 0.0005$ per time slot. We set the initial value of the warm-up period as the first 4000 records for the simulated user behavior input, and the first 10 hours (i.e., 3600 time slots) for real traces of each user. By tracking the convergence performance of Q-learning algorithm over all evaluation sets, we find the average convergence period is around 5080 transitions. Table 6.2 summarizes these experimental settings in detail.

6.4.2 Monetary Cost Evaluation

This subsection evaluates the monetary cost resulted from different algorithms with various parameters. Our focus is to understand the impact of those parameters, ultimately obtaining operational guidelines for the system implementation.

6.4.2.1 Monetary Cost vs. Delivery Path Length

We then investigate the monetary cost as a function of the content delivery path length L . Figure 6.8 shows the monetary cost and the cost savings compared to the random fixed placement, based on simulated data, where $p_t = p_p = 0.99$. And Figure 6.9 presents the same metrics based on real user traces. In this experiment, we set the VM migration size as $V_{mig} = 8$ MB.

We have a few observations from these two figures. First, the two figures present almost the same trend. It implicitly verifies that our Markov chain based user behavior

Figure 6.8: Minimal monetary cost vs. delivery path (simulated user behaviors).

model can represent the real user behaviors. This can also be observed in Figure 6.10 and Figure 6.11. Second, the monetary cost resulted from the semi-online method and the Q-learning method always falls between the ones resulted from the random fixed placement policy and the offline algorithm. It fits well with our analysis in Section 6.3. This observation can be also applied to Figure 6.10, Figure 6.11 and Figure 6.12. Third, the monetary cost resulted from all the four algorithms increases almost linearly as the delivery path length L increases. This can be traced by the linear cost model as in Eq. (6.1) and (6.2). Finally, compared to the random fixed placement method, our algorithm provides significant cost saving, which increases as the length of the content delivery path increases. For example, the cost saving of Q-learning algorithm is less than 5% when $L=4$. And it reaches more than 12% when $L=17$.

6.4.2.2 Monetary Cost vs. Migration Size

We first investigate the relationship between the overall monetary cost and the VM migration size V_{mig} . Figure 6.10 shows the monetary cost and cost savings compared to the random fixed placement, based on simulated data, where $p_t = p_p = 0.99$. And Figure 6.11 illustrates the same metrics based on real user traces. Here, we set the content delivery path length $L=10$.

Figure 6.9: Minimal monetary cost vs. delivery path (real traces).

This experiment reveals a few insights. First, the cost resulted from the random fixed method remains the same when V_{mig} changes, because there is no migration based on this policy. This observation can be also applied to Figure 6.12 for the same reason. Second, a threshold effect is observed, where the monetary cost remains constant when the migration size is beyond the threshold (e.g., $V_{mig} \approx 120$ MB in the simulated based experiment). On the one hand, when $V_{mig} < 120$ MB, as V_{mig} increases, the migration cost grows. On the other hand, when $V_{mig} > 120$ MB, the monetary cost remains almost constant, and the cost savings of the semi-online method and the Q-learning method are getting closed to the one of the offline approach (i.e., the cost lower bound). This effect can be understood as follows. When the migration size is beyond a threshold, the migration cost would be higher than the minimum transmission cost when the cloud clone locates in the best location. As a result, it would be better if the cloud clone stays at its current location without any migration. And the optimal migration policy will eventually reduce to the case of the optimal fixed placement, as the migration size continues to increase.

Figure 6.10: Minimal monetary cost vs. migration size (simulated user behaviors)

Figure 6.11: Minimal monetary cost vs. migration size (real traces)

6.4.2.3 Monetary Cost vs. User Behavior

We study the impact of user behavior pattern on the monetary cost by changing the user device switching probability. Figure 6.12 presents the monetary cost and the cost savings compared to the random fixed placement method, as a function of user device switching probability $p_s = p_{tp} = p_{pt}$. In this set of experiment, we set the content delivery path length as $L = 10$, the migration size as $V_{mig} = 8$ MB. And we assume the probability of transiting from phone to phone, is the same with the one of transiting from TV to TV (i.e., $p_p = p_t$).

This experiment also reveals a threshold effect, in which the system behaves differently on the two sides of the threshold (e.g., $p_s \approx 2\%$ in this case). On one hand, when $p_s > 2\%$, the monetary cost and the according cost savings remain almost the same as the user device switching probability changes. Because in this case, the user switches his device too often. Thus, the high migration cost would prevent the cloud clone moving such often. In this way, similar to the case of large migration size in Section 6.4.2.2, the migration policy reduces to the optimal fixed placement policy. On the other hand, when $p_s < 2\%$, the cost of all migration methods grow, and their cost savings decrease accordingly, as the device switching probability p_s increases. This can be understood as follows. When the device switching probability is low, the state sojourn time on each device is long as shown in Eq. (6.9) and (6.10). As a result, the cloud clone can migrate as soon as the user switches his device. Because the transmission cost saving would easily overwhelm the incurred migration cost. As p_s increases, the state sojourn time decreases. In this way, the device switching frequency becomes higher, and the cloud clone would migrate less frequently than that for device switching. Thus, the chance in which the cloud clone is not placed in the optimal location increases, resulting in a higher transmission cost.

We also notice that, when the transition probability is extremely low (e.g., $p_s < 0.1\%$ in this case), the cost and its saving of the offline method remains constant. This flat

Figure 6.12: Minimal monetary cost vs. simulated user behavior patterns

curve can be understood by examining a saving function $\Delta(k, k')$ by migrating cloud clone from location k to k' , defined as,

$$\Delta(k, k') = t_m(C_{tr}(k') - C_{tr}(k)) - C_{mig}(k, k'), \quad (6.19)$$

where t_m is the time period from now to the next migration (or the end of period T if there is no further migration). By substituting Eq. (6.1) and (6.2) into Eq. (6.19), we obtain,

$$\Delta(k, k') = \begin{cases} c_{tr}(k' - k)(t_m(\bar{B}_t(t_m) - B_o) - V_{mig}) - c_m, & k' \geq k \\ c_{tr}(k' - k)(t_m(\bar{B}_t(t_m) - B_o) + V_{mig}) - c_m, & k' < k \end{cases}$$

where $\bar{B}_t(t_m)$ is the mean retargeted content size during t_m ,

$$\bar{B}_t(t_m) = (t_p B_t^p + (t_m - t_p) B_t^t) / t_m, \quad (6.20)$$

and t_p is the time the user spends on phone during t_m . We can see that, in this case, the cost saving is determined by $(\bar{B}_t(t_m) - B_o)$, which is in turn determined by t_p . Since t_p can not vary in a large scale when $p_s < 0.1\%$, the monetary cost and the cost saving are almost the same, when p_s changes.

Finally, we find the cost saving of Q-learning based algorithm can reach nearly 25% when $p_s < 0.01\%$, where its performance gets closed to the ones of offline and semi-online algorithm. In this case, if the service would be offered to 100 users for one year,

Figure 6.13: Examples of the migration scheduling based on Q-learning method.

the monetary cost saving can be as much as \$250,000 per year. The reason is that, when the switching probability is small, the time spent on one device is more likely to dominate the other. In addition, the sojourn time on each device becomes longer. Therefore, such pattern is easier to be learnt, and future user behaviors following this pattern is easier to be predicted.

6.4.3 Optimal Migration Policies

In this subsection, we investigate the optimal migration scheduling generated by Q-learning algorithm through a few examples as shown in Figure 6.13.

First, Figure 6.13-(a) and Figure 6.13-(b), investigate the impact of content delivery path length, based on the real trace over latest 1000 time slots from User A, who switches his device the least frequently among all active users. We find that, when the delivery path length is short (e.g., Figure 6.13-(a)), the cloud clone only migrates for once. While

in Figure 6.13-(b), the cloud clone migrates for three times. The migration is easier to be triggered when the delivery path is longer, because the cost savings generated by migration scheduling are much higher in this case. It verifies our discussions in Section 6.4.2.1.

Second, Figure 6.13-(c) and Figure 6.13-(d) focus on the impact of VM migration size, based on real traces over latest 1000 time slots from User B, who switches his device most frequently among all active users. We find that, when migration size is small (e.g., Figure 6.13-(c)), the cloud clone migrates more frequently, because low migration cost makes the transmission cost savings much easier to cover the incurred migration cost. As the migration size increases, the migration frequency decreases. When $V_{mig} = 144$ MB (i.e., Figure 6.13-(d)), there is no migration at all. This verifies our discussions in Section 6.4.2.2.

Third, Figure 6.13-(e) and Figure 6.13-(f) present the impact of user device switching probability based on simulated data. We find if the user switches the active device too frequently (i.e., 6.13-(f)), there will be much less migrations compared to Figure 6.13-(e). This verifies our discussions in Section 6.4.2.3.

Moreover, we have an interesting observation in all examples, that the optimal cloud clone location is either at the nearest or the furthest node to user, regardless of delivery path length, migration size and user behaviors. This can be understood from the cost saving function $\Delta(k, k')$ as Eq. (6.20). Specifically, the scheduler checks whether $\max \Delta(k, k') > 0$ to trigger a migration. If a migration is required, the place which leads to the maximal cost saving, is selected as the destination. It can be seen that, $\Delta(k, k')$ is a linear function of $(k' - k)$. Therefore, if $\Delta(k, k') > 0$, it is better to migrate the cloud clone to the furthest node to maximize $(k' - k)$, thus the cost saving can be maximized. Otherwise, the cloud clone would stay at its current location with no migration.

Figure 6.14: Comparisons between Q-learning algorithm and existing methods. Q-learning method outperforms the existing ones in all the cases.

6.4.4 Comparisons with Existing Methods

For comparison purpose, we regard the random fixed algorithm as the standard method adopted by existing social TV systems [39, 104]. In addition, we also implement another greedy algorithm inspired by [75], where the cloud clone always migrates to its best place, once the user switches the active device.

Figure 6.14 compares the performance of our Q-learning based algorithm with those two existing algorithms. First, we find the proposed Q-learning method outperforms the existing ones in all the cases. This verifies the effectiveness of our approach. Second, the greedy algorithm generates extremely high costs when the switching probability is high, or the migration size is large. In this case, the migration cost becomes very high,

resulting in tremendous total operational cost. Finally, when the switching probability is very low, the greedy algorithm performs very closed to our scheme, which is better than the fixed one. This implies that when user seldom switches the active device, the greedy scheduling stands for the optimal solution. Otherwise, we need to migrate the cloud clone very carefully to reduce the cost.

6.5 Conclusion

This chapter investigated the problem on minimizing monetary cost via cloud clone migration in multi-screen cloud social TV system. We formulated it as a Markov Decision Process, to balance a trade-off between the transmission and migration cost. Under this framework, we first considered a random fixed placement and an offline algorithm to obtain an upper and lower bound for the optimal cost. We then proposed a semi-online algorithm and a more practical Q-learning method. We use both simulated data and real user traces to evaluate all the four algorithms. The results indicated, up to 25% monetary cost compared with the random fixed placement can be saved in typical use scenarios, by optimally migrating the cloud clone. The cost savings can be affected by the delivery path length, the VM migration size and the user behavior pattern. Moreover, we also found the optimal cloud clone location is either at the nearest or the furthest node to the user. Those insights would offer operational guidelines to deliver cost effective multi-screen social TV services over Cloud Centric Media Network, potentially easing its adoption.

Chapter 7

Summary and Future Work

In this chapter, we summarize this thesis and discuss the future work for cloud centric media network.

7.1 Summary

To minimize the cost of operating Cloud Centric Media Network, this thesis has presented the optimal resource allocations and schedules in different application scenarios. In particular, we first study the cost optimization for video distribution services by optimizing platform services, including content caching, real-time transcoding, and multi-path routing scheme. We then build a proof-of-concept application (i.e., multi-screen cloud social TV) on top of Cloud Centric Media Network, and propose an optimal cloud clone scheduling policy specifically for this application. The details in the thesis are summarized as follows.

7.1.1 Optimization over Platform Services

In Chapter 3, we investigated the cost-optimal content placement problem, and formulated it into a constrained optimization problem, with the objective to minimize the total monetary cost, subject to the resource capacity. We tackled this problem via a two-step strategy. The first step focused on the placement for a single content, which was mapped

into a k -center problem. Using a graph-theoretic approach, we derive and verify a logarithmic relationship between the optimal mean hop distance from viewers to contents, and the optimal number of content replicas. The second step leveraged this analytical result to solve the cost optimization problem, via a feasible direction method. Our analytical approach was substantiated via numerical simulations, using a set of data traces from a top content website. The numerical investigation suggested that, the optimal number of content replica for each title follows a power-law distribution in respect to its popularity rank. Moreover, compared to existing heuristics, our proposed algorithm can obtain the optimal placement strategy, with a lower computational complexity.

In Chapter 4, we considered the distribution of adaptive video streaming over Cloud Centric Media Network, where in-network caching and transcoding resources can be dynamically scheduled in an on-demand fashion. Our objective was to minimize the total operational cost by optimally orchestrating multiple resources. Specifically, we formulated an optimization problem, by examining a three-way trade-off between the caching, transcoding, and bandwidth cost, at each edge server. Then, we adopted a two-step approach to analytically derive the closed-form solution of the optimal transcoding configuration and caching space allocation, respectively, for every edge server. Finally, we verified our solution throughout extensive simulations. The results indicated that, our approach achieves significant cost savings compared with existing methods. In addition, we also found the optimal strategy and its benefits can be affected by a list of system parameters, including the unit cost of different resource, the hop distance to the origin server, the Zipf parameter of users' request patterns, and the settings of different bitrate version for one segment.

In Chapter 5, we introduced Network Function Virtualization (NFV) in conjunction with Software-Defined Networking (SDN) to minimize the operational cost via joint orchestration of caching, transcoding and routing functions in Cloud Centric Media Network. Specifically, we proposed a two-step iterative approach. First, in NFV-based

resource allocation phase, we maximized total cache hits by optimally allocating storage and computing resources for a given routing policy. Second, in SDN-based routing phase, we minimized the networking cost by optimally configuring the routing matrix for a given resource placement. Finally, we analytically proved their iterative repeat converges to the joint optimum. Through extensive simulations, we verified its convergence, and performance gains compared with the optimal solution of either phase alone. By examining numerical results, we obtained some operational guidelines. From the resource allocation aspect, it is better to allocate more resources to the node with heavier request rate. From the routing aspect, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path alone if there is only one.

7.1.2 Optimization for Application Service

In Chapter 6, we implemented a multi-screen social TV system on top of Cloud Centric Media Network, offering video teleportation as an attractive feature. The enabling technology is instantiating a cloud clone to support all media outlets of each user. As the user shifts his attention from one device to the other, the cloud clone might migrate to a better location to reduce its operational cost. Focusing on this specific application, we investigated the cloud clone migration problem, aiming to minimize the monetary cost on operating video teleportation. Specifically, we formulated a Markov Decision Problem, to balance the trade-off between the migration cost and the content transmission cost. Under this framework, four algorithms were proposed to solve this optimization problem. We first characterized an upper and a lower bound for the optimal cost, by considering a random fixed placement and an offline algorithm. We then presented a semi-online and a more practical Q-learning approach to make online decisions. Their performances were evaluated based on both simulated and real user traces. The results showed that,

the Q-learning method achieves up to 25% cost compared to random fixed placement in typical scenarios. The savings are affected by the delivery path length, the migration size and the user behavior pattern. Moreover, our investigations revealed the optimal cloud clone location is either at the nearest or the furthest node to the user along the content delivery path for a single user scenario.

7.2 Future Work

In the future, we will extend the work in the following aspects.

7.2.1 Optimization under Alternative Pricing Models

In this thesis, we only considered the on-demand cost model of utilizing cloud resources in Cloud Centric Media Network. In practice, there are some alternative pricing models in cloud computing paradigm, such as reserved model (i.e., pay a fixed amount of cost to reserve cloud resources in prior to real usage) and spots model (i.e., bid to use cloud resources). In the case that there is less knowledge or inaccurate prediction on real-time user request patterns, on-demand pricing model would be most suitable, because the spot instance cannot guarantee QoS and the reserved instance could be wasted. However, if the user request pattern can be accurately obtained, reserved instances and spot instances have the potential to further reduce the operational cost. We will try to exploit such possibilities in Cloud Centric Media Network in our future work.

7.2.2 Operation over Wireless Access Network

In this thesis, we mainly focused on the operations over the backbone network, and somehow ignore the operations over the access network, especial the unstable wireless cellular/3G/4G/WiFi network. However, as the significant increase of mobile devices and ubiquitous penetration of high speed wireless network, it is necessary to also explore the

optimal operations on wireless access network in cloud environment (e.g., cloud-based radio access network). In particular, we are interested in two future directions. First, we plan to investigate the optimal virtual resource allocation scheme for base stations, to improve its operational cost while meeting certain service-level agreement. Second, it is possible to deploy cloud clones at the access point, to enable a task offloading scheme between the mobile devices and the cloud clone. In this case, we need to consider the unstable nature of wireless channel, so that it would be advantageous to offload some tasks when the channel state is good. As a result, this provides an opportunity to reduce the cost by dynamically offloading tasks to the cloud clone.

7.2.3 Large-scale Deployment and More Applications

In this thesis, we implemented and evaluated our proposed mechanisms mostly based on small-scale testbeds or simulations. In the future, we are interested in deploying those policies and applications to a vendor-neutral cloud provider (e.g., Amazon EC2) to achieve the large-scale deployment and understand those practical implementation challenges. Specifically, for the platform services, we are in the process of implementing a future Internet experimental platform, including using OpenStack [105] as the NFV orchestrator, Open vSwitch [106] as the SDN switch, and OpenDaylight [107] as the SDN controller. For the application service (i.e., multi-screen cloud social TV), we have released it in campus at Nanyang Technological University. We are now in the process of migrating the system to Amazon to provide global access.

Based on our platform, we are also interested in developing and implementing a list of novel media applications (e.g., multi-screen apps, interactive video games, real-time video conference, etc.). As a result, these new proof-of-concept applications can verify the high usability of Cloud Centric Media Network, and demonstrate its flexibility for the deployment of different applications. After that, we will try to adopt the similar research rationale to investigate the performance optimization problem for them.

Appendix A

Cost Efficient Content Placement

A.1 Proof of Property 3.1

In this section, we provide the proof of property 3.1.

For any given sub-tree with N_s node, there is a lower bound of its height h defined by the Moore bound (i.e., $h \geq \lceil \log_{\Delta-1} \frac{\Delta N_s - 2N_s + 2}{\Delta} \rceil$). This bound can be reached, when all the levels are full probably except the last in each sub-tree (i.e., Figure 3.3(b)). Thus, the aggregated hop distance within one sub-tree also reaches its lower bound.

A.2 Proof of Property 3.2

In this section, we provide the proof of property 3.2. Instead of directly proving the statement, we show that any other solution cannot yield the smaller aggregated hop distance than the one generated by the uniform partition.

Assuming there is such an optimal solution, where the node number difference between the largest sub-tree S_i and the smallest sub-tree S_j ($i \neq j$) is $N_i - N_j \geq 2$, and $N_i \geq \lceil N/n \rceil + 1$. According to property 3.1, the lower bound of the height h_k of any sub-tree S_k ($k = 1, \dots, n$), is a monotonic increasing function of its node number N_k , for a given node degree Δ (i.e., $h_k(N_k + 1) \geq h_k(N_k)$). Thus, we have $h_i(N_i - 1) \geq h_j(N_j + 1)$. It indicates by re-partitioning the node at the last level of the largest sub-tree S_i to the last

level of the smallest sub-tree S_j , the aggregated hop distance reduces or at least remains the same. This conflicts with the optimality of this solution, which aims to a lower bound of the aggregated hop distance.

Note, the uniform partition is only a sufficient condition to guarantee the optimality. Indeed, the optimal partition may not be unique, since the leaf nodes may have the equal shortest distance to more than one replica (e.g., in Figure 3.4(b), node 1 is 2 hops away from the root of either sub-tree S_1 or sub-tree S_3). As a result, they can be moved from one sub-tree to another one with the same height, while the aggregated hop distance remains the same.

Appendix B

Cost Efficient In-Network Transcoding

B.1 Proof of Theorem 4.1

In this section, we provide the proof of theorem 4.1. that the combined bandwidth and transcoding cost minimization is a convex optimization problem with respect to the local transcoding configuration x_i at each node.

First, we approximate the local cache hit ratio by using a continuous function as,

$$p_i^{hit} \approx \frac{\int_1^{x_i} t^{-\alpha} dt}{\int_1^m t^{-\alpha} dt} = \frac{x_i^{1-\alpha} - 1}{m^{1-\alpha} - 1}, \alpha > 0, \alpha \neq 1. \quad (\text{B.1})$$

Similarly, we have the probability of local transcoding as,

$$p_i^{tr} \approx \frac{h_i^{1-\alpha} - x_i^{1-\alpha}}{m^{1-\alpha} - 1}(1 - p_1), \alpha > 0, \alpha \neq 1. \quad (\text{B.2})$$

By substituting Eq. (4.8) and (4.7) into Eq. (4.11), we also have the combined transcoding and bandwidth cost function as,

$$C_i^{tot} = \mathbb{E}[C_i^{tr}] + \mathbb{E}[C_i^{bw}] = a_1 p_i^{tr} / (1 - p_1) + a_2 p_i^{miss}, \quad (\text{B.3})$$

where $a_1 = (1 - p_1) \min\{w_{tr} b'_m, w_{bw} d_i b'_m\}$, and $a_2 = w_{bw} d_i b_m$ are two positive constants.

Then we substitute Eq. (B.1) and (B.2) into Eq. (B.3), and derive its first-order derivative as,

$$\frac{\partial C_i^{tot}}{\partial x_i} = \frac{1 - \alpha}{m^{1-\alpha} - 1} ((a_2 - a_1) a_3 h_i^{-\alpha} - a_1 x_i^{-\alpha}), \quad (\text{B.4})$$

where $a_3 = (b_s - b_h)/b_h$ is also a positive constant.

And its second-order derivative can be derived into,

$$\frac{\partial^2 C_i^{tot}}{\partial x_i^2} = \frac{\alpha(1-\alpha)}{m^{1-\alpha}-1} (a_1 x_i^{-\alpha-1} + (a_2 - a_1) a_3^2 h_i^{-\alpha-1}). \quad (\text{B.5})$$

In this way, we need to check the positivity of the second-order derivative as shown in Eq. (B.5), to prove the objective function is convex in terms of x_i . In particular, we have $\frac{\alpha(1-\alpha)}{m^{1-\alpha}-1} > 0$ for $\alpha > 0, \alpha \neq 1$. Then, there is $a_2 > a_1$, because $1 - p_1 < 1$, $w_{bw} d_i b'_m < a_2$ and $a_1 = (1 - p_1) \min\{w_{tr} b'_m, w_{bw} d_i b'_m\}$. Besides, we also find that $a_1 x_i^{-\alpha-1} \geq 0$, $a_3^2 > 0$, and $h_i^{-\alpha-1} > 0$. As a result, $\frac{\partial^2 C_i^{tot}}{\partial x_i^2} \geq 0$ always holds in the context of our problem.

In addition, it is clear that the constraint (4.12) is affine with respect to x_i . Therefore, this completes the proof.

B.2 Proof of Theorem 4.3

In this section, we provide the proof of theorem 4.3. that the total operational cost minimization is a convex optimization problem with respect to the caching space c_i at each node.

First, we re-write Eq. (4.13) into,

$$x_i^* = \gamma_1 c_i, \quad (\text{B.6})$$

where $\gamma_1 = \min\{(b_h \eta^{1/\alpha} + b_s - b_h)^{-1}, b_s^{-1}\}$ is a positive constant. In this way, it is clear that, x_i is linear to c_i .

Then, we substitute Eq. (B.6), and (4.6) into the objective function (4.15) as,

$$\begin{aligned} C_i^{tot} &= \mathbb{E}[C_i^{tr}] + \mathbb{E}[C_i^{bw}] + \mathbb{E}[C_i^{bw}] \\ &= a_1 p_i^{tr} / (1 - p_1) + a_2 p_i^{miss} + w_{ca} c_i, \end{aligned} \quad (\text{B.7})$$

where a_1 and a_2 are exactly the same positive constants, as we used in the proof of Theorem 4.1.

Similarity, we check its first-order derivative as,

$$\frac{\partial C_i^{tot}}{\partial c_i} = \frac{(1-\alpha)c_i^{-\alpha}}{m^{1-\alpha}-1}((a_1-a_2)\gamma_2^{1-\alpha}-a_1\gamma_1^{1-\alpha})+w_{ca}, \quad (\text{B.8})$$

where we introduce a positive constant $\gamma_2 = \frac{1-(b_s-b_h)\gamma_1}{b_h}$ to simplify the expression.

And its second-order derivative is given by,

$$\frac{\partial^2 C_i^{tot}}{\partial c_i^2} = \frac{\alpha(1-\alpha)c_i^{-\alpha-1}}{m^{1-\alpha}-1}(a_1\gamma_1^{1-\alpha}+(a_2-a_1)\gamma_2^{1-\alpha}). \quad (\text{B.9})$$

Finally, by following the same way as in the proof of Theorem 4.1, we show that $\frac{\partial^2 C_i^{tot}}{\partial c_i^2} > 0$ always holds. This completes the proof that the problem is convex.

Appendix C

Cost Efficient Orchestration of Caching, Transcoding and Routing

C.1 Proof of Lemma 5.1

In order to ease the analysis and derive meaningful results, we assume the total segment amount m is sufficiently large ($m \gg 1$), and the Zipf parameter α can not be exactly equal to 1 (but can be arbitrarily closed to 1). Thus, we approximate the local cache hit ratio by using a continuous function as,

$$p_{hit}^v \approx \frac{\int_1^{x_v} t^{-\alpha} dt}{\int_1^m t^{-\alpha} dt} = \frac{x_v^{1-\alpha} - 1}{m^{1-\alpha} - 1}, \quad \alpha > 0, \quad \alpha \neq 1. \quad (\text{C.1})$$

Similarly, we approximate the local transcoding ratio as,

$$p_{tr}^v \approx \frac{x_v^{1-\alpha} - y_v^{1-\alpha}}{m^{1-\alpha} - 1} (1 - p_r), \quad \alpha > 0, \quad \alpha \neq 1. \quad (\text{C.2})$$

Based on those approximations, we then check the convexity of the objective function and each constraint.

First, by importing $z_v = x_v^{1-\alpha}$ as an auxiliary variable, we rewrite the objective function (5.9) into, $g_0(\mathbf{y}, \mathbf{z}) = \sum_{v \in V} p_{hit}^v = \sum_{v \in V} \frac{z_v - 1}{m^{1-\alpha} - 1}$, which is clearly linear to both z_v and y_v .

Second, we rewrite the constraint (5.10) into,

$$g_1(\mathbf{y}, \mathbf{z}) = \sum_{v \in V} (z_v^{\frac{1}{1-\alpha}} b_r + y_v (\sum_{i=1}^{r-1} b_i - b_r)) - o_{tot} \leq 0, \quad (\text{C.3})$$

where we have $\frac{\partial^2 g_1}{\partial y_v^2} = \frac{\partial^2 g_1}{\partial y_v \partial z_v} = \frac{\partial^2 g_1}{\partial z_v \partial y_v} = 0$, and $\frac{\partial^2 g_1}{\partial z_v^2} = \frac{\alpha b_r z_v^{\frac{-1-2\alpha}{1-\alpha}}}{(1-\alpha)^2}$, where $\alpha > 0$, $b_r > 0$, and $z_v^{\frac{-1-2\alpha}{1-\alpha}} > 0$, leading to $\frac{\partial^2 g_1}{\partial z_v^2} > 0$. Therefore, we obtain its Hessian matrix with respect to $\mathbf{y}$ and $\mathbf{z}$ as a diagonal matrix that, entries outside the main diagonal are all zero. Thus, the k -th determinant minor of $\mathbf{H}(\mathbf{g}_1)$ is,

$$|\mathbf{H}(\mathbf{g}_1)_k| = \begin{cases} \prod_{i=1}^k \frac{\partial^2 g_1}{\partial z_i^2}, & k \leq |V| \\ 0, & |V| < k \leq 2|V| \end{cases} \quad (\text{C.4})$$

This readily shows that, the Hessian matrix is positive semidefinite, that the determinant of all principal minors are greater or equal to 0. Thus, $g_1(\mathbf{y}, \mathbf{z})$ is jointly convex in $\mathbf{y}$ and $\mathbf{z}$.

Next, we examine constraint (5.11) by rewriting it into,

$$g_2(\mathbf{y}, \mathbf{z}) = \sum_{v \in V} \lambda_v \frac{(z_v - y_v^{1-\alpha})(1 - p_r)}{m^{1-\alpha} - 1} \sum_{j=1}^{r-1} p_j b_j \leq 0, \quad (\text{C.5})$$

where we still have $\frac{\partial^2 g_2}{\partial z_v^2} = \frac{\partial^2 g_2}{\partial y_v \partial z_v} = \frac{\partial^2 g_2}{\partial z_v \partial y_v} = 0$, and,

$$\frac{\partial^2 g_2}{\partial y_v^2} = \frac{\lambda_v \alpha (\alpha - 1) (1 - p_r)}{1 - m^{1-\alpha}} y_v^{-\alpha-1} \sum_{j=1}^{r-1} p_j b_j. \quad (\text{C.6})$$

When $0 < \alpha < 1$, there are $\alpha - 1 < 0$, and $1 - m^{1-\alpha} < 0$, resulting in $\frac{\partial^2 g_2}{\partial y_v^2} > 0$. When $\alpha > 1$, we have $\alpha - 1 > 0$, and $1 - m^{1-\alpha} > 0$, which still leads to $\frac{\partial^2 g_2}{\partial y_v^2} > 0$. Therefore, we prove $\frac{\partial^2 g_2}{\partial y_v^2} > 0$, for all $\alpha > 0$, $\alpha \neq 1$. Following this, we have its Hessian matrix $\mathbf{H}(\mathbf{g}_2)$ still as a positive semidefinite diagonal matrix. Thus, $g_2(\mathbf{y}, \mathbf{z})$ is jointly convex in $\mathbf{y}$ and $\mathbf{z}$.

Finally, since $p_{miss}^v = 1 - p_{hit}^v$ is linear function in $\mathbf{z}$, the constraint (5.12) is also linear over $\mathbf{y}$ and $\mathbf{z}$.

In summary, since all these equations are jointly convex in $\mathbf{y}$ and $\mathbf{z}$, we conclude that, the virtualized resource allocation problem is a convex optimization problem.

C.2 Proof of Theorem 5.1

In this subsection, we accomplish the proof of theorem 5.1, by examining the sufficient conditions for convergence of the Gauss-Seidel algorithm. In particular, from [86], if h and all constraints are 1) bounded from below; 2) differentiable; 3) marginally convex in $\mathbf{R}$, $\mathbf{x}$, and $\mathbf{z}$; and 4) jointly convex in $\mathbf{R}$, $\mathbf{y}$, and $\mathbf{z}$, then it will converge to the minimizer of h .

The first three conditions are satisfied through the definitions from our system models. Specifically, condition 1 is satisfied, because $\mathbf{x} \succeq 0$, $\mathbf{y} \succeq 0$, $\mathbf{z} \succeq 0$, and $\mathbf{R} \succeq 0$ by definition. Condition 2 is satisfied, since h and all constraints are differentiable. Condition 3 is satisfied, since p_{hit}^v , f and all constraints are convex in either $\mathbf{z}$ or $\mathbf{R}$ as shown in Lemma 5.1.

Next, we show the last condition is also satisfied. In particular, we introduce a new variable $w_{lv} = R_{lv}\lambda_v b_m p_{miss}^v$ to indicate the traffic flow size at link e that is heading towards node v . Thus, constraint (5.8) also changes into,

$$\sum_l w_{lv} = \lambda_v b_m p_{miss}^v, \quad (\text{C.7})$$

which is still linear jointly to $\mathbf{w}$ and $\mathbf{z}$. Following this, those original constraints are still jointly convex over $\mathbf{w}$ and $\mathbf{z}$, because constraint (5.10) and (5.11) are independent of $\mathbf{R}$ and $\mathbf{w}$, and constraint (5.12) is clearly linear to $\mathbf{w}$. Finally, we complete the proof on the convexity of the modified objective function,

$$h(\mathbf{w}, \mathbf{z}) = \sum_{e \in E} f\left(\sum_{v \in V} \sum_l w_{lv} I_{lv}^e / d_e\right) - \sum_{v \in V} \lambda_v p_{hit}^v, \quad (\text{C.8})$$

where $f(\sum_{v \in V} \sum_l w_{lv} I_{lv}^e / d_e)$ is strictly convex in $\mathbf{w}$ by definition, and p_{hit}^v is convex in $\mathbf{z}$ as shown in Lemma 5.1.

Abbreviations

AWS — Amazon Web Service.

CCMN — Cloud Centric Media Network.

CDN — Content Distribution Network.

EON — European Optical Network.

IPTV — Internet Protocol Television.

KKT — Karush Kuhn Tucker.

LFU — Least Frequently Used.

MDP — Markov decision process.

NFV — Network Function Virtualization.

NSF — National Science Foundation.

P2P — Peer to Peer.

PE — Performance Enhancement.

PSNR — Peak Signal-to-Noise Ratio.

QoS — Quality of Service.

RMSE — Root of Mean Squared Error.

SDN — Software Defined Networking.

UGC — User Generated Content.

VM — Virtual Machine.

WAN — Wide Area Network.

Bibliography

- [1] Cisco, “Cisco visual networking index: Forecast and methodology, 2013-2018.” White Paper, 2013.
- [2] 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*, pp. 25–34, ACM, 2009.
- [3] J. Xin, C.-W. Lin, and M.-T. Sun, “Digital video transcoding,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, 2005.
- [4] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, “Energy aware network operations,” in *INFOCOM Workshops 2009, IEEE*, pp. 1–6, IEEE, 2009.
- [5] AT&T *et al.*, “Network functions virtualisation -?introductory white paper.” http://portal.etsi.org/nfv/nfv_white_paper.pdf, 2012.
- [6] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A survey on software-defined networking,” *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 27–51, 2014.
- [7] L. Qiu, V. Padmanabhan, and G. Voelker, “On the placement of web server replicas,” in *IEEE INFOCOM*, vol. 3, pp. 1587–1596, 2001.
- [8] J. Kangasharju, J. Roberts, and K. Ross, “Object replication strategies in content distribution networks,” *Computer Communications*, vol. 25, no. 4, pp. 376–383, 2002.

- [9] M. Karlsson and C. Karamanolis, “Choosing replica placement heuristics for wide-area systems,” in *IEEE ICDCS*, pp. 350–359, 2004.
- [10] X. Tang and J. Xu, “Qos-aware replica placement for content distribution,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 10, pp. 921–932, 2005.
- [11] J. Dai, Z. Hu, and et al., “Collaborative hierarchical caching with dynamic request routing for massive content distribution,” in *IEEE INFOCOM 2012*, pp. 2444–2452, 2012.
- [12] T. Wauters, J. Coppens, F. Turck, B. Dhoedt, and P. Demeester, “Replica placement in ring based content delivery networks,” *Computer Communications*, vol. 29, pp. 3313–3326, June 2006.
- [13] I. Cidon, S. Kutten, and R. Soffer, “Optimal allocation of electronic content,” in *IEEE INFOCOM*, pp. 205–218, 2001.
- [14] O. Kariv and S. Hakimi, “An algorithm approach to network problems. i: the p-center,” *SIAM Journal of Applied Mathematics*, vol. 37, no. 3, pp. 513–538, 1979.
- [15] I. Baev, R. Rajaraman, and C. Swamy, “Approximation algorithms for data placement in arbitrary networks,” in *ACM-SIAM on Discrete Algorithms*, pp. 661–670, 2001.
- [16] J. Zhou, X. Zhang, L. Bhuyan, and B. Liu, “Clustered k-center: effective replica placement in peer-to-peer systems,” in *IEEE GLOBECOM*, pp. 2008–2013, 2007.
- [17] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, “Distributed placement of service facilities in large-scale networks,” in *IEEE INFOCOM*, pp. 2144–2152, 2007.
- [18] W. Zhu, C. Luo, J. Wang, and S. Li, “Multimedia cloud computing,” *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

- [19] D. Díaz-Sánchez, F. Almenarez, A. Marín, D. Proserpio, and P. Arias Cabarcos, “Media cloud: an open cloud computing middleware for content management,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 970–978, 2011.
- [20] F. Wang, J. Liu, and M. Chen, “CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities,” in *IEEE INFOCOM*, pp. 199–207, 2012.
- [21] X. Wang, T. T. Kwon, Y. Choi, H. Wang, and J. Liu, “Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users,” *IEEE Wireless Communications*, vol. 20, no. 3, pp. 72–79, 2013.
- [22] Y. Wen, X. Zhu, J. Rodrigues, and C. Chen, “Cloud mobile media: Reflections and outlook,” *IEEE Transactions on Multimedia*, vol. 16, no. 4, pp. 885–902, 2014.
- [23] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, “Simple-fying middlebox policy enforcement using SDN,” in *ACM SIGCOMM*, pp. 27–38, 2013.
- [24] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, “Multi-resource packing for cluster schedulers,” in *ACM SIGCOMM*, pp. 455–466, 2014.
- [25] L. Zhou and H. Wang, “Toward blind scheduling in mobile media cloud: Fairness, simplicity, and asymptotic optimality,” *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 735–746, 2013.
- [26] Y. Wu, C. Wu, B. Li, X. Qiu, and F. C. Lau, “Cloudmedia: When cloud on demand meets video on demand,” in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 268–277, 2011.
- [27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
- [28] H. E. Egilmez, B. Gorkemli, A. M. Tekalp, and S. Civanlar, “Scalable video streaming over openflow networks: An optimization framework for qos routing,” in *IEEE ICIP*, pp. 2241–2244, 2011.

- [29] S. Agarwal, M. Kodialam, and T. Lakshman, “Traffic engineering in software defined networks,” in *INFOCOM, 2013 Proceedings IEEE*, pp. 2211–2219, IEEE, 2013.
- [30] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford, “Live migration of an entire network and its hosts,” in *ACM HotNets*, pp. 109–114, 2012.
- [31] A. Arefin, R. Rivas, R. Tabassum, and K. Nahrstedt, “OpenSession: SDN-based cross-layer multi-stream management protocol for 3D teleimmersion,” in *IEEE ICNP*, 2013.
- [32] D. Li, Y. Shang, and C. Chen, “Software defined green data center network with exclusive routing,” in *IEEE INFOCOM*, 2014.
- [33] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, “Collaborative hierarchical caching with dynamic request routing for massive content distribution,” in *IEEE INFOCOM*, pp. 2444–2452, 2012.
- [34] Y. Li, H. Xie, Y. Wen, and Z.-L. Zhang, “Coordinating in-network caching in content-centric networks: Model and analysis,” in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 62–72, 2013.
- [35] Z. Wang *et al.*, “Joint online transcoding and geo-distributed delivery for dynamic adaptive streaming,” in *IEEE INFOCOM*, 2014.
- [36] T. Chang and Y. Li, “Deep shot: a framework for migrating tasks across devices using mobile phone cameras,” in *ACM CHI*, pp. 2163–2172, 2011.
- [37] Y. Lu, S. Li, and H. Shen, “Virtualized screen: A third element for cloud mobile convergence,” *IEEE Multimedia Magazine*, vol. 18, no. 2, pp. 4–11, 2011.
- [38] Y. Wu, Z. Zhang, C. Wu, Z. Li, and F. C. Lau, “Cloudmov: Cloud-based mobile social TV,” *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 821–832, 2013.
- [39] X. Wang, M. Chen, T. Kwon, L. Yang, and V. 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.

- [40] M. Bernaschi, F. Cacace, R. Clementelli, and L. Vollero, “Adaptive streaming on heterogeneous networks,” in *ACM workshop on Wireless multimedia networking and performance modeling*, pp. 16–23, 2005.
- [41] B. Shen, S.-J. Lee, and S. Basu, “Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks,” *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 375–386, 2004.
- [42] J. Chakareski and P. Frossard, “Adaptive systems for improved media streaming experience,” *IEEE Communications Magazine*, vol. 45, no. 1, pp. 77–83, 2007.
- [43] M. Armbrust, A. Fox, *et al.*, “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [44] J. Llorca, K. Guan, G. Atkinson, and D. Kilper, “Energy efficient delivery of immersive video centric services,” in *IEEE INFOCOM*, pp. 1656–1664, 2012.
- [45] I. Zhovnirofsky and M. Montpetit, “Internet discovery of enhancement and acceleration services (IDEAS),” *Internet-Drafts*, 2013.
- [46] F. Chen, K. Guo, J. Lin, and T. La Porta, “Intra-cloud lightning: Building cdns in the cloud,” in *IEEE INFOCOM*, pp. 433–441, 2012.
- [47] Y. Tian, R. Dey, Y. Liu, and K. W. Ross, “China’s internet: Topology mapping and geolocating,” in *IEEE INFOCOM*, pp. 2531–2535, 2012.
- [48] M. Crovella, M. Taqqu, and A. Bestavros, “Heavy-tailed probability distributions in the world wide web,” in *Statistical Techniques and Applications*, pp. 3–25, 1998.
- [49] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and zipf-like distributions: Evidence and implications,” in *IEEE INFOCOM*, pp. 126–134, 1999.
- [50] B. Krishnamurthy, J. Wang, and Y. Xie, “Early measurements of a cluster-based architecture for p2p systems,” in *ACM SIGCOMM Workshop on Internet Measurement*, pp. 105–109, 2001.

- [51] G. Pallis and A. Vakali, “Insight and perspectives for content delivery networks,” *Communications of the ACM*, vol. 49, no. 1, pp. 101–106, 2006.
- [52] “Topology lists.” <http://www.optical-network.com/topology.php>.
- [53] B. Bollobas and W. F. D. I. VEGA, “The diameter of random regular graphs,” *Combinatorica*, vol. 2, pp. 125–134, 1982.
- [54] K. Guan, G. Atkinson, D. Kilper, and E. Gulsen, “On the energy efficiency of content delivery architectures,” in *IEEE ICC workshops on Green Communications*, pp. 1–6, June 2011.
- [55] K. Guan, D. Kilper, and G. Atkinson, “Evaluating the energy benefit of dynamic optical bypass for content delivery,” in *INFOCOM workshops on Green Communications and Networking*, pp. 313–318, 2011.
- [56] D. Topiks and A. Veinott, “On the convergence of some feasible direction algorithms for nonlinear programming,” *J. SIAM Control*, vol. 5, pp. 280–294, 1967.
- [57] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.
- [58] “GoGrid CDN pricing.” <http://www.gogrid.com/cloud-hosting/content-delivery-network.php>.
- [59] “Windows azure media services pricing details.” [online], <http://www.windowsazure.com/en-us/pricing/details/media-services/>.
- [60] “Rackspace pricing.” <http://www.rackspace.com/cloud/servers/pricing/>.
- [61] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, “A server’s perspective of Internet streaming delivery to mobile devices,” in *IEEE INFOCOM*, pp. 1332–1340, 2012.
- [62] Cisco, “Cisco digital media encoder 2000.” [online], <http://www.cisco.com/c/en/us/products/video/digital-media-encoder-2000/index.html>, 2014.

- [63] Microsoft, “An introduction to live encoding with azure media services.” [online], <https://azure.microsoft.com/en-us/blog/an-introduction-to-live-encoding-with-azure-media-services/>, 2015.
- [64] T. Stockhammer, “Dynamic adaptive streaming over http: standards and design principles,” in *ACM MMSys*, pp. 133–144, 2011.
- [65] L. Xu, S. Kwong, H. Wang, Y. Zhang, D. Zhao, and W. Gao, “A universal rate control scheme for video transcoding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 4, pp. 489–501, 2012.
- [66] T. Shanableh, E. Peixoto, and E. Izquierdo, “MPEG-2 to HEVC video transcoding with content-based modeling,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 7, pp. 1191–1196, 2013.
- [67] L. Guo, S. Chen, and X. Zhang, “Design and evaluation of a scalable and reliable P2P assisted proxy for on-demand streaming media delivery,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 5, pp. 669–682, 2006.
- [68] W.-P. Yiu, X. Jin, and S.-H. Chan, “VMesh: Distributed segment storage for peer-to-peer interactive video streaming,” *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1717–1731, 2007.
- [69] S. Guo, H. Xie, and G. Shi, “Collaborative forwarding and caching in content centric networks,” in *IFIP Networking*, pp. 41–55, 2012.
- [70] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and Zipf-like distributions: Evidence and implications,” in *IEEE INFOCOM*, vol. 1, pp. 126–134, 1999.
- [71] Y. Jin and Y. Wen, “PAINT: Partial in-network transcoding for adaptive streaming in information centric network,” in *IEEE/ACM IWQOS*, pp. 208–217, 2014.
- [72] K. Li, H. Shen, F. Y. Chin, and W. Zhang, “Multimedia object placement for transparent data replication,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 2, pp. 212–224, 2007.

- [73] M. Bjorkqvist, L. Y. Chen, M. Vukolic, and X. Zhang, “Minimizing retrieval latency for content cloud,” in *IEEE INFOCOM*, pp. 1080–1088, 2011.
- [74] R. Grandl, K. Su, and C. Westphal, “On the interaction of adaptive video streaming with content-centric networking,” *arXiv preprint arXiv:1307.0794*, 2013.
- [75] C. Peng, M. Kim, Z. Zhang, and H. Lei, “VDN: Virtual machine image distribution network for cloud data centers,” in *IEEE INFOCOM*, pp. 181–189, 2012.
- [76] S. Lederer *et al.*, “Distributed dash dataset,” in *ACM Multimedia Systems Conference*, pp. 131–135, 2013.
- [77] S. K. Fayazbakhsh, Y. Lin, *et al.*, “Less pain, most of the gain: Incrementally deployable ICN,” in *ACM SIGCOMM*, pp. 147–158, 2013.
- [78] R. Muñoz, R. Vilalta, R. Casellas, R. Martinez, T. Szyrkowiec, A. Autenrieth, V. López, and D. López, “Integrated sdn/nfv management and orchestration architecture for dynamic deployment of virtual sdn control instances for virtual tenant networks,” *Journal of Optical Communications and Networking*, vol. 7, no. 11, pp. B62–B70, 2015.
- [79] Z. Li *et al.*, “On the geographic patterns of a large-scale mobile video-on-demand system,” in *IEEE INFOCOM*, 2014.
- [80] AmazonEC2, “Amazon elastic transcoder pricing.” [online], <http://aws.amazon.com/elastictranscoder/pricing/>, 2014.
- [81] A. Chanda, C. Westphal, and D. Raychaudhuri, “Content based traffic engineering in software defined information centric networks,” in *IEEE Infocom NOMEN workshop*, 2013.
- [82] S. Jain *et al.*, “B4: Experience with a globally-deployed software defined WAN,” in *ACM SIGCOMM*, pp. 3–14, 2013.
- [83] M. Yu, L. Jose, and R. Miao, “Software defined traffic measurement with opensketch,” in *NSDI*, vol. 13, pp. 29–42, 2013.

- [84] B. Fortz, J. Rexford, and M. Thorup, “Traffic engineering with traditional IP routing protocols,” *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002.
- [85] J. Rexford, “Route optimization in IP networks,” in *Handbook of Optimization in Telecommunications*, pp. 679–700, 2006.
- [86] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Athena Scientific, 1997.
- [87] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.
- [88] “Internet2 layer 3 topology.” <http://www.internet2.edu/media/medialibrary/2013/10/01/I2-Network-Infrastructure-Layer-3.pdf>.
- [89] M. Montpetit and M. Médard, “Social television: Enabling technologies and architectures,” *Proceedings of the IEEE*, pp. 1395–1399, 2012.
- [90] Z. Yu, X. Zhou, L. Zhou, and K. Du, “A hybrid similarity measure of contents for TV personalization,” *Multimedia Systems*, vol. 16, pp. 231–241, 2010.
- [91] Y. Jin, X. Liu, Y. Wen, and J. Cai, “Inter-screen interaction for session recognition and transfer based on cloud centric media network,” in *IEEE ISCAS*, pp. 877–880, 2013.
- [92] Y. Jin, T. Xie, Y. Wen, and H. Xie, “Multi-screen cloud social TV: transforming TV experience into 21st century,” in *Proceedings of ACM international conference on Multimedia*, pp. 435–436, 2013.
- [93] Y. Zhang, C. Yan, F. Dai, and Y. Ma, “Efficient parallel framework for h. 264/avc deblocking filter on many-core platform,” *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 510–524, 2012.
- [94] “Amazon cloudfront.” <http://aws.amazon.com/cloudfront/>.
- [95] “Metacdn.” <http://www.metacdn.com/>.

- [96] T. Mei, J. Guo, X.-S. Hua, and F. Liu, “AdOn: toward contextual overlay in-video advertising,” *Multimedia Systems*, vol. 16, pp. 335–344, 2010.
- [97] Ş. Gündüz and M. T. Özsu, “A web page prediction model based on click-stream tree representation of user behavior,” in *ACM SIGKDD*, pp. 535–540, 2003.
- [98] V. S. Tseng and K. W. Lin, “Efficient mining and prediction of user behavior patterns in mobile web systems,” *Information and Software Technology*, vol. 48, no. 6, pp. 357–369, 2006.
- [99] V. Gopalakrishnan, R. Jana, *et al.*, “Understanding couch potatoes: measurement and modeling of interactive usage of IPTV at large scale,” in *ACM IMC*, pp. 225–242, 2011.
- [100] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [101] G. Tesauro, “Temporal difference learning and td-gammon,” *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.
- [102] “Amazon pricing.” <http://aws.amazon.com/pricing/>.
- [103] Cisco, “User guide for cisco media experience engine 3000.” http://www.cisco.com/en/US/docs/video/mxe/2_x/2.0/user/guide/mxe_20_ug.pdf.
- [104] Y. Wu, Z. Zhang, C. Wu, Z. Li, and F. Lau, “Cloudmov: Cloud-based mobile social tv,” *IEEE Transactions on Multimedia*, pp. 1–12, 2013.
- [105] “Openstack.” <https://www.openstack.org/>, 2014.
- [106] “Open vswitch.” <http://openvswitch.org/>, 2014.
- [107] “Opendaylight.” <http://www.opendaylight.org/>, 2014.

Publications

Journals

- **Y. Jin**, Y. Wen, and C. Westphal, “Optimal Transcoding and Caching for Adaptive Streaming in Media Cloud: An Analytical Approach,” *accepted to IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, Feb, 2015.
(Chapter 4 in this thesis is based upon this paper)
- **Y. Jin**, Y. Wen, H. Hu, and M-J. Montpetit, “Reducing Operational Costs in Cloud Social TV: an Opportunity for Cloud Cloning,” *IEEE Transactions on Multimedia (TMM)*, vol. 16, no. 6, pp. 1739-1751, Oct, 2014.
(Chapter 6 in this thesis is based upon this paper)
- **Y. Jin**, Y. Wen, and W. Zhang, “Content Routing and Lookup Schemes Using Global Bloom Filter for Content Delivery as a Service,” *IEEE Systems Journal*, vol. 8, no. 1, pp. 268-278, Mar, 2014.
- **Y. Jin**, Y. Wen, Q. Chen, and Z. Zhu, “An Empirical Investigation of the Impact of Server Virtualization on Energy Efficiency for Green Data Center,” *The Computer Journal*, vol. 51, no. 8, pp. 968-975, Aug, 2013.
- Y. Dong, L. Zhou, **Y. Jin** and Y. Wen, “Improving Energy Efficiency for Mobile Media Cloud via Virtual Machine Consolidation,” *ACM Springer Mobile Networks and Applications (MONET)*, vol. 20, no. 3, pp. 370-379, Jun, 2015.

Conferences

- **Y. Jin**, Y. Wen, and C. Westphal, “Towards Joint Resource Allocation and Routing to Optimize Video Distribution over Future Internet,” *IFIP/IEEE Networking*, pp.

150-158, May, 2015.

(Chapter 5 in this thesis is based upon this paper)

- H. Hu, **Y. Jin**, Y. Wen, T-S. Chua, and X. Li, “Toward a Biometric-Aware Cloud Service Engine for Multi-Screen Video Applications,” *ACM SIGCOMM*, pp. 581-582, Aug, 2014.
- **Y. Jin**, H. Hu, X. Liu, and Y. Wen, “MUTAS: Multi-screen TV Experience as a Service through Cloud Centric Media Network,” *IEEE International Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON)*, pp. 146-148, Jul, 2014.
- **Y. Jin**, and Y. Wen, “PAINT: Partial In-Network Transcoding for Adaptive Streaming in Information Centric Network,” *IEEE/ACM International Symposium on Quality of Service (IEEE/ACM IWQoS)*, pp. 208-217, May, 2014.
(Chapter 4 in this thesis is based upon this paper)
- **Y. Jin**, Y. Wen, and H. Hu, “Minimizing Monetary Cost via Cloud Clone Migration in Multi-screen Cloud Social TV System,” *IEEE Global Communications Conference (IEEE Globecom)*, pp. 1747-1752, Dec, 2013. (**Best Paper Award**)
(Chapter 6 in this thesis is based upon this paper)
- **Y. Jin**, T. Xie, Y. Wen, and H. Xie, “Multi-Screen Cloud Social TV: Transforming TV Experience into 21st Century,” *ACM international conference on Multimedia (ACM MM)*, pp. 435-436, Oct, 2013. (**Best Demo Finalist**)
- **Y. Jin**, Y. Wen, K. Guan, D. Kilper, and H. Xie, “Towards Monetary Cost Effective Content Placement in Cloud Centric Media Network,” *IEEE International Conference on Multimedia and Expo (IEEE ICME)*, pp. 1-6, Jul, 2013.
(Chapter 3 in this thesis is based upon this paper)
- **Y. Jin**, X. Liu, Y. Wen, and J. Cai, “Inter-Screen Interaction for Session Recognition and Transfer Based on Cloud Centric Media Network,” *IEEE International Symposium on Circuits and Systems (IEEE ISCAS)*, pp. 877-880, May, 2013.

- **Y. Jin**, and Y. Wen, “Content Routing and Lookup Schemes using Global Bloom Filter for Content-Delivery-as-a-Service,” *IEEE Global Communications Conference (IEEE Globecom)*, pp. 2131-2136, Dec, 2012.
- **Y. Jin**, Y. Wen, and Q. Chen, “Energy Efficiency and Server Virtualization in Data Centers: An Empirical Investigation,” *IEEE INFOCOM Workshop on Communications and Control for Sustainable Energy Systems*, pp. 133-138, Mar, 2012.
- **Y. Jin**, Y. Wen, G. Shi, G. Wang, and A.V. Vasilakos, “CoDaaS: An Experimental Cloud-Centric Content Delivery Platform for User-Generated Contents,” *IEEE International Conference on Computing, Networking and Communications (IEEE ICNC)*, pp. 934-938, Jan, 2012.