Delivery and transcoding for large scale live streaming systems

Karine Pires

UNIVERSITÉ PIERRE ET MARIE CURIE
TELECOM BRETAGNE

École doctorale Informatique, Télécommunications et Électronique (Paris)

T H È S E

pour obtenir le grade de

Docteur en Sciences

de l'Université Pierre et Marie Curie et de Télécom Bretagne

Mention : INFORMATIQUE

Présentée par

Karine PIRES

Diffusion et Transcodage à Grande Échelle de Flux Vidéo en Direct

Thèse dirigée par Pierre SENS
et encadrée par Gwendal SIMON et Sébastien MONNET
préparée à l'UPMC - LIP6, équipe REGAL et Telecom Bretagne
Soutenue le 31 mars 2015

Devant le jury composé de :

Rapporteurs :	Eddy CARON	-	ENS Lyon - LIP
	Toufik AHMED	-	Université Bordeaux INP - LaBRI
Examinateurs :	Cyril CONCOLATO	-	Télécom ParisTech
	Maria POTOP-BUTUCARU	-	UPMC - LIP6 - NPA
	Shadi IBRAHIM	-	INRIA Rennes - KerData
Directeur :	Pierre SENS	-	UPMC - LIP6 - REGAL
Encadrants :	Gwendal SIMON	-	Télécom Bretagne
	Sébastien MONNET	-	UPMC - LIP6 - REGAL

Acknowledgments

These three years that I have worked to achieve my doctorate would not have been done without the help, support and advice of many people. I am glad to have the opportunity to express my gratitude to them.

First of all, I would like to say my sincere thanks to my thesis director, Pierre Sens, my advisor Sébastien Monnet from LIP6-UPMC, and my advisor Gwendal Simon from Telecom Bretagne. With their precious guidance I was able to complete a rich study, which I am very proud of.

I also want to thank the committee that evaluated this thesis. I offer my sincere gratitude to Eddy Caron and Toufik Ahmed who spent their valuable time in order to help me with their careful insights. Thanks to their efforts, the quality of this document was significantly improved.

My sincere gratitude to my master's degree advisor Elias Procópio Duarte Júnior, who introduced me to the academic community and recommended me to this thesis.

In both institutions I was associated, I met amazing colleagues and friends. I would like to thank all the PhD students from the Regal Team on LIP6 and the RSM Department on Telecom Bretagne. I also had the support of wonderful people that help me with several administrative tasks. I would like to thank in LIP6 Marguerite Sos, Anne Quezel, and Eugène Kamdem. In Telecom Bretagne Armelle Lannuzel, Marie-Pierre Yvenat and Anais Renaud. My thanks to my friends Luciana Arantes, who is an amazing scientist and helped me in so many ways during this thesis, and Gabriela Schroeder for so many good moments in these three years.

During my thesis I have the great pleasure to collaborate in works with various remarkable people. Thanks to Guthemberg Silvestre, who is not only a dear friend, who helped me when I first arrived in France, but also a great scientist. Thanks to the excellent scientists Ramon Aparecido Pardo, Alberto Blanc, Patrick Maillé, Bruno Tuffin, Laura Toni and Pascal Frossard for all the hard work and fruitful discussions.

My heartily thanks to my lovely family. My parents Maria Aparecida de Oliveira Pires and Maurício Pires, and my brothers Rogers de Oliveira Pires and Alexander Iwanko Pires, who always supported me and have being present in this journey even being geographically far. My boyfriend Maxime Véron who has brought to my life so much love and happiness. My family in law, Evelyne Véron, Patrick Véron and Claire-Line Véron, who accepted me in their warm hearts.

Résumé

Aujourd'hui, de nombreux appareils sont capables de capturer des vidéos en Full HD et d'utiliser une connexion réseau pour accéder à Internet. La popularisation des dispositifs et les efforts continus pour améliorer la qualité du réseau ont apporté un environnement propice à l'essor de la diffusion en direct. De part la grande quantité de contenu généré par les utilisateurs, la diffusion de flux en direct présente de nouveaux défis. Dans cette thèse, nous intêresons à la fois à distribution et du transcodage des systèmes de diffusion en direct.

Nous avons commencé par créer un ensemble de données de sessions de streaming en direct. Pour étudier les aspects que nous ciblons des systèmes de diffusion en direct nous avons besoin de les caractériser et d'évaluer les solutions proposées avec des traces pertinentes. Par conséquent, notre première contribution est un ensemble de données et son analyse, contenant trois mois traces de deux services de streaming en direct généré par les utilisateurs. Avec des millions de sessions en direct et des centaines de milliers d'utilisateurs nous avons rendu ces données librement disponibles pour la communauté.

Ensuite, nous avons exploré et développé une solution pour la distribution du contenu massif produit par ces platesformes. L'un des défis est l'immense variation du nombre total de téléspectateurs et la grande hétérogénéité des flux populaires. Ceci implique généralement un surdimensioment des services et par conséquent un important gaspillage de ressources. Nous proposons trois solutions: (i) l'une basée sur une prévision de popularité afin de placer les flux sur des plateformes telles que le *nuage* (cloud) ou des machines virtuelles distribuées; (ii) l'autre utilisant une distribution hybride entre les serveurs propriétaires et les reseaux de distribution de contenu (CDN pour Content Delivery Network); (iii) nous discutons des aspects economics sur la diffusion a basé de CDN.

Enfin, nous passé puis les difficultés posées par le transcodage des flux en direct. Les opérations de transcodage sont coûteuses en ressources CPU et sont des étapes clés pour le Streaming à Débit Adaptatif (SDA). Nous présent que le SDA est capable de réduire le coût en bande passante pour la distribution et d'augmenter la qualité d'expérience des téléspectateurs en échange d'un coût en ressources CPU pour transcodage. Pour comprendre le compromis entre les avantages et les coûts, nous formulons deux problèmes de gestion. Le premier est une version simplifiée dans laquelle nous concevons deux stratégies pour décider quels flux devraient être livrés par SDA. Pour la deuxième formulation, nous présentons une programmation linéaire en nombres entiers pour maximiser la qualité moyenne de l'expérience de l'utilisateur et un algorithme heuristique capable de passer à échelle d'un grand nombre de vidéos et utilisateurs.

Abstract

Today many devices are capable to capture full HD videos and use its network connection to access Internet. The popularization of devices and continuous efforts to increase network quality has brought a proper environment for the rise of live streaming. Associated to the large scale of Users Generated Content (UGC), live streaming presents new challenges. In this thesis we target the *delivery* and *transcoding* of live streaming systems.

First, we created a live streaming sessions data set. To study the aspects we target of live streaming systems we need to characterize them and evaluate the proposed solutions with relevant input traces. Therefore our first contribution is a data set, and its analysis, containing three months traces of two UGC live streaming services. With millions of live sessions and hundreds of thousands broadcasters we made it freely available for the community.

Second, we explored and developed solutions for the delivery of the massive content produced by these platforms. One of the challenges is the huge variation in the total number of viewers and the great heterogeneity among streams popularity, which generally implies over-provisioning and consequently an important resource waste. In this thesis, we show that there is a trade-off between the number of servers involved to broadcast the streams and the bandwidth usage among the servers. We also stress the importance to predict streams popularity in order to efficiently place them on the servers. We explore three solutions, one based on platforms such as clouds or distributed virtual machines and uses popularity predictions to map live-streams on the servers, another based on assisted delivered involving proprietary servers and Content Delivery Network (CDN), and finally we discuss the economics aspects related to CDN based delivery.

Lastly, we target the difficulties concerning transcoding of live streams. The transcoding operations over streams are computing consuming and are key operations on adaptive bit rate streaming. We show that adaptive streaming is able to reduce the delivery bandwidth cost and to increase viewer quality of experience at the cost of computing resources for transcoding purposes. To address the trade-off between benefits and costs, we formulate two management problems. The first is a simplified version in which we design two strategies for deciding which online channels should be delivered by adaptive bit rate streaming. The second formulation we present an integer linear program to maximize the average user quality of experience and a heuristic algorithm that can scale to large number of videos and users.

Contents

List of Figures

Chapter 1

Introduction

Contents

We are currently witnessing the emergence of two phenomena: the popularization of video capture devices and the explosion of network quality with a continually increasing number of Internet users. Indeed, nowadays, any laptop, netbook or even cell phone has a camera. Full HD devices have become affordable, including their mobile version. It implies that a large portion of the population has the necessary equipment to create streaming videos.

Regarding networks, we have seen in recent years an influx of users and a globally better coverage. In parallel, network quality in terms of latency and throughput improved significantly, and this improvement continues, in particular with the arrival of very high speed networks like the optical fiber. In addition, connection charges have become reasonable, and the vast majority of Internet users has unlimited access. The mobile network coverage is extensive and of good quality. The progression rate of the coverage of mobile broadband (3G/4G) suggests that the population could have a permanent access to a network of good quality at a reasonable price.

The increase of network quality allowed users to consume more video content through the Internet. Netflix, a Video on Demand (VoD) streaming service, surpasses the 50 million subscribers mark on the second quarter of 2014 with a $1.34 billion revenue [Sha14]. Twitch, a live video streaming service, becomes the fourth largest source of US peak Internet traffic in February, 2014 and was acquired on the same year by Amazon.com for near a billion dollars [FW14]. Already by 2013 video content represented more than half of global Internet traffic [Inc14].

Combined, these phenomena, and the tendency of users to expose portions of their live led to a popularity rise of live video streaming systems. Many

different actors are involved on User-Generated Content (UGC) live video streaming services. Figure 1.1 illustrates a simplification of this rich environment. Live video streaming consists of users that both consume and produce the video content, service providers responsible for the platform, and different processes for transcoding and delivering the live content with time constraints on large scale.

In this thesis we study many challenges involving these systems. We discuss the challenges raised by these systems in the next section. Further, we present our contributions on this domain. Lastly, an organization of the remaining of this thesis is given.

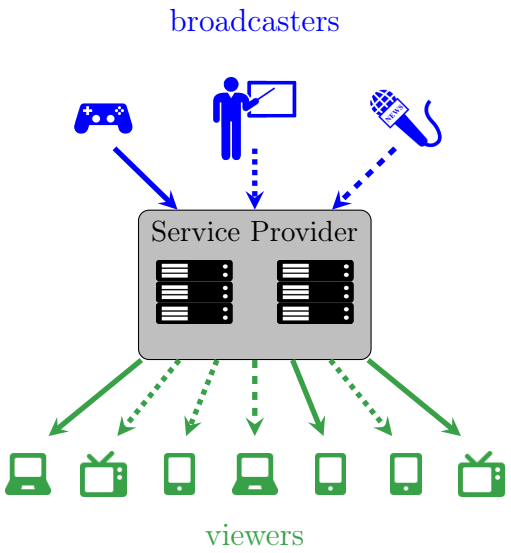


Figure 1.1: Simplified live streaming architecture

1.1 Motivation

The core functionalities of different UGC live streaming platforms, for instance Twitch and the live branch of YouTube, share a lot of similarities. Broadcasters produce video content, upload it to the platform and the platform is then in charge to deliver it to viewers. However, the infrastructure and solutions designed by the services are different. Google uses multiple data centers distributed over the globe to deliver its services, including YouTube [AJCZ12]. A composition of Peer-to-Peer (P2P) assisted by Content Delivery Network (CDN) to improve viewers Quality of Experience (QoE) was deployed on LiveSky [YLZ+09a]. Twitch uses self own data center assisted by CDN [Hof12].

We target in this work two main live streaming components that are critical for the services infrastructure: the delivery and the transcoding.

The challenge of delivering multimedia content on a large scale is essentially a problem related to the reservation of physical resources, specially outgoing bandwidth (data flow from servers to viewers). To address this problem, the scientific community has designed P2P algorithms to improve the delivery performances [Pas12]. However, various constraints have limited the deployment of P2P systems for commercial purpose, for example firewalls and Network Address Translator (NAT) still prevent many direct connection between users [JDMP11]. Common industry models use CDNs for delivery of video content. Cisco claims that CDNs will carry over half of video Internet traffic by 2018 [Inc14]. Recent successes of major live event delivery [Fie12] demonstrate the reliability of CDN. Additionally studies have dealt with other concerns, like economical costs [ASV11] of the delivery.

Large scale transcoding also requires an important reservation of processing power resources. Adaptive Bit Rate (ABR) was conceived to support the growing video consumption over heterogenous devices and network conditions. To prepare the different video representations, required by ABR, the raw video must be processed by transcoding operations. Cloud-based transcoding has been the subject of several studies. Most of these works [GB06, JAL+13, LZG12, LZY+13, HMLW11] take advantage of modern video compression techniques by splitting the video content into parts that can be processed individually. These techniques require video pre-fetching for the processing and are hard to apply to on-the-fly transcoding required by live streaming. Few works target real-time transcoding, for example a cloud-based video transcoding framework was designed in the context of mobile video conferencing [CWLC14] and the feasibility of low latency approach with ABR was validated [BCF14]. However, no further application on large scale, as the one found on UGC video streaming, were proposed.

1.2 Live Streaming Services Challenges

Although the importance of live video streaming systems and continuously growing in terms of Internet traffic there are still remaining open challenges. We consider the following important aspects that must be target for the improvement of live streaming systems.

Unreliable and Heterogeneous Content. The broadcasters of UGC live streaming systems can be less reliable since at first there is no official content requirements. The introduction of programs like partnership, proposed by Twitch, requires that the broadcaster maintains in its channel a constant production of sessions (at least three times a week). This content reliability is raised by certain problems associated to UGC live video streaming. First, a

channel can switch from offline to online and vice versa at any time. Second, the content produced by broadcasters are highly heterogeneous, the emitted video streams have various bit rates and resolutions, as well as various encoding parameters. Third, the broadcasters do not give much information about their video streams.

Massive Scale Systems. Our focus is on the thousands of broadcasters who use live streaming services such as Twitch, YouTube Live, Ustream,[1] Livestream,[2] and Dailymotion[3] to broadcast live an event that they are capturing from their connected video device (*e.g.*, camera, smartphone, and game console). Many challenges are still open when regarding to live video streaming services. Although its relevance on nowadays, solutions for coping with the massive data produced by UGC live video streaming services are not trivial. Typically, Twitch announced in 2013 a significant increase in the delay [Sta13], while the live service from YouTube, for more than 2 years, was only offered to a subset of users [You13].

Data Set Inexistent. The first obstacle on when proposing new approaches and solutions for live streaming services is the need to validate the proposal with real traces from service providers. Although the multimedia community have made efforts to make real data sets available, for example the collection of raw video content [Xip14] and the Dynamic Adaptive Streaming over HTTP (DASH) sessions [BSMM14], unfortunately no public data set comprising live streaming sessions were existent when we started this work.

CDN Impact is Neglected. The major live video content providers, like Twitch and YouTube Live have distinct organizations of their delivery structure. Despite the existence of multiple options for delivering the live video content, no formal definition considering their differences have been made. CDNs are notable actors over diverse delivery compositions. Whereas their importance on the delivery chain, no evaluation considering the profit-driven policies of CDNs were investigated, neither their impact on viewers quality of experience.

Multiple Video Qualities. As opposed to the traditional TV providers and the content owners from the entertainment industry, the broadcasters of UGC systems usually do *not* emit ultra-HD video streams (2160p also known as *4k*), however they transmit diverse other video qualities. Besides source videos having different qualities, viewers consume the video content with distinct devices and network conditions. The implementation of ABR is a solution to improve the delivery for heterogeneous clients conditions. On live streaming

[1] http://www.ustream.tv/
[2] http://new.livestream.com/
[3] https://www.dmcloud.net/features/live-streaming

services, it can diminish the needs of network capacity for the sessions delivery. Viewers that have no capacity or smaller resolutions devices could with ABR select lower video bit rates, leading to less bandwidth usage on the server side. However, real-time constraints of the live scenario and high CPU costs of transcoding operations needed to create distinct video representations make the ABR implementation challenging. For instance, Twitch ABR streaming is only offered to some premium broadcasters. That is, only a small subset of channels is transcoded into multiple representations. Smarter selection of channels to transcode and analyze of the trade-off between benefits and costs when implementing ABR were not studied.

1.3 Summary of Contributions

This thesis presents contributions related to live video streaming services. We organized them into three themes: Data Set, Delivery and Transcoding.

Live Streaming Video Services Data Set. We first create a data set with real traces extracted from two major live video stream systems, namely Twitch and YouTube Live. This is our first contribution: an analysis over the created data set and the availability of the data set and scripts for the community [PS15]. Over 10 millions live sessions and more than a million broadcasters were registered over three months of collected data, from January 6, 2014 to April 6, 2014 and are publicly available.[4]

Live Streaming Video Content Delivery. The delivery of live streaming video content is done in various ways. We analyze benefits and weaknesses of different industry delivery compositions and literature studies. The first delivery composition we explore is based on cloud platforms. The heterogeneity among channel popularity generally implies over-provisioning, leading to an important resource waste. We show that there is a trade-off between the number of servers involved in the delivery of the sessions and the bandwidth usage among the servers. We present an approach for delivering live stream videos tailored for the cloud, which profits from the knowledge of channels popularity [PMS14a, PMS14b]. We then propose a hybrid composition based on CDNs and data center. We define a formalization for hybrid delivery problem and perform simulations using our real data set traces. In our results we compare strategies to the optimal solution. Lastly we work on a CDN delivery

[4] http://dash.ipv6.enstb.fr/dataset/twitch-youtube/

and its economical consequences to net neutrality [MPST14]. Using our data set, we notice that a CDN remains a relatively neutral actor even when one of the content providers it serves tries to monopolize the CDN storage space by implementing an aggressive policy to harm its competitors.

Transcoding of Live Streaming Video. The adoption of ABR on live streaming services can increase viewers QoE and reduce delivery bandwidth costs. However, this adoption adds to the platform new transcoding costs required by the preparation of multiple representations of the video session that enables the use of ABR. We contribute with an analysis of the trade-off between benefits and costs when implementing ABR for live streaming systems [PS14]. We formulate a management problem and we design two strategies for deciding which online channels should be delivered by ABR. Our evaluations, still based on our real traces, show that these strategies can reduce the overall infrastructure cost by 40% in comparison to an implementation without adaptive streaming. Next we provide a fine-grained analysis on the management problem with two other data sets and a new heuristic strategy [APSB15]. Based on the optimal results obtained from the problem optimization we offer a new heuristic algorithm. We compare our heuristic with current industry standards, showing that the latter are sub-optimal. Our heuristic can satisfy a time varying demand by efficiently exploiting an almost constant amount of computing resources.

1.4 Thesis Organization

This manuscript is organized into three distinct parts. First we present in detail the characteristics of live streaming services. In Chapter 2 we describe live streaming platforms and related work. We introduce in Chapter 3 the first contribution of this thesis, the live streaming data set and its analysis. This data set is composed by two major live streaming services traces, Twitch and YouTube Live, with information about their broadcasters and viewers. Our analysis points out important key elements of the live streaming systems that we further explore on the following contributions.

The second part describes our contributions for the delivery of live streaming video content. Chapter 4 presents our approach of delivery using popularity aware solution based on cloud platforms. Chapter 5 explores hybrid compositions of delivery, such as the one adopted by Twitch.

The third part exposes our contributions on the transcoding of live streaming videos. In Chapter 6 we investigate the implementation impact of adaptive

bit rate streaming in live streaming platforms.

We conclude and present future works of this thesis in Chapter 7.

Additionally, we describe other related contributions in Appendix C, which we applied our data set introduced in Chapter 3. Section C.2 assumes entirely delivery by CDN and analyses the economic impact of this delivery model. We use our data set to make two services providers compete for the CDN delivery resources. We study in Section C.3 the problem of preparing live video streams for delivery using cloud computing infrastructure. In this work we evaluate a new heuristic strategy with our data set.

1.5 List of Publications

International Journals

[SBP+15] Guthemberg Silvestre, David Buffoni, Karine Pires, Sébastien Monnet, and Pierre Sens. Boosting streaming video delivery with wisereplica. *Transactions on Large-Scale Data and Knowledge-Centered Systems (TLDKS)*, 2015.

[TAP+14] Laura Toni, Ramon Aparicio-Pardo, Karine Pires, Gwendal Simon, Alberto Blanc, and Pascal Frossard. Optimal selection of adaptive streaming representations. *Transactions on Multimedia Computing, Communications and Applications (TOMM)*, 2014.

International Conferences

[APSB15] Ramon Aparicio-Pardo, Karine Pires, Gwendal Simon, and Alberto Blanc. Transcoding live adaptive video streams at a massive scale in the cloud. In *ACM MMSys*, 2015.

[MPST14] Patrick Maillé, Karine Pires, Gwendal Simon, and Bruno Tuffin. How neutral is a cdn? an economic approach. In *CNSM*. IEEE, 2014.

[PMS14b] Karine Pires, Sébastien Monnet, and Pierre Sens. Pops: a popularity-aware live streaming service. In *ICPADS*. IEEE, 2014.

[PS14] Karine Pires and Gwendal Simon. Dash in twitch: Adaptive bitrate streaming in live game streaming platforms. In *VideoNext CoNEXT Workshop*. ACM, 2014.

[PS15] Karine Pires and Gwendal Simon. Youtube live and twitch: A tour of user-generated live streaming systems. In *MMSys Data Sets*. ACM, 2015.

French Conference

[PMS14a] Karine Pires, Sébastien Monnet, and Pierre Sens. Pops : service de diffusion de flux vidéos live prenant en compte la popularité. In *ComPAS*, 2014.

Book Chapter

[PSed] Karine Pires and Gwendal Simon. Interactive multimedia applications on clouds: Video services, games, others. In *Cloud Services, Networking and Management*, to be published.

Chapter 2

State of the Art of Live Streaming Systems

Contents

2.1 Introduction

Live video streaming has got the attention of both the academic and the industrial community. This resulted on a high number of works. As we previously discussed, the consumption of video streaming increase stress the previous solutions and current infrastructures. Different restrictions were applied by major live streaming services to cope with the scale problem. For example, Twitch announced in 2013 a significant increase in the delay [Sta13], while the live service from YouTube, for more than 2 years, was only offered to a subset of users [You13].

There is no consensus, both in the academia and the industry, on an ideal infrastructure for these systems. In this chapter we present the state-of-the-art associated to live streaming video services.

In the first section we explore related topics as Internet video consumption and video streaming. Since our study is mainly based on traces of UGC

services we introduce a discussion about the various studies related to UGC services.

The following section relates to the delivery part of our work. It introduces the multiple delivery models designed for the diffusion of video streaming. We discuss models that focus on a unique technology, followed by hybrid delivery models, which associate multiple technologies.

The last section presents works related to the transcoding of video content. As previously mentioned, video transcoding is a key piece of the process for delivering video streaming with ABR. Finally, we also discuss works related to ABR.

2.2 Background

Many different actors are involved on UGC live video streaming services. This rich environment consists of users that both consume and produce the video content, service providers responsible for the platform, and different processes for transcoding and delivering the live content with time constraints on large scale. In the following we describe important aspects of live streaming systems.

User-Generated Content (UGC). We call UGC any form of content such as blogs, tweets, images, videos, and other forms of media that are created by users of an online system or service. This content is often made available via social media websites. UGC is responsible for the emergence of concepts such as citizen journalism, where public citizens collect and report news and information. UGC is the central subject of popular Internet services such as YouTube[1] and Wikipedia[2]. It is generally created outside of professional routines and practices.

Broadcaster. The broadcasters are the users responsible for creating the video content in UGC live streaming video services. Normally, each broadcaster is authenticated on the service and owns a *channel*.

Channel. On live video streaming services, channel is the collection of videos that a determined broadcaster have produced. The different videos produced on live are called *sessions*.

Session. A session is a video stream created by a broadcaster. On live streaming systems, the channel can be either *online* or *offline*. Each online period results in a video session that can be, or not, archived. In this work we are interested by live sessions.

[1] https://www.youtube.com/
[2] https://www.wikipedia.org/

Viewer. Viewers are users that are not necessarily registered on the service that watch the sessions produced by the broadcasters. The number of viewers watching a session can change over time. We call channel *popularity* the total number of viewers of an online channel at a given moment in time. Figure 2.1 shows the evolution of the popularity of a given channel over time, this channel contains two sessions.

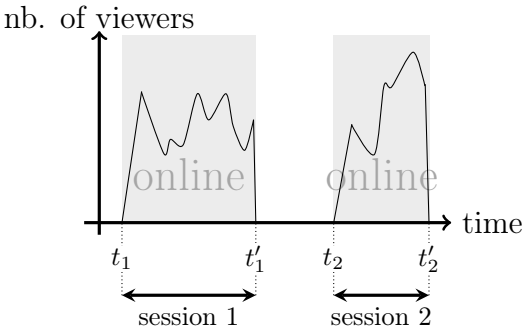


Figure 2.1: A life in a channel

Service Provider. The service provider is the platform where the broadcaster can upload its produced video content. The service provider is then responsible to process the content and deliver it to the interested public, namely viewers. Normally service providers have ways to monetize the platform, for example distributing in the beginning of the video some paid advertisement.

Content Delivery Network (CDN). CDN is a large distributed system of servers across the Internet. Normally deployed in the Internet *edge*, inside Internet Service Providers (ISPs) network and close to the end-users. The goal of a CDN is to serve content to end-users with high availability and high performance. The service providers pays the CDN to deliver its content to the users. The CDN then pays ISPs for hosting its servers in their data centers.

Adaptive Bit Rate (ABR). ABR streaming is a technique of video streaming. On ABR a single video content source is encoded at multiple different resolutions and bit rates, called video *representations*. The user video player can then switch between the different representations depending on its device and network capacities. More specifically, the current implementations, like DASH (standard created by Moving Picture Experts Group (MPEG)) and HTTP Live Streaming (HLS, implemented by Apple Inc.), work over HTTP. The multiple representations are segmented into small parts, typically between two and ten seconds.

A manifest contains the information about all the multiple available representations and segments. At the start, the client requests the segments at lowest bit rate. If the client has enough bandwidth to download a bigger bit rate it will request the next higher bit rate segment. If the network conditions deteriorate, the player will request a lower bit rate segment. ABR can achieve little video buffering and better user QoE.

Transcoding. Transcoding is the process that transforms a video source into a different representation, i.e. a version of original video with a different resolution and/or bit rate. This process is needed for ABR streaming.

The process to absorb and deliver the sessions varies among service providers. Based on the live streaming services solutions, there are mainly two general types of processes.

Traditional process. The traditional process consists in preparing the raw live stream (e.g. for sanity check and better webpage integration), and then delivering it directly to the viewers requesting it. Figure 2.2 illustrates this process. The raw video is produced by a broadcaster and sent to the system by the ingest servers. The system is then in charge to deliver this video among heterogeneous viewers.

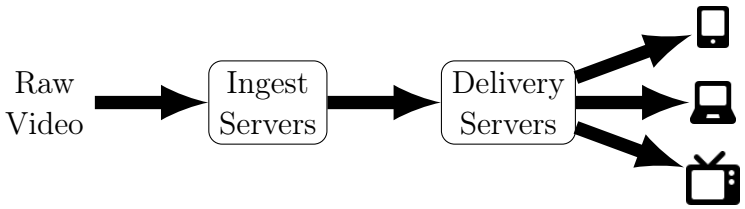


Figure 2.2: Traditional process of live video delivering

Transcoding process. The transcoding process is illustrated by Figure 2.3. It consists of transforming the raw live stream into multiple live video streams and of using ABR streaming to deliver the session to the viewers, typically with a standardized technology like DASH [Sto11]. Using the transcoding process the system is able to deliver different versions of the initial raw video for each viewer, accordingly to its device capacities and network conditions.

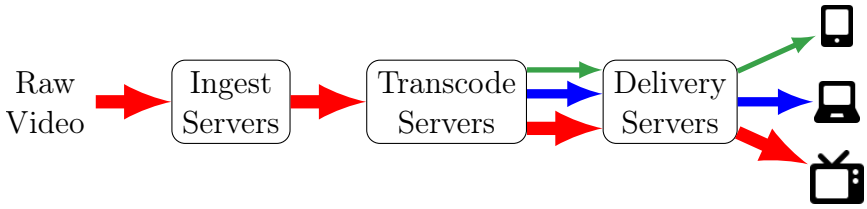


Figure 2.3: Transcoding process of live video delivering

2.3 Live Video Streaming Traffic Studies

Many papers have studied video network bandwidth usage over the Internet. In particular, the authors of [IP11] have dealt with more than five years of users web traffic data to examine different characteristics of Internet usage. They highlighted the increasing importance of video content (up to 28% within the five years). In [FMM+11], the YouTube traffic generated by mobile devices is compared to the traffic generated by regular desktop computers. Their results showed access patterns, which are similar across the sources of traffic. In the example illustrated by Figure 2.4 we observe that people using different devices and networks are interested by the same type of content in YouTube: short videos. In both environments, half of the population watches videos shorter than 4 minutes and smaller than 20 MB. In [ZLAZ11] the total amount of YouTube videos allows the authors to draw conclusions about the bounds of total bandwidth and storage space that is necessary for YouTube. This study emphasizes the critical needs of resources for video systems. The video traffic generated by YouTube is analyzed from the standpoint of an ISP in [AJZ10]. Overall, these studies have emphasized the importance of services like YouTube over the whole Internet traffic and the exploding needs of resources to serve the population. For our first contribution, we use similar techniques to analyze the behavior of people who contribute to a live video service as well as the bounds of total bandwidth usage for live videos delivery.

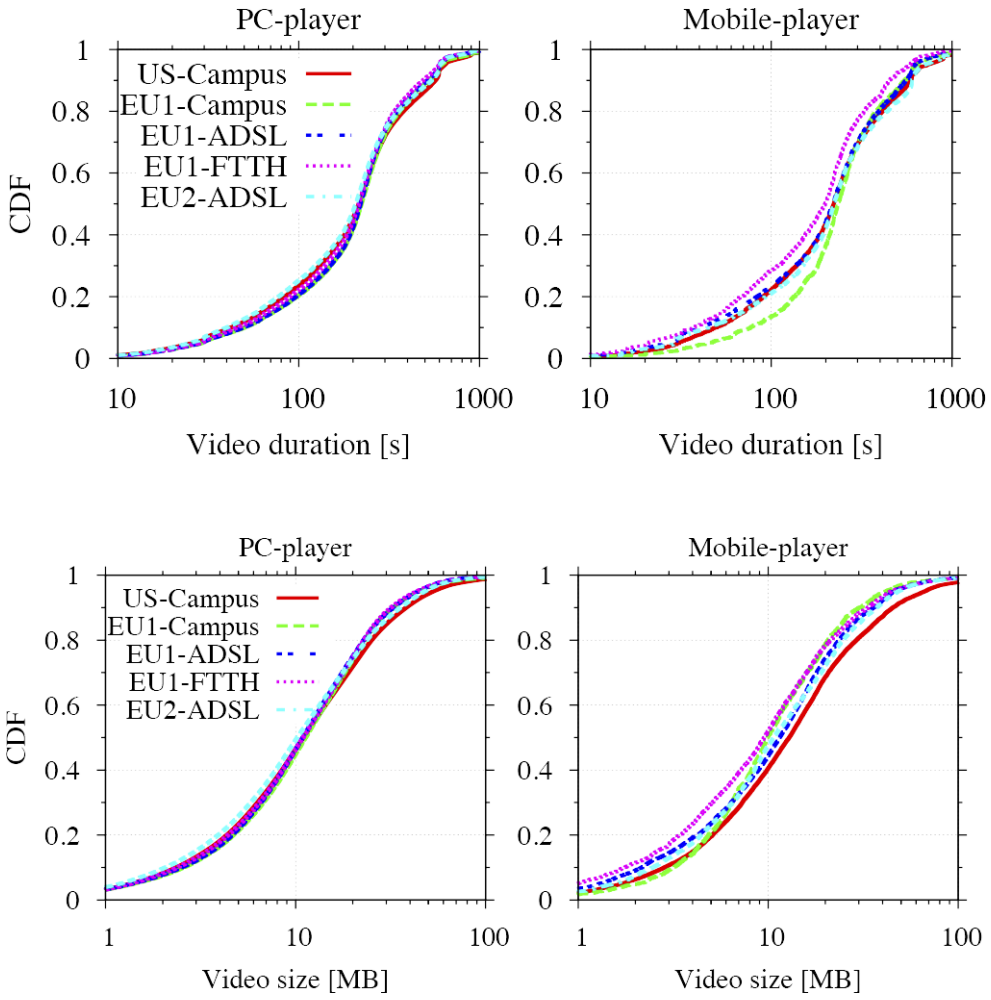


Figure 2.4: CDF of video duration (top) and video size (bottom). Extracted from [FMM+11].

Regarding live video streams, fewer works have been published. One of them analyzed the Zattoo system [CJW09], one of the largest production live streaming providers in Europe. In this paper, the authors had information about the network architecture from the provider point of view. However, regardless their claims for it, Zattoo cannot be considered as a large-scale system. The peak load they presented from Zattoo is one order of magnitude lower than what we observed in Twitch. An extensive study was done over data collected from the largest CDN in China during 2008 Olympic Games [YLQ+09]. The authors highlight the new demands that such events and dynamics impose over live systems. In particular, they characterize Internet Protocol Television (IPTV) systems regarding time dynamics of content provider and users activities. This work, however, focuses on regular content produced by a few well-established video channels. Similarly [QGL+09]

shed some light at these time dynamics over data collection from a large scale IPTV provider in the United States. Additionally, they proposed models for probability distribution of user activities and an IPTV user activity workload generation tool.

One of our contributions, a data set of live sessions from two real services, differs from the previous works by exploring aspects of live video streaming systems that are present in two different traces. The large scale produced by the UGC component of these real services stress the importance of our data set. We discuss further the works related to UGC platforms.

2.4 User-Generated Content

Many measurement campaigns have been conducted to understand the motivations of contributors to UGC platforms. In particular, the YouTube system has been extensively studied since 2007 [CKR+07]. Typically, a study of YouTube uploaders behavior is given in [DDH+11], where it is explained that the most popular uploaders upload copied content. However, to the best of our knowledge, only few papers have addressed data traces with user-generated in *live* platforms. Two of them deal with "gamecasting", *i.e.* gamers capturing and broadcasting their activity within a game. In the first one [KSC+12], eSport and Twitch users behavior are discussed. During a 100 days trace, evidence of the relationship between peaks of popularity in the platform with the major eSports events were shown and are illustrated by Figure 2.5. A prediction of session popularity based on its early popularity is proposed, while in our contribution we offer prediction based in the past sessions. Typically, scheduling of tasks related to the session processing and delivering should be done as soon as the session starts, therefore at this point of session life cycle early popularity is not available. The authors of [SI11] study XFire, which is a social network for gamers featuring live video sharing. The authors focus on analyzing the similarities between the activity of gamers in XFire and their activity in the actual games. Another study dealing with live video sharing is [VAJ+06]. The authors analyzed 28 days of data from two channels associated with a popular Brazilian TV program that aired in 2002. Our contribution differs fundamentally in a quantitative manner since we evaluated several thousands of channels.

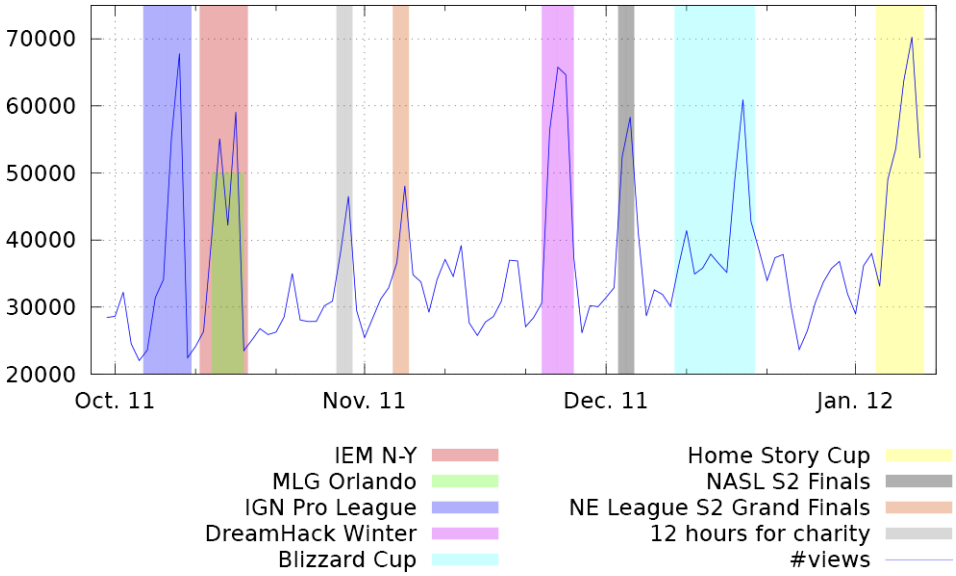


Figure 2.5: Average viewer count each day related with major eSport events. Extracted from [KSC+12].

P2P is another line of research dealing with user-generated live streaming. A survey is given in [ZH12]. Many papers claim to own traces from popular live streaming systems, for instance PPLive and PPStream, but none of them make a thorough study of these traces. In most cases, authors cannot have an accurate view of channel popularity due to the distributed nature of these systems. These measurements, typically for PPLive, are also limited to a small number of channels (320 in [HLL+07]), for example our data set characterizes millions of channels. Moreover, the most popular systems are almost exclusively used in a specific region. Thus the population represented in these systems is locally biased and is not representative for a global view. Finally, many works focus on characterizing the P2P topologies and the behaviors of peers in terms of bandwidth contribution [VGN+12], but such studies are not relevant in the context of our work.

We discussed the state of the art related to the importance our data set contribution. The remaining of our contributions focus on solutions for delivery servers and transcode servers, essential components of the live video streaming chain. We follow with a discussion of the state of the art regarding these components.

2.5 Multimedia Delivery Architectures

As previously mentioned, the delivery servers are critical components of live streaming services. In this section we present the various architectures present

in literature and industry. We discuss as well, the benefits and weaknesses of each type of delivery.

2.5.1 Video Delivery Models

We introduce the several options that service providers use to build their "delivery cloud".

Data Center (DC). The most common way to deliver content is to use a DC, which is basically a large set of servers [BH09]. The DC can be either owned or rented by the service provider. In the former case, the infrastructure is almost exclusively paid at the construction, however it has some fixed capacity limitations. In the latter case, usually referred as *cloud*, the infrastructure can scale up and down on demand but the service provider has to deal with another actor (the cloud provider). Although DCs are attractive, easy-to-manage infrastructures, they do not enable low response time for a large population of users because they are located in one location (or few locations if the service provider deals with several DCs) [CWSR14]. That is, the network latency is too high for a vast fraction of the population because transit network between final users and service providers are large. Moreover, the monetary cost to transfer data is higher because the traffic should cross several networks until the content eventually reaches the users.

Peer-to-Peer (P2P). The challenge of delivering multimedia content on a large scale is essentially a problem related to the reservation of physical resources, such as outgoing network bandwidth. To address this problem, the scientific community has advocated for years for a P2P based infrastructure, where users themselves contribute to the delivery by forwarding the content they received. A lot of algorithms have been designed to improve the delivery performances [Pas12]. However, various constraints have limited the deployment of P2P systems for commercial purpose. First, firewalls and NAT still prevent many direct connection between users [JDMP11]. Second, P2P require users to install a program on their computers. Such a "technical", security-sensitive requirement can prevent users from using the service. Moreover, despite some new browser-based technologies (e.g., WebRTC), a P2P software depends on the configuration of the computer of end-users, which is a cause of many development difficulties. Third, the service provider has a low control on the users QoE since it does not directly control the performances. Finally, the complexity of P2P system can increase the delay. Many initiatives have aimed at ensuring that peers connect

preferentially with the other peers that are located in the same network [SKS09], which improves latency. However, a peer usually gets data from multiple other peers. Even if the direct connection between two peers has low latency, aggregating data from multiple peers requires synchronization and buffering, which causes extra-delay.

Content Delivery Network (CDN). In the recent years, CDNs have emerged as the privileged way for large-scale content delivery. For example, Akamai, one of the largest CDNs, is responsible for 15% to 30% of all web traffic [Inc]. CDN is composed of three types of communication devices: a relatively small number of *sources*, which directly receive the content from the service producer, a medium size network of *reflectors*, and a large number of *edge servers*, which are deployed directly in the access networks, close to the users. The proximity between the end-users and the edge-servers makes network latency small.

For a decade, the CDN providers have met the demand of two families of actors in the value chain of content delivery: service providers (because large-scale Internet services have to be distributed for redundancy, scalability and low-latency reasons) and network operators (because minimizing inter-domain traffic while still fulfilling their own users' requests is a business objective). CDNs have thus emerged as a new category of market players with a dual-sided business. They provide caching capacities "as a service" to network operators and they provide a distributed hosting capacity to service providers. The CDNs provide both scalability and flexibility, they deal with the distribution complexities and manage multiple Internet operators. They propose all of these services at a unique selling point.

Several works confirm that edge-servers can be used for serving other types of information besides static content. As studied in [Pas12], current CDNs infrastructures have the ability to serve millions of end-users and are well-positioned to deliver game content and software [Ale12]. However, CDN edge servers are generally built from commodity hardware that have relatively weak computational capabilities and often lack graphics processing units (GPUs), which can be restrictive for transcoding operations.

2.5.2 Composing Hybrid Delivery Models

There is no clear consensus about the best solutions to deploy. Typically for video streaming, we observe that the main actors have made different choices. To name a few:

Google uses multiple DCs distributed over the globe to delivery their services, including YouTube [AJCZ12];

NetFlix used a composition of multiple CDNs on its delivery chain by 2012 [AGH+12]. In 2012 they started to deploy their own CDN [ope12];

LiveSky is a composition of P2P assisted by CDN to improve viewers QoE was deployed on LiveSky [YLZ+09a];

Twitch is the major live streaming service, it uses its own private DC assisted by CDN [Hof12].

A recent trend is to build *hybrid* delivery models that compose several of the aforementioned models. We list hereafter some frequent compositions, each one with its own pros and cons.

CDN-P2P. Such a composition is managed by either the service provider, as shown in [MCAB12] or the CDN provider, as described in [AZL+12]. The CDN gives some guarantees on the QoE by offering a minimum amount of resources and by reducing the response time. This composition is illustrated by Figure 2.6. The CDN also allows users behind NAT to be properly served. On its side, the P2P system assists the CDN in case of traffic peaks. The more users to be served by the system, the more resources in the system. The potential problem with such a composition is that service providers can not control all parts of the delivery chain. Indeed, most profits come from a clear understanding of the demand from end-users and a capacity to adapt the delivered content to every user (e.g., embedded advertisement). Another potential problem comes from the lack of QoE guarantees. Finally CDN-P2P compositions suffer from the same drawback as P2P alone, including the requirement of installing a software on users' computer.

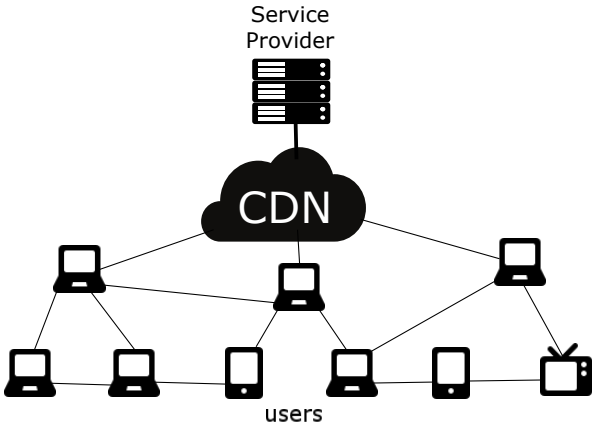


Figure 2.6: CDN-P2P

Multi-CDN. The service provider is commonly the main manager of this composition. A typical example of such a composition is depicted on Figure 2.7. This composition has been thoroughly studied in [AGH+12]. The main idea is that the service provider relies on several CDNs to deliver the content. For each user, the service provider decides the CDN in charge of serving this user. The main advantage in this composition is the possibility to achieve the best QoE for the viewers with the lowest cost by exploiting the different prices applied by each CDN. Another advantage is that the delivery is more robust since a downtime from one CDN provider can be mitigated by using another CDN. However, this delivery is only based on third-party actors, which means that even the consolidated background traffic is dealt in a pay-as-you-go way. Therefore the overall price of this composition can be higher than others that include self-own infrastructures.

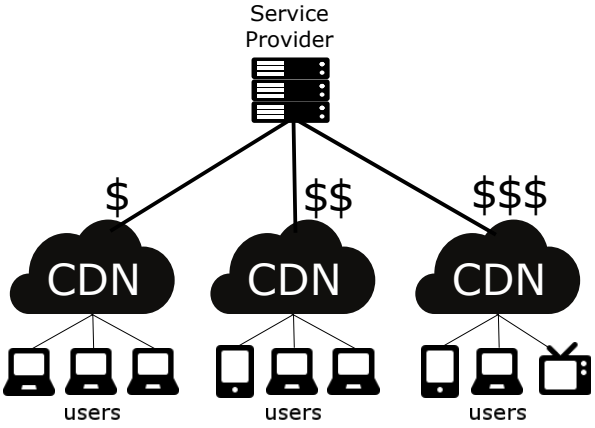


Figure 2.7: Multi-CDN

Multi-DC. Service providers that want to provide features beyond the basic delivery of the same content are interested in hosting the service in their own servers in a DC. However, response time requirements force service providers to deploy multiple DCs in order to serve the whole population with low response times [BYGJ+09]. This composition is shown at Figure 2.8. In this case, it becomes crucial to manage the traffic such that the load is well balanced among the different DCs [LLSY11] and to manage the sharing of content over the multiple DCs [LSYR11]. The advantage is a lower cost per gigabit per second (Gbps) than Multi-CDN, the total control of the delivery chain, and a relatively low response time since every end-users should have a DC nearby (so latency is reduced). The cons include substantial high cost for the initial deployment for multiple DCs.

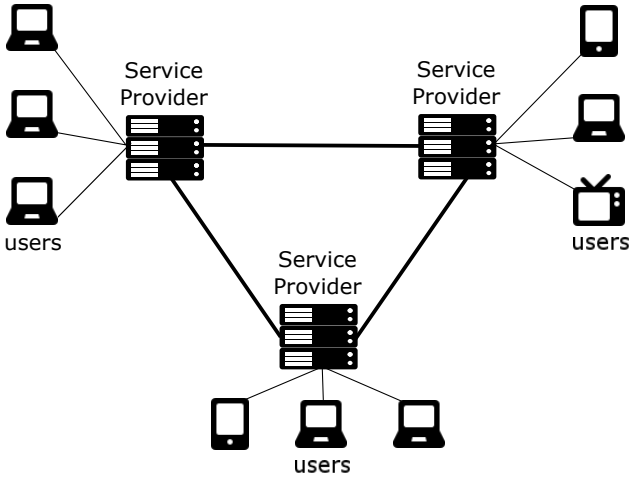


Figure 2.8: Multi-DC

DC-CDN. In order to mitigate the disadvantages of the aforementioned models, it is frequent that video service providers deploy hybrid DC-CDN compositions [Hof12]. Figure 2.9 illustrates this composition. DC-CDN hybrid composition is expected to combine the main advantages of both delivery solutions at a minimum cost. The high prices paid for CDN are minimized by using the CDN resources only when the DC is out of capacity, usually at traffic peaks. The DC dimension is adjusted so that the consolidated background traffic (or valleys of usage) is dealt by their own servers. In the cloud computing context such a composition is often called hybrid cloud where conventional data centers and cloud solutions are deployed together to aim the same combined advantage [AFG+10]. Various studies have shown that it is not trivial to

outsource tasks from the internal data centers to the external delivery infrastructure [BVB10], typically due to security [SSSL12], QoE [ZJY+09] and economics [BVB10] reasons.

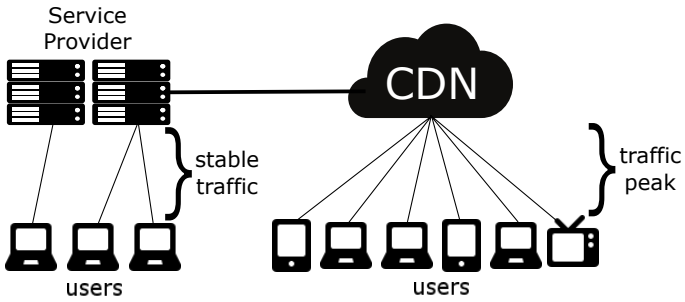


Figure 2.9: DC-CDN

We have presented diverse architectures found in both research papers and the industry related to the delivery of video streaming. As shown, there is no consensus regarding the delivery model. Each provider adapts its infrastructure accordingly to its resources or needs. In our contributions, at Chapters 4 and 5, we explore this discussion further and present some approaches that benefit from the conclusions derived from the live streaming data set, our first contribution.

2.6 Video Transcoding

As we previously described, transcoding operations are essential when implementing ABR. Specially cloud-based transcoding has been the subject of several papers. Most of these works [GB06, JAL+13, LZG12, LZY+13, HMLW11] take advantage of the fact that some modern video compression techniques divide the video stream into non-overlapping Group Of Pictures (GOPs) that can be treated independently. The encoding time of each GOP depends on its duration and on the complexity of the corresponding scene. The algorithms exploit this fact to increase the utilization of each computing node at the expense of an increased complexity, including the time and resources needed to split the input video into appropriately sized GOP.

One downside of these solutions is that they need to know the transcoding time of each GOP in order to assign it to the most suitable computing node. Some authors [GB06, JAL+13] propose fairly complicated systems to estimate the encoding time of each GOP based on real-time measurements, while others [HMLW11, LZG12, LZY+13] assume that this information is directly available, for instance [HMLW11] by profiling the encoding of a few representative videos of different types. Another downside of a GOP-based solution

is that the encoding of each GOP can be completed out of order and then need to be reordered before being delivered to the users. This out-of-order problem is especially important when dealing with live content, which requires real-time constraints. Only one work [HMLW11] explicitly considers real-time constraints in a GOP-based system. Further, the cloud is explored for parallelizing multiple videos transcoding and scheduling algorithms to minimize the overall encoding time [LZG12, LZY+13].

Some studies have explored the CDN resources for transcoding. A study [WSW+14] proposes to leverage underused CDN computing resources to jointly transcode and deliver videos by having CDN servers transcode and store the most popular video segments, illustrated by Figure 2.10. Such a solution can offer significant gains, especially for non-live popular streams, but it requires the cooperation of the CDN, which is not always owned and operated by the cloud provider.

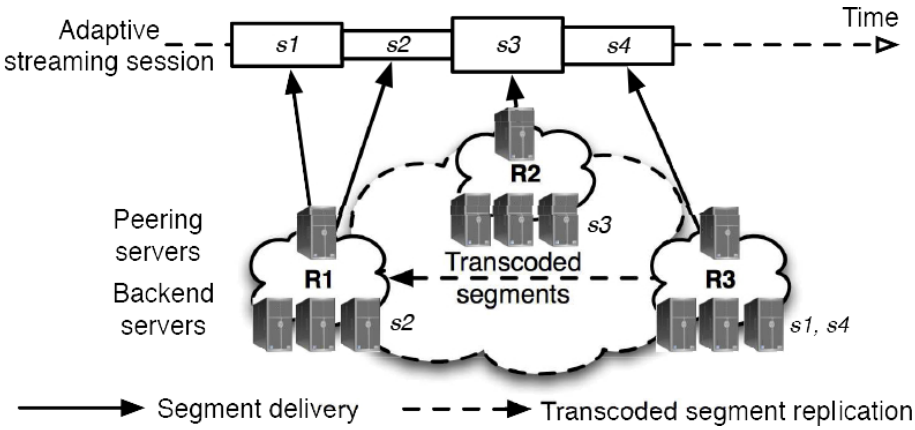


Figure 2.10: An illustration of joint online transcoding and geo-distributed streaming. Extracted from [WSW+14].

Few papers have studied the relationship between Power consumption, Rate and Distortion (often abbreviated as P-R-D). The first paper to investigate the P-R-D model [HLC+05] contains a detailed analysis and corresponding model of the video encoding process. The authors use this model to define an algorithm that, given rate and power constraints, minimizes the distortion of the compressed video. A different definition for the distortion is given by [SLW+09] which proposes an algorithm to solve the P-R-D as an optimization problem. These works deal with *a single video flow* and take the rate as an input parameter, they do not address how to choose this value as in our transcoding contributions.

Results of an empirical study based on the H.264 Scalable Video Coding (SVC) reference software JSVM-9.19 [RSW07] are presented by [YCWF11].

While non-SVC H.264 can be considered as a special case consisting of only one layer, the authors emphasize the results related to the SVC part. Since their data is not publicly available, and is not possible to extract the numbers from the paper, we run similar experiments with a wider range of parameters (bigger video bit rates and video resolution) leading to the data set that is presented in Appendix C.3.3.

2.7 Adaptive Bit Rate Streaming

Adaptive Bit Rate Streaming is a technique used to deliver video content. It detects the client conditions (for example download bandwidth, screen size, etc...) in real time and adjusts the video quality accordingly. On the server side, it is necessary to create multiple video bit rates, normally called *representations*. To create the representations, transcoding operations must be performed on the original video (in the live scenario it is the video content send by the broadcasters). A *manifest* file is downloaded by the client and contains the information about all the available representations of the video. Each representation is composed by *segments* that have usually few seconds. The client is then in charge of choosing a representation and download the segments: it starts by downloading the lowest bit rate one. If, for example, the client download speed is greater than the segment bit rate, it will request the next higher bit rate segment. Figure 2.11 gives an example of manifest with the video representations and the client decisions.

Delivery bandwidth can be saved with the implementation of ABR. For example, a user can quit watching a video that is not yet finished. All the remaining video data that have been downloaded is then wasted. Figure 2.12 illustrates an example comparing the traditional progressive download and ABR on YouTube [KBZ13]. In this example, the amount of extra data downloaded by employing ABR at any time t_s is $d(t_s + t_{switching\_interval}) - d(t_s)$, where $t_{switching\_interval}$ is usually between 2 and 10 seconds. In the case of YouTube progressive download, the amount of extra data downloaded is up to $d(t_s + 50) - d(t_s)$. At any moment during video play, the progressive download implemented by YouTube downloads up to 50 seconds of extra data. Therefore, accordingly to the ABR client choice in terms of switching interval time and number of segments, ABR can save bandwidth costs by limiting the amount of extra data download.

One of our contributions is an analysis of the bandwidth reduction that ABR could perform on Twitch. A similar study have been done for VoD of YouTube [KBZ13]. Using traces collected on the University network, the implementation of DASH obtained considerable reduction in overall bandwidth

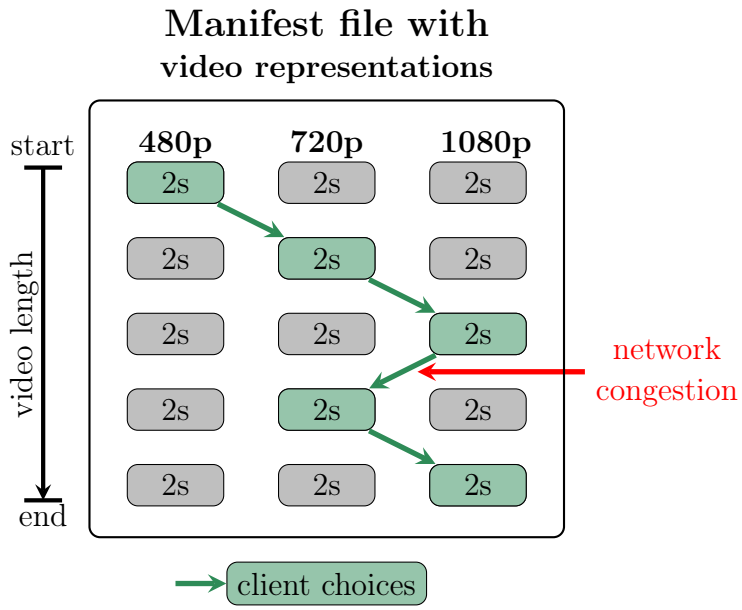


Figure 2.11: An example of adaptive streaming manifest and client choices over time.

usage. For the cases where the viewers do not entirely watch the video the reduction of bandwidth was 95% for lower quality videos and 83% for the high quality ones.

The DASH standard [Sto11] is a popular ABR design. This standard has been the common choice of various academic studies and industrial implementations. As we previously presented in Section 2.2, ABR can demand extra computing power for the transcoding of different video representations. This overhead was evaluated on DASH for low latency scenarios. The overhead represents only 13% of the overall video streaming process [BCF14]. Even with a relatively small overhead for one live session, it can generate an important stress on infrastructure in the case of UGC live streaming, where the scale of concurrent sessions can be of thousands broadcasters. We explore in our contributions the trade-off between the benefits of the ABR and this overhead of computing power for the transcoding operations.

A CDN live DASH approach is analyzed by [LSRT14]. Both theoretical formulation and practical implementation are given. The focus of this work is to maximize viewers QoE subject to under provisioned CDN infrastructure. An implementation of DASH assisted by P2P has reduced the servers outgoing network bandwidth up to 25% thanks to the peer assistance [LMT12]. An optimization based on viewers QoE for the DASH standard for VoD services was explored in [JdV14]. Similar our transcoding contributions aim viewers QoE but on live services. We consider service providers assisted by CDN

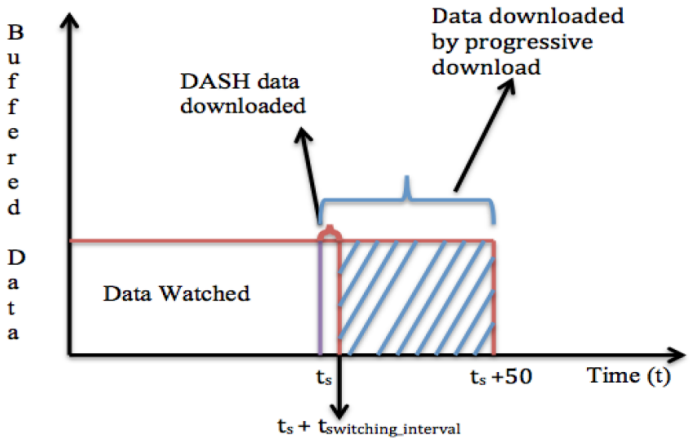


Figure 2.12: An illustration of potential data savings by employing DASH. Extracted from [KBZ13].

that have the needed infrastructure to satisfy all viewers. And in one of our contributions the bandwidth burden on the service providers servers is reduced by migrating it to the CDN.

A dynamic scheduler for transcoding jobs of DASH that allows high-priority process and load balance in a cloud environment was designed in [MSZ14]. Although near to live transcoding, their objective was different from ours. They aim for video completion time, system load balance, and video playback smoothness while in our contributions we target viewers QoE and resources costs reduction.

2.8 Conclusion

Video streaming and live services have been interesting topics of research for decades. The challenges raised on such system have inspired many contributions and yet many issues remain open. The ever increasing consumption of video streaming have stressed the previous solutions and current infrastructure. Different restrictions were applied by major live streaming services to cope with the scale problem. Finally, there is no consensus, both in the academia and industry, on an ideal infrastructure for these systems.

Many studies had been made for VoD services, like YouTube. But, as we presented, these studies are not extended to live services. Additionally, the data sets used in these studies are generally not made publicly available, which restraint the usage of the results. In terms of proposed solutions for delivery and transcoding, academic works have addressed mostly P2P solutions, which

can restrict the implementation on real services. The targeted scale of the live streaming solutions present in the literature are small compared to the one present on UGC services.

Our contributions, which will be presented in the next chapters, target large scale UGC live video streaming services. With a data set of millions live sessions, which is publicly available, we hope to help the community to achieve reproducible solutions. Our data set analysis reveals important live streaming properties. We design solutions for delivery and transcoding of live streaming that are tailored for the cloud or assisted by CDNs, which can enable the implementation to real services.

Chapter 3

Live Streaming Sessions Data Set

Contents

3.1 Introduction

The flexibility of Internet communication to accommodate all types of content, including video media, has led to a considerable effort from the research community, as discussed in the previous chapter. The popularity of UGC live streaming services has however not grown as fast as some expected. Yet, the last couple of years has seen a surge of interest for some new usages, including crowdsourced journalism [MWN+13] and eSport [KSC+12]. The release of the live version of YouTube [You13] by the end of 2013 has boosted the attention on UGC live streaming services, especially on the current leader in the sector, namely Twitch [Hof12]. Indeed, the growth of popularity for UGC live streaming system has the potential to impact industrial actors in various

areas, from CDN providers to TV broadcasters, so there is a need for a better understanding of the main characteristics of these UGC services.

We present in this chapter a data set with two live streaming services. To better understand these services, we started by collecting information of the produced live sessions. This data set aims to provide the community traces that allows anyone to replay the sessions that were diffused by the services providers, namely Twitch and YouTube Live. The information present in the data is the properties of each session and broadcaster, as well the amount of viewers watching each session. We constantly fetched the Application Programming Interface (API) information, further detailed in Section 3.3, each five minutes during three months, January 6, 2014 to April 6, 2014.

This chapter also comprises an analysis of the two live streaming service providers. Along with the description of the data set, we introduce parameters that influence how video streams are prepared and delivered. With the data fetch from their APIs we could determine many characteristics of the services. We then developed solutions based on the findings of our analysis that are presented in the next chapters.

To design solutions for live streaming services is essential to understand the system behavior. In our simulations we use the data set to reproduce the real services load and identify some key characteristics of UGC live streaming services. The goal of this analysis campaign was in particular to give answers to some critical questions:

How big are today's main UGC live streaming? These services are still frequently under-estimated. An answer to this question typically matters for the Internet actors dealing with hardware infrastructure. Statistics include the number of channels, the number of viewers and most importantly the evolution of these numbers over time.

Are these services really 24/7? The TV broadcasters have a long experience of programming TV shows so that the right content is available at the right time for the population. An UGC system can barely be "programmed" since it depends on its broadcasters, who generously contribute. There are still doubts about the sustainability of the offer (the aggregated set of online channels).

How heterogeneous is the popularity of channels? The implementation of large-scale delivery services commonly relies on CDN, which leverages the high heterogeneity of the popularity of content. Typically caches are efficient because the most popular content are highly accessed. It has been shown that many large-scale services have a popularity that can be modeled by a Zipf's law [AH02] where the parameter

α can range from 0.5 (for the most homogeneous services) to more than 2 (for services with a high heterogeneity).

What is the behavior of broadcasters? The management of an UGC service requires a good understanding of the behavior of broadcasters, in particular how often a channel is online, how long does it stay online, and how often it switches on. Characterizing broadcasters is critical for UGC service providers, which need to predict behavior, typically to adjust the infrastructure.

In this chapter we first introduce the live streaming services we crawled: Twitch and YouTube Live. Then, we present the methodology of data retrieval. Following, details about applied filters are given, these filters clean up testers and irrelevant sessions from the traces. Next, the answers to the previous questions are explored and the results from our analysis are discussed.

3.2 Live Streaming Providers

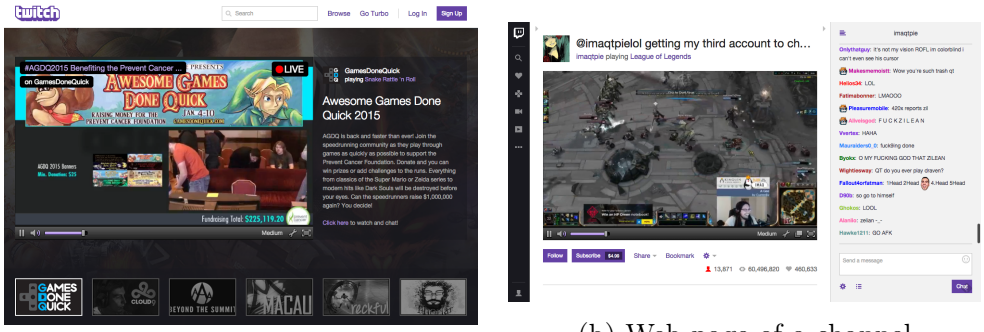
We collected data from two platforms to build our data set, Twitch and YouTube Live. Both platforms have an API available that allows to retrieve information about the sessions that are delivered at any given moment. First, we chose Twitch because it is the leader of the live streaming sector. Our second choice, YouTube Live, was made because it represents a strong competitor since YouTube is the leader of the VoD sector.

3.2.1 Twitch

Twitch is a live streaming video platform owned by Amazon.com Inc. It first appeared in June 2011, as a side service from Justin.tv focused on video game live streaming, including for example eSport competitions, coverage of gaming conferences and users playthroughs of video games.

Justin.tv was a general content streaming platform created in 2007 by Justin Kan, Emmett Shear, Michael Seibel and Kyle Vogt. On February 10, 2014, Twitch's and Justin.tv's parent company was rebranded from *Justin.tv Inc.* to *Twitch Interactive* in order to align with the company focus on the gaming content. On August 5, 2014, Justin.tv was officially shut down [Pop14]. In September 2014, Amazon.com acquired Twitch for $970 million [Ama14].

Figure 3.1a shows Twitch's home page, taken at January 6, 2015. In its home page are highlighted some Twitch partners. The program of partnership is offered to broadcasters that have average of 500 or more concurrent viewers



(a) Home web page

(b) Web page of a channel

Figure 3.1: Twitch web interface

and regular schedule of at least 3 times a week. Moreover, Twitch also proposes this partnership to broadcasters that are popular in other medias, as YouTube or Twitter.

An example of a Twitch channel web page is given on Figure 3.1b, also taken at January 6, 2015. Viewers can interact with the broadcaster and other viewers by the chat aside the video. Past sessions and highlights can be available for the viewers depending on the broadcaster choice of settings for his channel. In our work we focus only on the live aspect of Twitch.

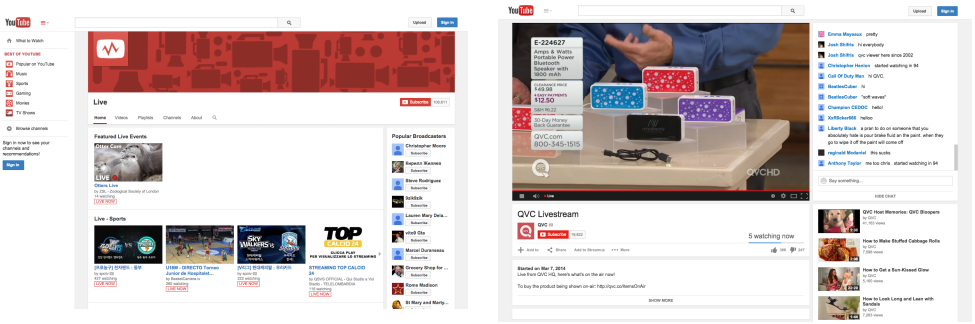
3.2.2 YouTube Live

YouTube Live is the live streaming service owned by Google Inc. Offered as the live branch of the biggest VoD service, YouTube, it provided live support to global events, such as popular concerts, sports events, and interviews. YouTube Live became available (in beta) to certain partners at April 8, 2011.[1] Two years later, the service became available for all YouTube verified accounts (confirmed by phone SMS or call) and with good standings (by following the community guidelines and copyright policies) at December 12, 2013.[2]

The YouTube Live interface follows the same design as the VoD YouTube. Illustrated by Figure 3.2a taken at January 6, 2015 is the home page. It lists channels that are online at the moment in time and some schedule sessions. The channel web page, at Figure 3.2b, also taken at January 6, 2015, follows as well the YouTube design for VoD channels. The difference between the VoD video page and the live version is the chat addition (similar to Twitch channel page) and the suppression of comments.

[1] http://youtube-global.blogspot.fr/2011/04/youtube-is-going-live.html

[2] http://youtubecreator.blogspot.fr/2013/12/now-you-can-live-stream-on-youtube.html



(a) Home web page

(b) Web page of a channel

Figure 3.2: YouTube Live web interface

3.3 Data Retrieval

Twitch and YouTube Live provide two different APIs, which allow anybody to fetch information about the current state of the systems. We used a set of synchronized computers to obtain a global state every five minutes (in compliance to APIs restrictions) between January 6, 2014 and April 6, 2014. We fetched information about the total number of viewers, the total number of concurrent online channels, the number of viewers per session, and some channels metadata. The data set, containing more than five millions sessions, is available on a public website.[3]

The YouTube Live API does not contain as many fields as the Twitch one. We summarized in Table 3.1 the main information that we fetched from both APIs.

	Twitch	YouTube Live
channel id	✔	✔
session id	✔	✔
number of viewers	✔	✔
video bit rate	✔	✘
video resolution	✔	✘
broadcaster country	✔	✘

Table 3.1: Summary of the APIs information

[3] http://dash.ipv6.enstb.fr/dataset/twitch-youtube/

3.4 Filters Used to Clean Up Traces

We observed in the measurements that a significant number of channels were typical from a broadcaster who tests the service. Two main behaviors were identified. The first one is a broadcaster who launched a channel for only one session with a duration lesser than ten minutes overall in the three months. In other words, there is only one occurrence of this channel over the whole set of traces. The second type of "tester" is the one who set a channel with sessions longer than ten minutes, but the channel has remained with no viewer at all during the analyzed period.

	Twitch	**YouTube Live**
Total number of broadcasters	1,570,844	248,563
Total number of sessions	12,352,691	818,857
Sessions with less than 10 minutes	3,940,330 (32%)	219,959 (27%)
Sessions with no viewers	1,692,233 (14%)	351,502 (43%)
Filtered number of broadcasters	1,094,094 (70%)	129,310 (52%)
Filtered number of sessions	7,763,331 (63%)	411,845 (50%)

Table 3.2: Overview of Twitch and YouTube Live scale and impact of testers

As shown in Table 3.2, testers represent a significant part of both Twitch and YouTube Live broadcasters with almost half of the registered sessions. These testers impact the transcoding infrastructure of the services but they harm the delivery infrastructure only on the up-link since no viewers request these sessions.

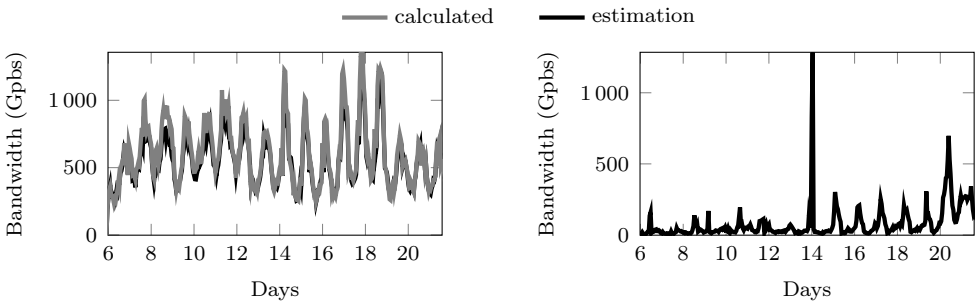
In the following, the testers (30% of broadcasters on Twitch and 48% on YouTube Live) are discarded from our measurements in order to keep attention on the filtered broadcasters. Nonetheless, the ability of UGC live streaming systems to prevent testers to harm the service, especially at activity peaks, is a concern that deserves some further studies.

3.5 Status of Live Streaming Services

3.5.1 How Big are the Systems?

The first question to answer is how big the live streaming system can be. To approximate how much bandwidth is used by each of the systems we summed up the bit rates multiplied by the number of viewers. In the case of YouTube Live, where the bit rates are not available, we attributed the average value of 2 Megabit per second (Mbps) from Twitch sessions as the bit rate for all YouTube Live channels.

In Figure 3.3 we present the results of the bandwidth approximation. The *calculated* line indicates, for both services, the sum of average bit rate (2 Mbps) multiplied by the number of viewers in each channel. The *estimation* line indicates for Twitch the sum of each session bit rate multiplied by the number of viewers. It was not possible to make an estimation for YouTube Live since its API does not offer the sessions bit rate. Both services had peaks of bandwidth of more than 1 Terabit per second (Tbps) on the 14th day. On Twitch these peaks near and over 1 Tbps are frequent. This information about the volume of bandwidth consumption is not only important for the live streaming services themselves but also for ISP and operators, who need to deliver all this content information to end users. Also remind that this content is live, and therefore it can not be pre-fetched or previously cached, and there is new content at every moment.



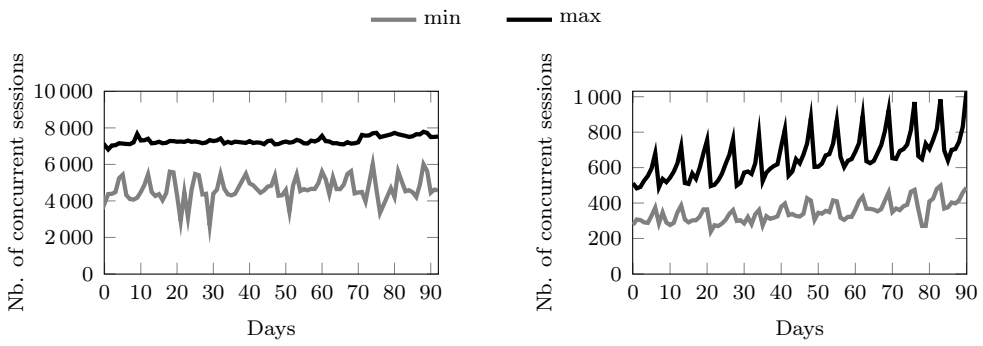
(a) Twitch (b) YouTube Live

Figure 3.3: Bandwidth consumption estimation for live video delivery

Another characteristic we evaluate on the live streaming platforms is the number of concurrent online channels. These platforms essentially differ from other UGC systems by the fact that the content must be produced at the same time viewers consume it. This imposes to the system a need of constant amount of online broadcasters in order to provide an interesting range of content to the viewers.

Figure 3.4 shows the minimum and maximum number of concurrent online channels registered per day. This is a useful metric to estimate the limits of computing power needed and thus the data-center dimensions. In Twitch between 4,000 and 8,000 concurrent sessions always require data-center processing. While YouTube Live concurrent sessions ranges from 200 to 350.

Furthermore, we are interested on the evolution over time of the channels and in discovering how many different broadcasters were registered on the analyzed data. Figure 3.5 shows the total number of different channels registered (unique *channel_ids*) over the analyzed period. Twitch has a bigger number



(a) Twitch (b) YouTube Live

Figure 3.4: Number of minimum and maximum concurrent sessions by day

of unique channels over time when compared to YouTube Live. Moreover, the detected growth of Twitch users is also significantly bigger than YouTube Live. This can be explained by a consequence of YouTube Live decision to give access to the live feature gradually to users [You13]. Although YouTube Live had this restriction, the number of testers are similar to the ones found on Twitch.

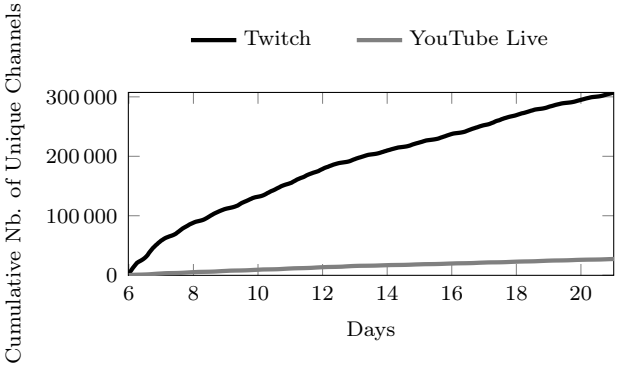


Figure 3.5: Cumulative number of unique channels by day

3.5.2 Are they 24/7 Services?

We have to recall that live video streaming essentially differs from other UGC services like VoD in the sense that the service depends on the activity of broadcasters *at every moment in time*. There is a critical need for online channels. Fortunately, both services have loyal broadcasters, who manage to be more consistently active (here online) than on other typical UGC platforms. It thus guarantees service continuity.

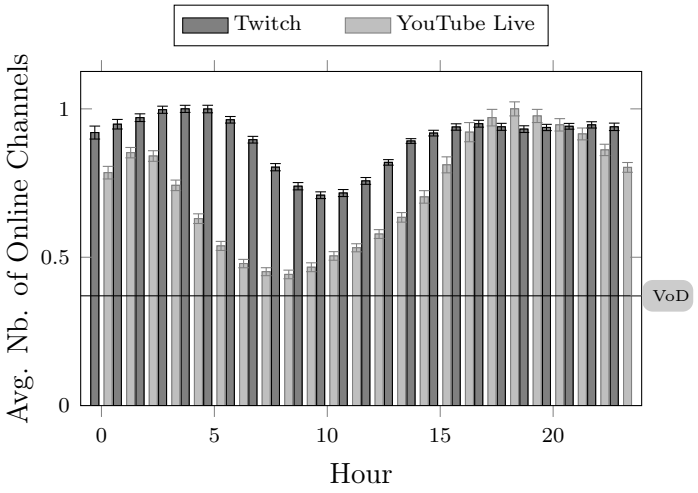


Figure 3.6: Average number and confidence interval of simultaneous online channels by hour

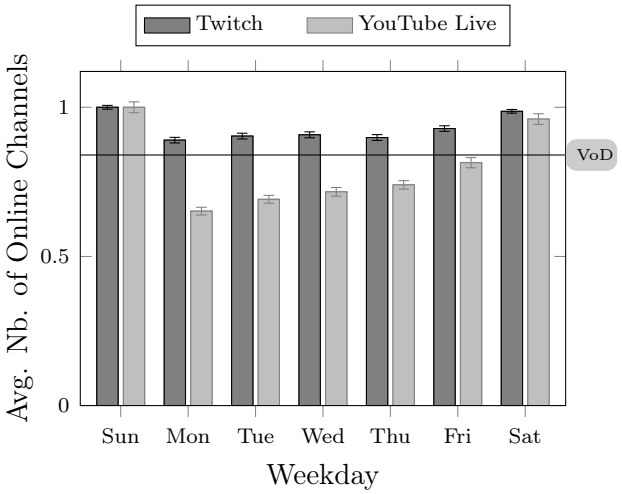


Figure 3.7: Average number and confidence interval of simultaneous online channels by weekday

We measured the number of simultaneous online channels over the collected data. We then computed the average numbers per hour of a day (respectively per day of the week). With these values we are able to measure diurnal (respectively weekly) patterns. We show our results in Figure 3.6 and Figure 3.7. In both figures, we normalize the results so that the peak of the number of online channels is equal to 1.

Our first observation is that Twitch is less sensitive to both diurnal and weekly patterns than YouTube Live. For the weekdays, the difference between the lowest number of online channels (0.9) and the peak (1) is not significant on Twitch (we note it 0.90 : 1). At least two explanations can be advanced. First the number of broadcasters in Twitch is one order of magnitude bigger than in YouTube Live and they come from a wider number of countries spanning the whole planet. The release of a closed beta of YouTube Live service was in 2013, and the full availability for all users was in late December of the same year. Second, the fact that Twitch channels are related to eSport can have an impact too. Indeed, eSport are especially popular in Asia. Since both Twitch and YouTube Live are already popular in Europe and America, all continents are covered by engaged broadcasters.

We now compare the diurnal and weekly patterns globally. The main important point to notice in Figure 3.6 is that the diurnal pattern is weaker than what has been observed on other UGC platforms, such as VoD. We indicate with a horizontal line the lowest number of new uploaded videos as it was measured for the YouTube VoD service (discussed in [CKR+09] and [CSF10]). The diurnal difference on Twitch is 0.65 : 1 although it is as low as 0.37 : 1 on YouTube VoD.

3.5.3 Zipf's Law in UGC Live Streaming

The distribution of popularity found on UGC systems and VoD typically follows the Zipf's law [AH02]. The Zipf's law function is given by Equation 3.1. The function variable α is the Zipf rank exponent. This exponent will dictate the popularity homogeneity. The bigger is the exponent, the bigger is the difference of content popularity. For example, the difference between 0.5 exponent and 2, is that the difference between the popularity of the content ranked as first and second (as well for other ranks) is bigger for exponent 2.

$$F_i(x) = A_i x^{-\alpha} \tag{3.1}$$

We checked with our traces whether live videos followed Zipf's law as well. First, we represent in Figure 3.8 examples of popularity distribution found on YouTube Live traces at two different hours picked on January 6, 2014. With the traditional logarithmic scales, we then produced an approximation of the Zipf parameters using a fitting curve process on the R software [R C14].

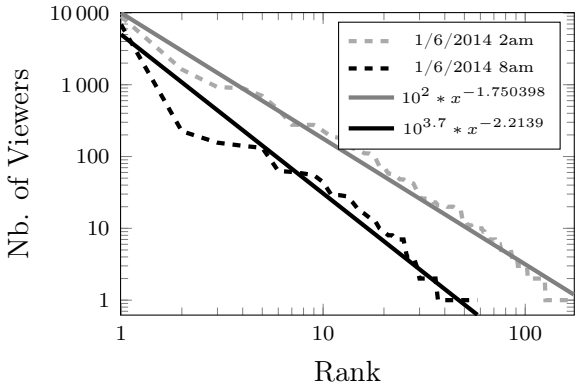


Figure 3.8: Number of viewers by rank (dashed lines) and Zipf approximation (solid lines)

To understand the evolution of the popularity pattern on both systems we calculated, for every five minutes of our traces, the same Zipf approximation formerly mentioned. We validate the results of the approximation by calculating the Normalized Root-Mean-Square Deviation (NRMSD) between the real data and the fitted curve. The mean NRMSD value obtained for YouTube Live was 0.0365 and the 95% confidence interval between 0.0362 and 0.0368, meaning less than 4% error in our fittings. For Twitch NRMSD value obtained was 0.0095 with confidence intervals of 0.0094 and 0.0096, *i.e.* less than 1% error.

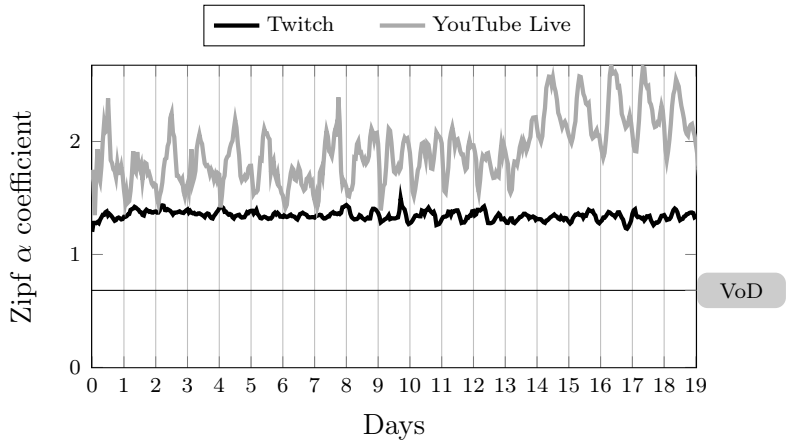


Figure 3.9: Zipf α coefficient evolution over time

Figure 3.9 shows the results obtained for the Zipf α coefficient for the first 19 days of our data set, and as well a horizontal line indicating the value

0.68198 found on classic VoD [GKMS13]. Figure 3.9 reveals an interesting point: YouTube Live channels popularity are very heterogeneous and peak hours have a big influence on it. While other UGC services ranges from 0.5 to 2, the Zipf α coefficient for YouTube Live is typically over 2, characterizing a sharper difference between channel popularity and a shorter tail. Although Twitch have also a high coefficient value (more than 1), the α parameter is constant over time. This result confirms again the popularity of Twitch system, with a larger range of channels and a more constant popularity distribution. Also, as aforementioned, broadcasters are less affected by day/night patterns on Twitch, which increases the homogeneity of the popularity distribution.

3.6 Identifying Popular Broadcasters Sessions

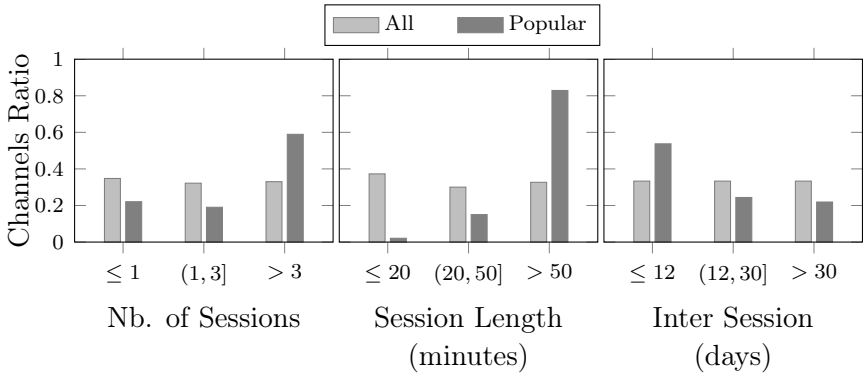
As previously explained, the most popular sessions should be identified as early as possible, if possible immediately when they start, in order to decide the delivery mean and to dimension the infrastructure (transcoding and delivery). Furthermore, the results related to the Zipf distribution of popularity indicate that the most popular channels are more popular (hundreds of thousands more) than the long tail, which puts even more pressure on identifying them early. We selected the 1% most popular channels of both services. We defined them as simply *popular*.

3.6.1 Broadcasters Characteristics

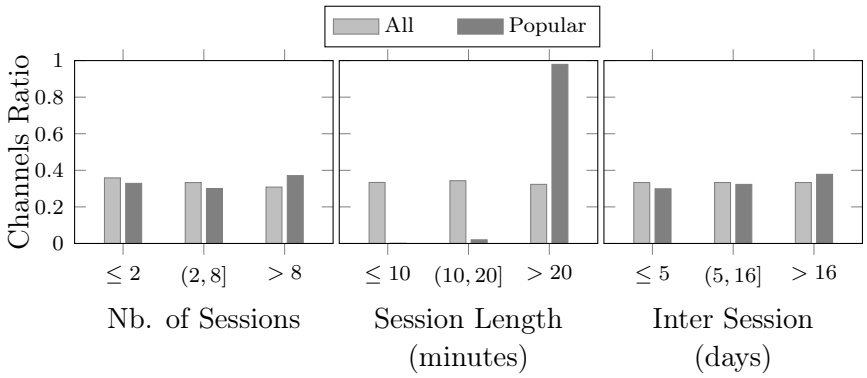
The most obvious characteristics of channels are the length of their sessions, the interval between sessions and the number of sessions that we observed during the three months. Intuitively, the most popular channels can be identified from these three characteristics. For each characteristic, we distinguish three "bins". To select the partition bins, we took the total group of channels and divided equally into three parts for each characteristic. We then applied the same limits of the total division for the popular group. The description of the characteristics and partition limits used are described at Table 3.3. The results obtained by the partition are depicted by Figure 3.10.

Characteristic	Partition Limits	
	Twitch	**YouTube Live**
nb. of sessions	$2, 8$	$1, 3$
session length (minutes)	$10, 20$	$20, 50$
inter session (days)	$5, 16$	$12, 30$

Table 3.3: Limits for each channel characteristic

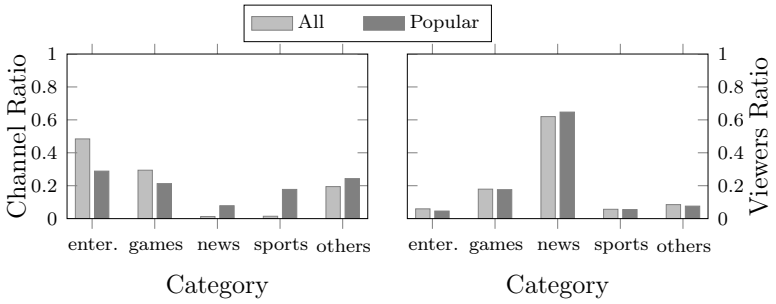


(a) YouTube Live

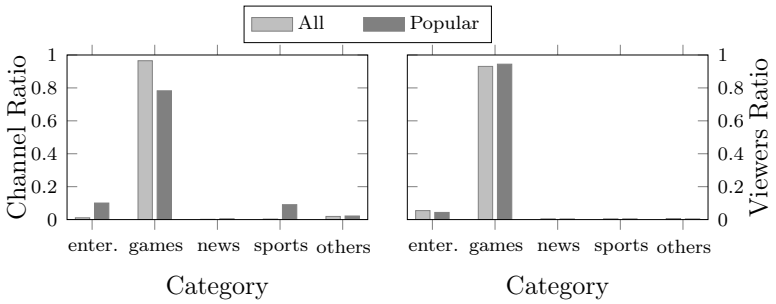


(b) Twitch

Figure 3.10: Channels ratio by characteristic partition



(a) YouTube Live



(b) Twitch

Figure 3.11: Channels Ratio by Category

The first remark in Figure 3.10 is that YouTube Live and Twitch popular channels characteristics are not identical. While YouTube Live popular channels have influence of all the characteristics, Twitch popular group remains equally distributed regarding the number of sessions and the inter-session length. Another point is that popular channels on YouTube Live can be defined as more frequent and longer than the full group of channels. Popular channels of Twitch differentiate from the full group by having longer sessions.

We also evaluate the distribution of popular channels over the most representative categories of each service. Figure 3.11 shows the distributions. The picked categories are Entertainment, Games, News, Sports, and the remaining categories grouped into Others. The categories are represented in the figure as enter., games, news, sports, others, respectively. We see that categories do not influence the popularity of both services, YouTube Live and Twitch. When comparing "All" channels with the "Popular" channels in respect to the selected categories, we noted that "All" channels and "Popular" channels have similar distributions for each category. Another remark is that both data sets are distinct in terms of partition over the same categories. YouTube Live have a good representation of entertainment and games channels, while Twitch have more than 90% of games channel. The category News on YouTube Live,

although not very representative over the number of channels, attracts a high volume of viewers to the service.

3.6.2 Video Quality and Popularity

To understand the high diversity of videos that UGC brings to live services, Figure 3.13 shows the Cumulative Distribution Function (CDF) of the bit rates of sessions for the three most popular resolutions. The key observation is the wide range of the bit rates, even for a given resolution. For example, the bit rates of 360p sources range from 200 kilobit per second (kbps) to more than 3 Mbps.

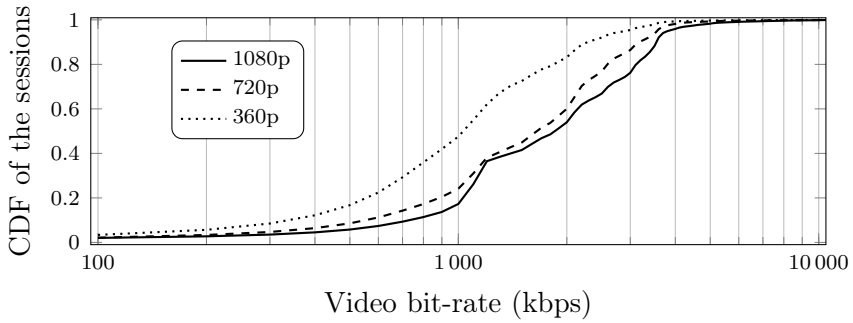


Figure 3.12: CDF of the session source bit rates

Our data set also confirms that the better is the quality of the video, the more popular it is. In Figure 3.13, we associate each range of video bit rate with a determined video resolution based on the values of the YouTube Live Stream Guide.[4] Sessions with better quality are more popular (720p being the resolution for which sessions are the most popular) although these sessions represent a small portion of the total. Typically, the sessions for videos with a resolution lesser than 720*p* quality represent 40% of the total amount of session but they attract only 8% of the total number of viewers.

We can see in Figure 3.14 the bit rates distribution between all Twitch sessions. The majority of the sessions have bit rates ranging from 500 kbps to 3 Mbps. This result confirms our first estimation of 2 Mbps as video bit rate average for live streaming services on the bandwidth calculation represented on Figure 3.3. As seen in [SdDF+12], the home broadband upload capacity is averaging 1 Mbps which we indicated in the Figure 3.14 as a vertical line. The lower half of the sessions have at most 1Mbps video bit rate. It indicates that broadcasters are limited by their broadband capacity when transmitting their live sessions. Users will not face this problem when uploading their videos to

[4]http://is.gd/BxG9hb

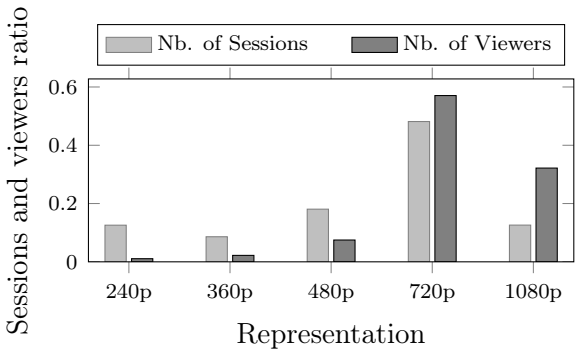


Figure 3.13: Number of sessions and viewers by video representation

VoD services, since restrict time constraints does not apply in this context. Figure 3.14 also point out a crucial information for providers: the correlation between video bit rates and broadband capacity, added to the growing of users upload capacity over the next years, surely predicts that providers will face an increasingly number of high quality videos, thus more bandwidth demanding sessions. Therefore, studies about large scale live video streaming services, like the one presented in this thesis, are important in order to make future solutions to cope with the growing user demands.

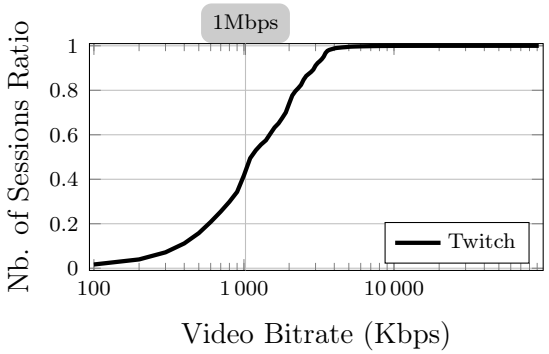


Figure 3.14: CDF of the session video bit rates

3.7 Conclusion

To the best of our knowledge this data set of Twitch and YouTube Live is the first attempt to understand the behavior of UGC live streaming videos systems with comparison of two largely used systems. We explore in our study

useful insights on what can be expected of a massive UGC live streaming video service. We show that 1 Tbps of bandwidth is commonly achieved in order to deliver the contents of such system. We also point out differences between Twitch and YouTube Live, especially regarding the broadcaster behavior. Broadcasters on Twitch are clearly more engaged on producing live streams than the ones found in YouTube Live. We detected many characteristics that indicates the strong engagement of Twitch broadcasters. Compared to VoD service, live streaming video services broadcasters are more active. Aside from other systems, the popularity found on live is highly heterogeneous.

We use our findings and real data sets information in the following chapters, as the design foundation of our solutions and as input to our simulations, which provides a close to real simulation scenario. By offering this analysis and making the data set public available we hope to enrich the community and help to improve live streaming systems.

Chapter 4

Mapping Sessions to Servers

Contents

4.1 Introduction

Live video streaming systems, such as Twitch or YouTube Live, offer to users the possibility to broadcast or to watch live video sessions. The amount of data to be processed and delivered to viewers is considerable. For example, every minute, Twitch platform must absorb more than 30-hour live video sessions created by thousands of broadcasters at 2014. Sessions must be delivered to hundreds of thousands of viewers [twi].

The main difficulty in live video streaming delivery systems comes from: (i) the scale, both in terms of video sessions and in terms of number of viewers; and (ii) the live sessions diffusion is different from VoD: it is not possible to replicate video files *in advance* on multiple servers, there is no video file before the start of a session. It also differs from IPTV: compared to television, the number of different sessions is greater and the number of users per session is extremely variable, with some sessions having very few viewers. For example, 88% of Twitch sessions, over three months, have less than 10 viewers.

It is thus necessary to dynamically adjust the number of sessions of delivery servers to meet the demand. When a session becomes very popular, it is possible that the server which distributes it reaches its maximum capacity, in

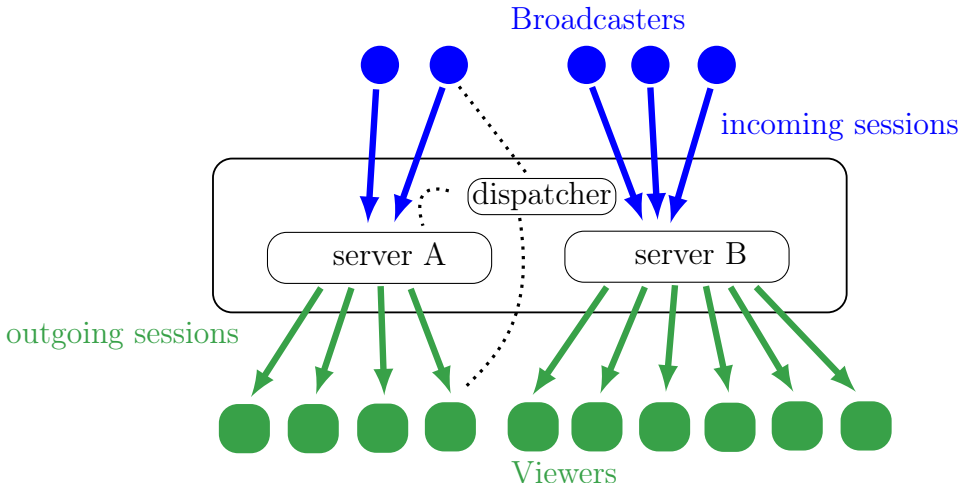


Figure 4.1: Live session overview.

that case it must share the load with another server. This induces an additional cost in terms of bandwidth consumption between servers. To minimize this overhead, it is necessary to smartly map the video sessions on servers. In this chapter, we describe POPS, a popularity-aware live streaming service. This approach is based on the study of real traces from Twitch and YouTube Live platforms, described in Chapter 3. It allows us to predict the popularity of a video session. POPS is based on these predictions to dynamically provision delivery servers. Our evaluation shows that when a history can be used to estimate the popularity of a session, POPS can decrease the amount of bandwidth used among servers without using too many servers.

The remainder of this chapter is organized as follows. The following Section 4.2 defines the model considered for the live streaming delivery. Section 4.3 presents our contribution: a placement strategy using popularity predictions. Finally, Section 4.4 describes our evaluation before Section 4.5 that concludes the chapter.

4.2 Model

We focus on platforms broadcasting UGC live video. Remind from Section 2.2 that users of these services can play both roles: some create video sessions, they are called *broadcasters*; some watch live sessions, they are called *viewers*. At a given time, a user can play either one of the two roles, or both simultaneously. However, in our study, we focus on each role separately. It is also possible for a user to watch multiple sessions simultaneously, even if it is not norm.

In our model, illustrated by Figure 4.1, when a user creates a video session, he sends it to the service provider. We consider that the service provider architecture is composed by a set of servers in a data center or cloud. We discussed this architecture in Chapter 2. The number of involved servers may evolve dynamically over time. Servers can be virtual machines in a *cloud* and be provisioned according to the load. The servers can be in a same data center but they can also be geographically distributed, in different data centers or located at the edge of the network. We do not consider limits on the maximum number of servers used by the service provider, and we neglect the time it takes to allocate a new server (servers can be pre-allocated, by pool). Each session is assigned to a server, which is responsible for delivering it to the set of viewers. A server can be assigned to deliver multiple sessions. It is also possible that a session is supported by more than one server, when its popularity (number of interested viewers) exceeds its server's outgoing bandwidth capacity.

We consider a *dispatcher* service, which can be either centralized or distributed. This service receives requests: (i) to deliver a new session, from a broadcaster; (ii) to subscribe to a session, from a viewer. This service is responsible for assigning a server to deliver a new session, it may either choose an already existing one that has outgoing bandwidth capacity left, or allocate a new one. Then, the broadcaster uploads its session content directly to the assigned server. While receiving a request from a viewer, the dispatcher service identifies the delivering server in charge of its diffusion and returns it to the user. The server then sends the session directly to the viewer.

We let the design of the dispatcher service architecture for future works. We focus here on the study of placement choices that such a service has to do and their consequences on the number of needed servers and on the amount of inter-server bandwidth consumed.

The number of clients that can be served by one server depends on the server capacity (in terms of outgoing bandwidth) and on the served sessions bit rate. A server can therefore broadcast to a certain number of destinations given by Equation 4.1.

$$server\ destinations = \frac{server\ outgoing\ bandwidth}{\sum_{i=1}^{\#viewers} bandwidth\ needed\ for\ viewer_i} \tag{4.1}$$

A destination can be either a viewer, or another server when a session has to be served by multiple servers (see Section 4.3). For each session it delivers, a server preserves enough bandwidth to be able to send one copy of the session to another server in the case new viewers arrive and it does

not have the capacity to deliver them. We do not consider the incoming bandwidth limitation: in the systems we observed, Twitch and YouTube Live, around two orders of magnitude more viewers than sessions. That means that the scalability problem is at the level of the outgoing bandwidth, to serve a high number of viewers. We thus consider that the incoming bandwidth is always sufficient.

The scales are important in our problem. As we discussed in Chapter 3 the number of users that these platforms must be able to handle at a given time is of the order of several hundreds of thousands. Over one million different broadcasters have been observed on Twitch and more than hundreds of thousands on YouTube Live. There are always many video sessions (thousands), but also many viewers (hundreds of thousands spread over the different sessions). And the distribution of these viewers among the sessions is very heterogeneous: a few units to thousands and even hundreds of thousands for some of the most popular sessions.

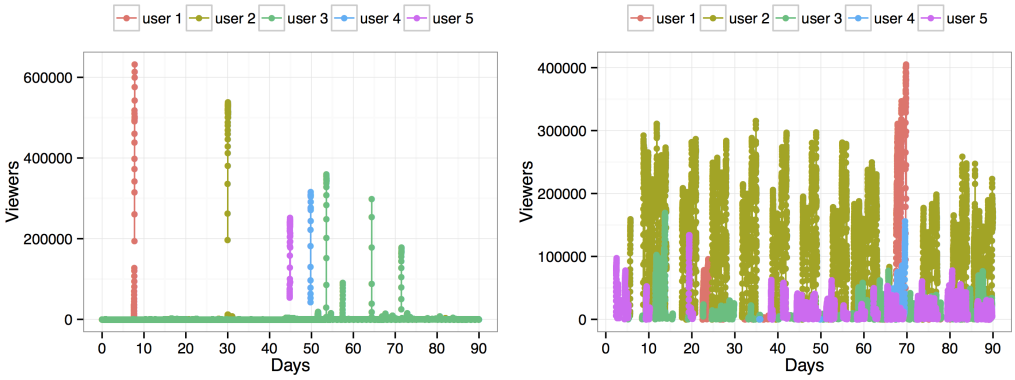
4.3 Mapping live video sessions on broadcasting servers

This section describes our contribution. We begin by briefly presenting the lessons we have learned from the collection of Twitch and YouTube Live user traces. We study different placement approaches to map video sessions on delivering servers. We present our approach based on session popularity prediction.

Twitch and YouTube Live are user-generated live video streaming platforms. Twitch is very popular: cumulative over three months, there have been more than five million sessions, generated by more than a million broadcasters. Everyday, several hundreds of thousands of viewers are simultaneous using the services. Furthermore, both platforms offers APIs providing the ability to periodically collect sessions and users information.

A complete analysis of the collected data is presented in Chapter 3. The main points, learned from the data, used in this chapter are:

Among video sessions, the popularity is highly heterogeneous. The biggest part of the traffic being generated by the few most popular ones, Figure 4.2 presents five different users and the heterogeneity between their sessions. As previously discussed, we have shown that in these systems, Twitch and YouTube Live, the Zipf coefficient is big and implies a high popularity heterogeneity among all sessions.



(a) YouTube Live (b) Twitch

Figure 4.2: Broadcaster sessions over the days of the five users with biggest peak viewers

The global system load varies a lot during time. Remind the bandwidth consumption estimated for both systems in Chapter 3 have huge variation. Both systems have peaks of 1 Tbps, while the valleys of usage are less than half of this value in Twitch and less than a hundred Gbps in YouTube Live.

There are always many video sessions. Orders of magnitude higher than the number of channels offered by a classical cable TV. Specially on Twitch, which in average has around 6,000 channels online.

Users behavior is predictable. It is possible to observe a night and day oscillation, but also that a session popularity is highly correlated to the broadcaster that creates it, and it tends to grow with the session length. That means that, popularity among broadcasters is highly heterogeneous, but the sessions of a same broadcaster have a similar peak popularity.

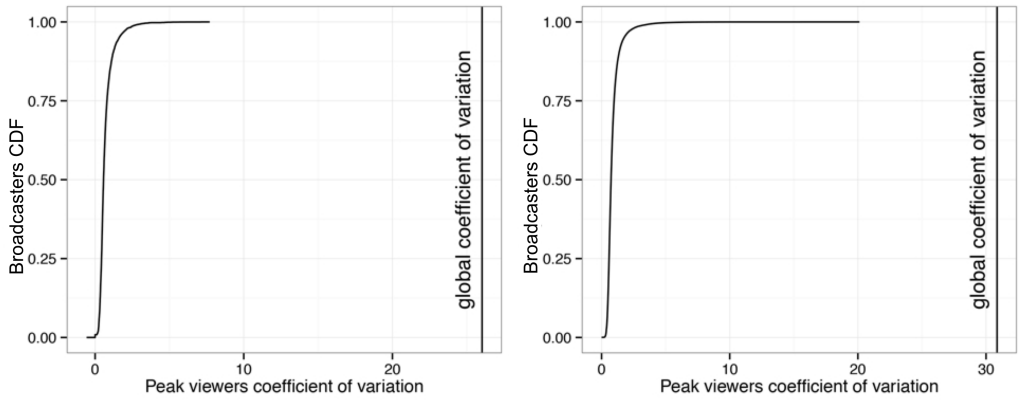
4.3.1 Popularity predictability discussion

The popularity prediction in live streaming services is a key aspect for resource provisioning but it is far from being trivial. We extracted from the data that the users (the broadcasters) popularity magnitude is predictable but not the precise popularity (the number of viewers the future sessions will gather).

It is not trivial task since among all the broadcasters there are extremely different behaviors. To illustrate this, Figure 4.2 shows the popularity evolution over time for five selected users from both services. We select the

five users that reached the highest peak of viewers over the collected period. Notice that, between the two services (YouTube Live and Twitch) there is already an important difference: on Twitch popular broadcasters broadcast more frequently than on YouTube Live. The popularity differences between broadcasters of a same service provider are clear. Taking as an example users 2 and 5 on Twitch, user 2 is about five times more popular than user 5.

However, sessions that are made by a same broadcaster have a tendency to have similar popularity. While taking a closer look at users 2 and 5 of Twitch, it is easy to identify such trend among each broadcaster's sessions. User 2 popularity stays around 250 000 viewers and user 5 around 80 000. To present such a tendency we calculate the popularity coefficients of variation presented by Figure 4.3. The coefficient of variation is defined as the ratio of the standard deviation to the average value. The coefficient of variation among all sessions is represented by a vertical line, while the coefficient of variation for sessions of each broadcaster are presented by the cumulative distribution curve. For simplicity, we only take into account broadcasters with more than 12 sessions, however cumulative distribution curves made with other minimum numbers of sessions were similar. As previously discussed, the global variation among all sessions is much higher than the popularity variation for sessions of a same broadcaster. This result shows the importance to take into account each broadcasters information to provide better popularity prediction for live content. However, predicting the precise number of viewers a session will gather is beyond our study reach, preventing us to be able to provision the exact amount of needed resources.



(a) YouTube Live (b) Twitch

Figure 4.3: Broadcasters CDF, with at least 12 sessions, of peak viewers coefficient of variation.

4.3.2 Number of servers versus bandwidth usage trade-off

In our model, a server can only serve a limited number of viewers depending on its bandwidth capacity and the bit rate of the served sessions. A naive approach consists in assigning new sessions to a server until it reaches its maximum capacity. However, video sessions popularity varies upon time. An already loaded server can thus acquire new viewers for sessions it already broadcasts, it may then exceed its capacity. In order to keep offering an acceptable quality of service, it is necessary to use a secondary server, either a new one, or one already serving other sessions (but with capacity left). The original server still has to broadcast the session to the viewers it already served, but it also has to send it to the secondary server, using a preserved bandwidth dedicated to this use. This session *replication* mechanism is illustrated by Figure 4.4, indicated by the curved arrow from Server S to Server T. Even if all the viewers can have an acceptable quality of service using this mechanism, it has a non negligible cost in terms of bandwidth. We can measure this extra cost by summing the quantity of data transferred across servers for session replication, in bytes transfered.

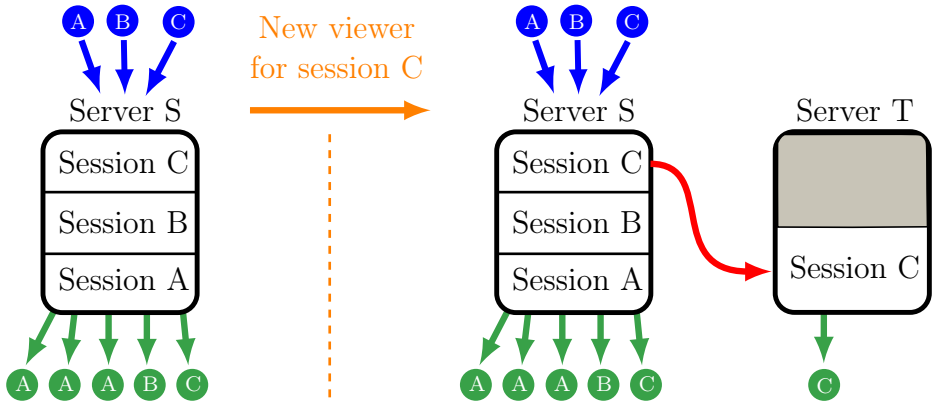


Figure 4.4: Server S has reached its maximal load, a new viewer for session C implies a transfer on a new server (T).

As the number of viewers for a session varies upon time (either increasing or decreasing), it is possible that the phenomenon described above leads to configurations for which many sessions have to be transferred among servers. In this thesis, we do not take into account the possibility to reconfigure the mapping: once a viewer is mapped on a server for a given session, it will

always receive this session from this server. A reconfiguration would consist in migrating viewers among servers, which will be considered in future works.

The simplest manner to limit the necessity to transfer/replicate sessions among servers is to over-provision, to provide a *margin* as illustrated by Figure 4.5. It allows the server to accept new viewers for the sessions it already broadcasts, diminishing the risk to replicate a session to a secondary server. The extra bandwidths provisioned for different sessions on a same server are merged: at the start of a session, extra bandwidth is provisioned, however, this extra bandwidth can be used to serve any session broadcasted by the server. Indeed, *the goal is to avoid a much as possible to replicate a session on a secondary server*. If the margin provisioned for a very popular session is already used and there is unused bandwidth capacity available on the server (reserved for other sessions margin), it would be a bad idea not to use it for the very popular session if it needs it (the other sessions on the server may not need it).

Provisioning extra bandwidth reduces the number of sessions that can be served by one server, it thus implies to allocate more servers. The trade-off between the number of servers used and the inter-server bandwidth extra-cost can be tune through the margin value. Other parameters are also important, the server choice policy made by the dispatcher, *bestfit*, *firstfit* or *worstfit*, can have an impact on the number of servers and on the inter-server bandwidth consumption.

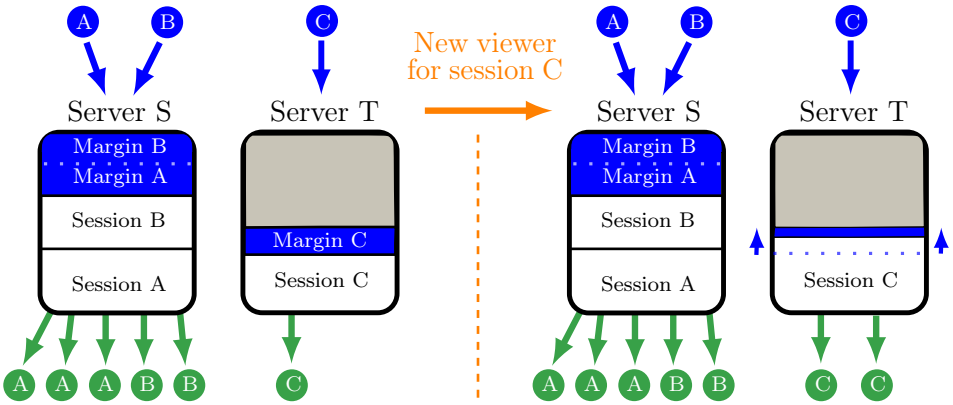


Figure 4.5: The server S has reached its maximal load (taking margins into account). It will not accept to serve a new session, however, a new client for session A or session B can be served using part of the provisioned margins.

4.3.3 Taking video sessions popularity into account

The value of the provisioned margin has an important impact on resources consumption. If it is too low, many sessions have to be served by multiple servers, consuming a lot of inter-server bandwidth, as argued above. On the opposite, if it is too high, the system involve many servers using only a small part of their capacity. Therefore it is important, for each session, to set a margin close to what is effectively needed.

Our approach, *POPS*, is based on the use of an estimator. It gives an estimation of the future popularity of a session which is used to set the margin. The goal is to have a margin large enough to diminish the risk to have to transfer the session to another server during popularity peaks, but not too large to limit resource (server) waste. This is based on the finding that the popularity of the video sessions broadcasted by a same broadcaster does not vary a lot. When a known broadcaster (having already at least one previous uploaded session in the system's history) uploads a new video session, our approach aims at provisioning enough resources to support its estimated viewer peak. We consider four different estimators:

Replication is the core solution. It defines a fixed margin for each session related to the maximum resource on the server. For example if margin is defined as 10%, one server can deliver 10 different sessions.

POPS estimates that a video session will have the same popularity as the previous video session uploaded by the same broadcaster.

A-POPS (for Adaptive POPS) has been designed to take into account broadcasters having a growing popularity. If the history contains at least two video sessions from one broadcaster, *A-POPS* expects the popularity of this broadcaster to continue to grow exponentially and computes the viewer peak of the next peak accordingly. If it can not estimate the growth (e.g., missing historical data), *A-POPS* adds an arbitrary 10% margin to the last popularity peak of the same broadcaster.

Oracle is the estimator that could be built if it was possible look in the future. It knows in advance the number of viewers that a session will gather, it can provision exactly the amount of resources that will be needed.

To illustrate each approach decision, we formulate the figurative scenario given by Table 4.1 and Figure 4.6 with two sessions. In this example we define the maximal broadcasting capacity (MAX_SERV) as 2 Gbps, to be distributed between the sessions needs. In Table 4.1 we have the history of

the sessions and the used capacity in the record. The broadcaster responsible for session B had no previous session. Figure 4.6 shows the reserved capacity calculation (margin) for each approach. Illustrated with dots is the amount of reserved capacity that was not used by the sessions. The hatch lines sections indicates the capacity needed but not reserved by the approaches. *Replication* is set to 50% of maximal capacity the server, which yields to 1 Gbps for each session. *POPS* assumes that this session popularity will be the same as the previous one. For session A it sets the margin to 0.4 Gbps. For session B without history it sets to 10% of server capacity, 0.2 Gbps. *A-POPS* adds the growth to the forecasting of session A, setting to 0.4 Gbps with the 10% growth added. For session B *A-POPS* acts just like *POPS*, setting 10% of server capacity, 0.2 Gbps. For *Oracle* sessions A and B are set to their respective peak popularities, 0.8 Gbps and 0.6 Gbps.

Session	Previous Popularity	Previous Growth	Peak Popularity
A	0.4 Gbps	10%	0.8 Gbps
B	No previous stream		0.6 Gbps

Table 4.1: Figurative sessions information used on margin calculation example.

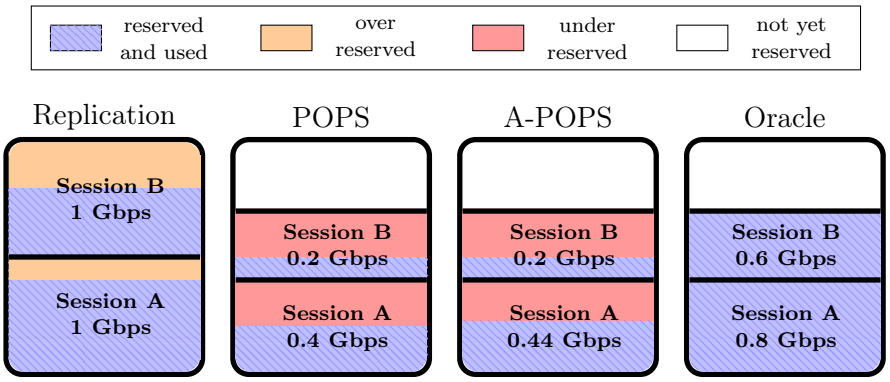


Figure 4.6: Figurative example of margin calculation for each approach. Illustrated in plain gray color and numerically indicated is the capacity reserved. The dotted sections are the amount of reserved capacity that were not used by the sessions. The hatch lines sections indicate the capacity needed but not reserved by the approaches.

It is important to notice that even with a perfect oracle, this solution does not give the optimum: it could be possible to serve a short session outside the peak load period using a portion of the provisioned margin. To do so, it would be necessary to be able to predict sessions arrival times, sessions lengths and

the evolution of the number of viewers during the session life. However, while predicting an approximate popularity peak for a session is doable (because the popularity linked with the broadcaster of the session), predicting when the peak of viewers will occur in the session life and for how long remains a very difficult task and beyond this thesis scope.

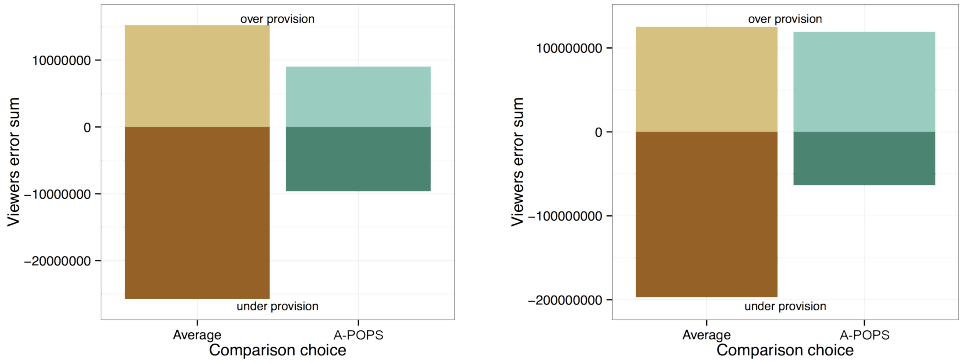
4.4 Evaluation

As discussed in Section 4.3 we have identified some trends on the broadcaster popularity. Our evaluation starts by showing how much taking the broadcaster popularity into account is important. To do so, we compared two alternatives for the broadcasters popularity prediction: (i) a very naive approach using the global average number of viewers per session (called *average*), and (ii) an approach that assumes that the next sessions of the same broadcaster will have a similar popularity then the previous ones, based on A-POPS (called *history*).

Figure 4.7 presents the comparison between the two approaches. We called *viewer error* the difference between the prediction and the real values (the real number of viewers). For the *average* approach, the prediction values are the services averages which are 81 for YouTube Live and 21 for Twitch. For the *A-POPS* the prediction for one session is calculated based on the previous sessions from the same broadcaster. We consider that the popularity for one session will be the same as the previous one times the increase (if any) between the two previous sessions. We present in the figure the sum of the viewer errors over the period. The positive values represents the over- provisioning of both predictions and the negative values the viewers not served. This figure confirms that a simplistic approach, such as using a global average, does not fit live popularity prediction and that an broadcaster history can indeed improve predictions and therefore the delivery provisioning.

Parameters	Values
Traces	Twitch, YouTube Live
MAX_SERV	2 Gbps
Allocation Policy	FirstFit, BestFit, WorstFit
Approaches	Replication, POPS, A-POPS, Oracle
Stream Margin	0%,10%,...,50% of MAX_SERV

Table 4.2: Simulation parameters



(a) YouTube Live (b) Twitch

Figure 4.7: Sum of the viewers provision errors, comparison between global average viewers and A-POPS

To further evaluate the approaches we set up several parameters summarized by Table 4.2. We used the traces collected in the Twitch and the YouTube Live platforms. The maximal broadcasting capacity MAX_SERV of each server has been set to 2 Gbps, which is a conservative value considering nowadays network cards. This limit is 2 Gbps of outgoing bandwidth for each server where for each session we consider the bit rate as 2 Mbps. We calculate the server cost in $servers - h$: one server used one hour counts for 1 $server - h$ and the network cost in bytes transfered between servers. An example of metrics calculation is given by Figure 4.8. In this example, there are two servers running for one hour. There are three sessions being served: A, B and C. The two servers running for one hour accounts for 2 $servers - h$. Session C has 2 Mbps bit rate. The replication of session C from server S to server T accounts for 900 megabyte (MB) transfered data.

To choose among multiple servers to which one a session will be given, we have evaluated the three well known allocation policies: FirstFit, BestFit and WorstFit. Concerning server provisioning, we have evaluated four approaches: *Replication*, *POPS*, *A-POPS*, and *Oracle*. *Replication* consists in the basic strategy using a *fixed* margin for each session instead of a popularity estimator, illustrated by Figures 4.4 and 4.5: when the bandwidth needed for a session (viewers × bit rate) exceeds the fix margin, the server forwards the session to another server. *POPS* is our approach in which the margin is estimated considering that broadcasters have a constant popularity. *A-POPS* takes into account broadcasters popularity growth as described in Section 4.3. Finally, *Oracle* gives the results *POPS* could have if the prediction was perfect (for

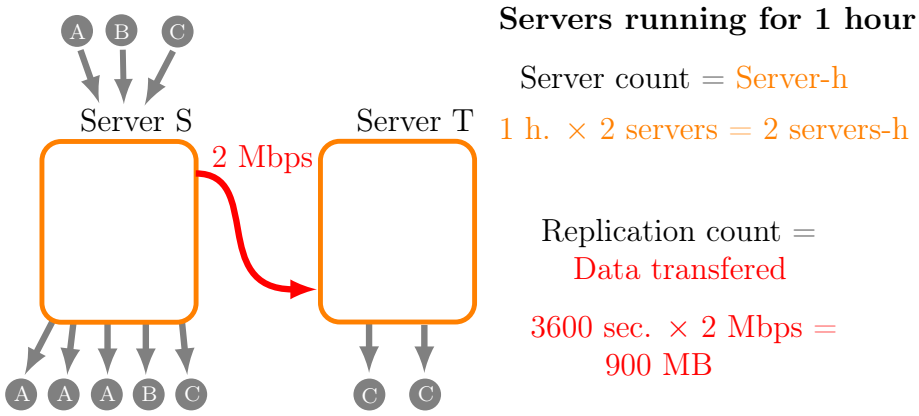


Figure 4.8: Example of our evaluation metrics calculation on servers running for one hour.

each session, we provision exactly the needed resources to face the viewer highest peak). This is possible because we know in advance the trace used to fit our simulator.

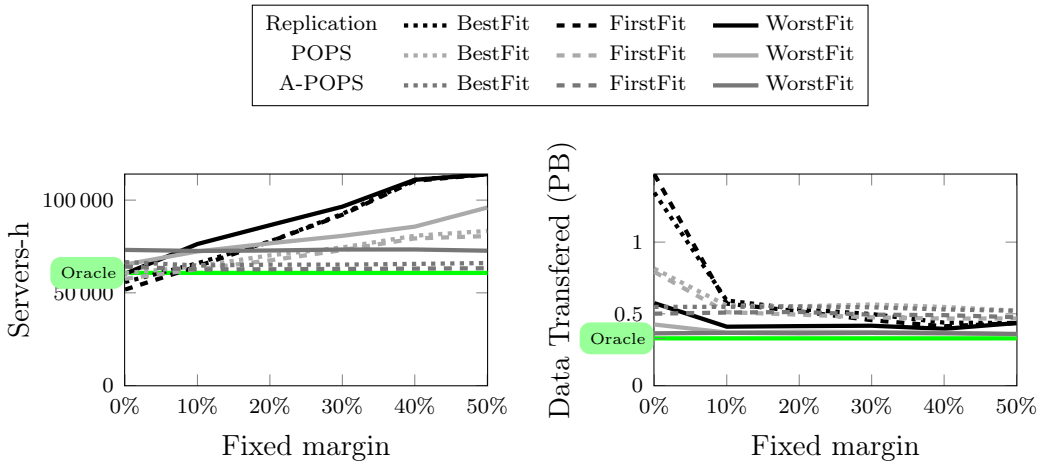
We vary the fixed margin from 0% to 50% of the total server capacity. Those fixed margins are used for *Replication*, *POPS*, and *A-POPS* approaches. In the case of *POPS* and *A-POPS* the fix margins are used when no prediction can be done, in absence of broadcaster history. As our simulation started with an empty history, a fixed margin is used for the first sessions of each broadcaster. For the next sessions *POPS* and *A-POPS* will estimate the margin. The traces we use are limited in time (two weeks). More the historical data covers a long period, more *POPS* and *A-POPS* will be able to use an estimated margin.

We compare all the strategies in Figure 4.9. For each one, we compute the server cost ($servers - h$) and the inter-server bandwidth cost (bytes transfered). An horizontal line shows the *Oracle* strategy cost.

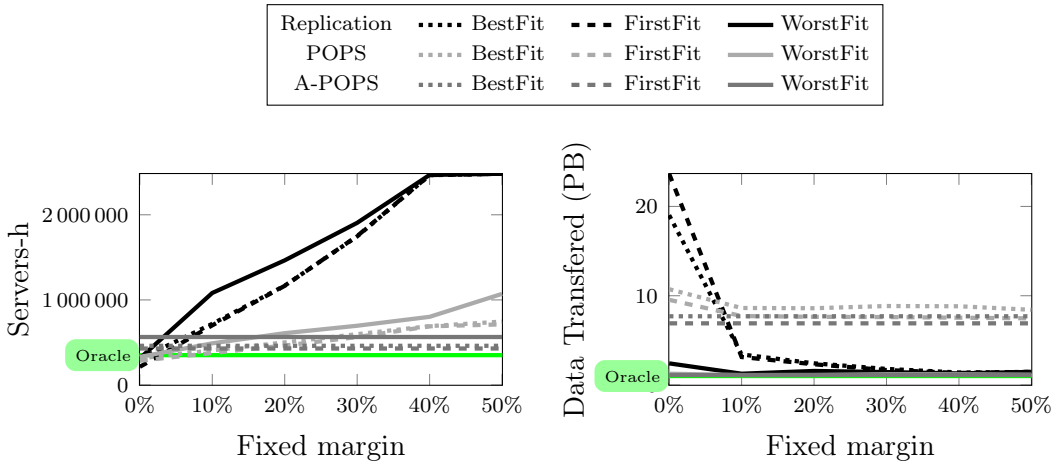
The first result of Figure 4.9 is that the allocation policy (FirstFit, BestFit or WorstFit) has a great impact. For the *Oracle* strategy, we only show the best of the three policies, the WorstFit one, for legibility reasons. The BestFit policy, as expected, has the lowest server cost. The FirstFit policy has a similar performance. However, these two strategies lead to many inter-server transfers.

Another observation is that increasing the fixed margin induces a proportional increase of the number of broadcasting servers used. However, in terms of bandwidth, above 20% of margin, the gain becomes negligible.

While comparing the results obtained with the different traces (Twitch and YouTube Live) we see that they are significantly different. This stresses the importance to work with multiple workloads. We can observe that the



(a) YouTube Live



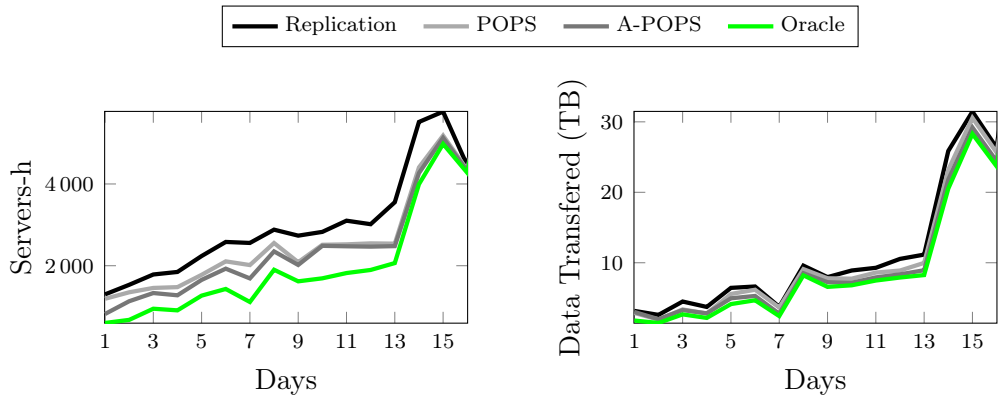
(b) Twitch

Figure 4.9: Comparing approaches.

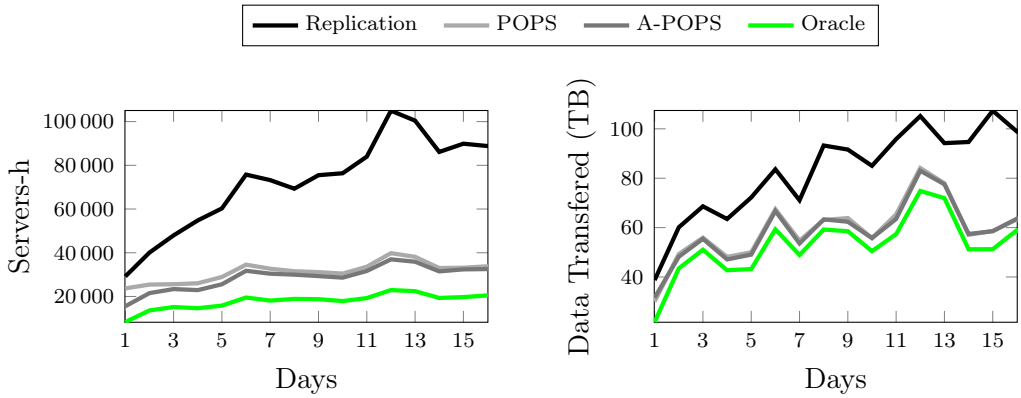
gain bring by *POPS* in terms of server cost is more important in the Twitch case. This is due to the fact that Twitch broadcasters tend to produce more frequent sessions while in the YouTube Live case the inter-broadcast delay of a given broadcaster is greater. In the case of Twitch, the proportion of sessions for which the system already knows the broadcaster is thus greater, allowing *POPS* to use more often its estimated margin instead of the fixed one.

However, we have detected that some Twitch broadcasters have an increasing popularity. Which means that their consecutive sessions are more and more popular. *POPS* does not add an *extra* margin, thus, it is not tai-

lored for such an increasing popularity. It just considers that a session will have the same popularity as the last session from the same broadcaster. This affects the bandwidth (bytes transfered) for Twitch's FirstFit and BestFit policies. *A-POPS*, which takes into account the popularity growth, provides the ability to decrease the amount of bandwidth used (specially in the case of Twitch) while having a cost close to the Oracle one in terms of servers, as illustrated by Figure 4.9.



(a) YouTube Live



(b) Twitch

Figure 4.10: Approaches behavior upon time.

To better understand the first results, we studied the evolution of the system with the four approaches. Figure 4.10 presents the evolution during time. For each approach we selected their best set of parameters regarding bandwidth cost (WorstFit allocation with 20% margin). As expected, *POPS* and *A-POPS* efficiency improves along time, as the history becomes bigger, allowing them to use an estimation instead of a fixed margin like with the *Replication* strategy. In a real system, most *broadcasters* are already known.

Users broadcasting a video session for the first time are rarely popular. However, our collected traces start at an arbitrary time in the system's life.

The benefits of *POPS* are significant and the results are similar to the ones obtain with the *Oracle* strategy on both traces. The *Replication* strategy performs poorly on Twitch traces, where we have a large number of broadcasters. The divergence between *POPS* and the *Replication* strategy, concerning number of servers cost, is due to the growing number of sessions coming from already known broadcasters; which is more important in the Twitch case than in the YouTube Live one. *A-POPS* offers slightly better performances compared to *POPS*, thanks to the fact that it takes into account the broadcasters popularity growth.

Improvements can still to be done in future works. For instance, it should be interesting to take into account an estimation of sessions length. A study of growing popularity curves could also help the placement strategy by allocating small sessions aside of longer sessions that grows slowly.

4.5 Conclusion

The popularization of both networks and video capture devices has lead to the birth of user-generated content live streaming platforms. These platforms have to face a high number of simultaneous video sessions that they have to collect and distribute to an even higher number of viewers. In this chapter, we show that taking into account the future popularity of incoming sessions is important while mapping them to a set of servers. This may help to prevent session partition across multiple servers, inefficient in terms of inter-server network cost. We base our study on real data sets collected on the YouTube Live and the Twitch platforms. We study the number of servers versus inter-server bandwidth usage trade-off. Our new approach, *POPS* uses popularity peak estimation of broadcasters while placing new sessions on broadcasting servers.

In this work we did not consider the possibility to migrate viewers across servers. At the end of viewer peaks, it should be interesting to migrate viewers and sessions in order to be able to shutdown servers. This is part of our ongoing work. Furthermore, we plan to use enhanced mechanisms (*e.g.* learning) to try to predict the popularity evolution during the life of a session.

Chapter 5

Mixing Data Center and CDN for Delivery

Contents

5.1 Introduction

Major video streaming services have different architectures of their delivery structure. As discussed in Chapter 2 the usage of multiple data centers, composition of multiple CDNs, P2P assisted by CDN and data center assisted by CDN are some of the architectures chosen by real services. Although there are multiple options for delivery of the live video content, no formal definition considering such differences have been made so far. Specific cases of studies, specially evolving P2P and CDN were explored [Pas12] and we previously presented a discussion of the differences between the different delivery infrastructures. Also we explored a private cloud delivery in Chapter 4. However, no evaluation comparing different delivery compositions was made, neither hybrid solutions were explored.

In this chapter our contributions are therefore two folded. Our first contribution is a formal presentation of the multiple delivery problem for live streaming videos. With this formalization we are able to fully understand the problem and retrieve optimal solution for any given scenario. The second contribution is an analysis with experiments using our traces of Twitch

and YouTube Live, presented in Chapter 3, which reveals the big gap between the optimal solution we found with the formalization and four intuitive algorithms.

In this chapter we first introduce basic definitions and our model for the hybrid delivery of live streaming videos in Section 5.2. Then, in Section 5.3, we formalize the multiple delivery problem. Section 5.4 motivates a hybrid delivery approach. Next, in Section 5.5, we evaluate the model with experiments using real traces of two different live streaming services, and Section 5.6 concludes this chapter.

5.2 Model for Hybrid Delivery

Our objective is to explore the different options for live streaming delivery systems. We consider, in the case of our hybrid delivery model, that the system objective is to deliver live streaming video content produced by broadcasters to a population of viewers.

Regarding the hybrid compositions of the delivery system for live streaming videos, we study here four common types, CDN-P2P, Multi-CDN, Multi-DC and DC-CDN, as described in Chapter 2. Therefore, in our model we consider multiple components, including the Content Provider (CP). The CP initially absorbs the live streaming video data to be delivered to viewers. The CP can be assisted by other components to achieve the total distribution required by the viewers. The assisting components can be either the viewers themselves, over a P2P based infrastructure; either DCs, an infrastructure owned or pre-paid (therefore no extra cost) by the CP but it has limited boundaries regarding the distribution capacity; either CDNs, an infrastructure that can be paid accordingly to the consumption of resources but with higher costs than DCs.

5.3 Theoretical Optimization Problem

To initially understand the problem of delivering live streaming videos, we decided to define it as a costs minimization linear problem. In this formalization we aim to include the previous cited types of compositions and components. As we are here interested in the comparison of the different compositions we defined P2P as well as a component with variable upload capacity, we do not target specifics of P2P systems. Detailed analysis of specific P2P compositions can be found in [YLZ+09b]. The notations used in the problem are listed in Table 5.1. The variables β and γ are used in the linear problem. Equation (5.1) defines β_{vit}, which represents a component i delivering video v

at time t. Equation (5.2) defines γ_{vit}, which indicates that a video v starts to be delivered by component i at time t. The linear problem itself is depicted by equations (5.3a) to (5.3k).

Notations	
α_{vit}	Number of users watching video v on component i at time t
P_{vt}	User Population at time t for video v
V	Live videos set
b_{vt}	Bit rate of video v in *kbps* at time t
I	Components of the system
r_i	1 if component i is P2P, 0 otherwise
e_i	P2P efficiency of component i (kbps per user, P2P components only)
u_i	Upload capacity of the component i (non-P2P components only)
c_i	Cost in $ per *kbps* of the component i
s_i	Cost in $ for setting up a video into component i
δ_{vt}	Number of components serving video v at time t
d	Penalty cost in $ for using multiple components for a video
$t \in [1, T]$	Discrete time index t
T	Total time of the problem

Table 5.1: Notations

$$\beta_{vit} = \begin{cases} 1 & \text{if video } v \text{ is served by } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

$$\gamma_{vit} = \begin{cases} 1 & \text{if } i \text{ starts serving video } v \text{ at time } t \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

$$\min_{\{\alpha,\beta\}} \sum_{t\in[1,T]} \sum_{v\in V} (\sum_{i\in I} \alpha_{vit} \cdot b_{vt} \cdot c_i + \gamma_{vit} \cdot s_i) + \delta_{vt} \cdot d \tag{5.3a}$$

$$\text{s.t.} \sum_{v\in V} \alpha_{vit} \cdot b_v \cdot (1 - r_i) \leq u_i \qquad i \in I, t \in T \tag{5.3b}$$

$$\sum_{i\in I} \alpha_{vit} \cdot b_v \cdot r_i \leq P_{vt} \cdot e_i \qquad v \in V, t \in T \tag{5.3c}$$

$$\sum_{i\in I} \alpha_{vit} = P_{vt} \qquad v \in V, t \in T \tag{5.3d}$$

$$\beta_{vit} = \begin{cases} 1, & \text{if } \alpha_{vit} > 0 \\ 0, & \text{otherwise} \end{cases} \qquad v \in V, i \in I, t \in T \tag{5.3e}$$

$$\gamma_{vit} = \begin{cases} 1, & \text{if } \beta_{vit} = 1\& \\ & \beta_{vit-1} = 0 \\ 0, & \text{otherwise} \end{cases} \qquad v \in V, i \in I, t \in T \tag{5.3f}$$

$$\delta_{vt} = \sum_{i\in I} \beta_{vit} \qquad v \in V, t \in T \tag{5.3g}$$

$$\alpha_{vit} \in \mathbb{R}^+ \qquad v \in V, i \in I, t \in T \tag{5.3h}$$

$$\beta_{vit} \in [0, 1] \qquad v \in V, i \in I, t \in T \tag{5.3i}$$

$$\gamma_{vit} \in [0, 1] \qquad v \in V, i \in I, t \in T \tag{5.3j}$$

$$\delta_{vt} \in \mathbb{R}^+ \qquad v \in V, t \in T \tag{5.3k}$$

The objective function (5.3a) minimizes the overall costs, which is the sum of costs per *kbps* delivered, costs of setting up a video into a new component and the penalty cost for using multiple components to deliver the same video. The constraint (5.3b) set up the maximum capacity of the component i (Non-P2P). The constraint (5.3c) set up the maximum capacity of the component i (P2P). The constraint (5.3d) set up the user video population. The constraint (5.3e) set up a video v served by a component i. The constraint (5.3f) set up a video v started to be served by a component i. The constraint (5.3g) set up the penalty for v served by multiple components. The constraints (5.3h) to (5.3k) define the limits of the variables.

5.4 Motivations for Hybrid Delivery

We recall a selection of insights, discussed at Chapter 3, that endorses the usage of a hybrid DC-CDN delivery model for UGC live streaming systems. First, a small number of contributors of UGC systems represents the vast majority of the global popularity of these platforms, typically following the Zipf's

		Twitch Top			YouTube Live Top		
	days range	10	20	50	10	20	50
# channels	1 - 30	487	1055	2425	557	1065	2923
	31 - 60	518	1032	2426	515	1143	3275
	61 - 90	462	956	2411	586	1197	3091
	1 - 90	975 (0.06%)	2013 (0.13%)	4880 (0.31%)	1167 (0.4%)	2345 (0.9%)	6409 (2.6%)

Table 5.2: Number of channels for top categories and percentage from total number of channels. Total channels on Twitch = $1,570,844$. Total channels on YouTube Live = $248,563$.

law. Such a distribution simplifies the management of CDN infrastructures. The provider is also interested in delegating the channels with the highest resolution to the CDN, which can throttle the limited bandwidth capacity of DCs. Finally, channels that are stable over time are easier to manage in CDN, with less configuration of edge-servers.

Most of the traffic comes from a small proportion of broadcasters.

As discussed at our data set analysis, the popularity of live streaming services follows a Zipf's law. Few broadcasters have the majority of the global system popularity and are, consequently, responsible for the most part of delivery traffic.

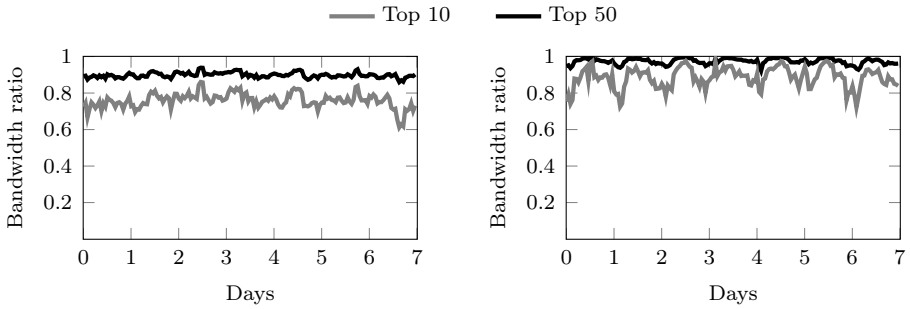
We expand our previous analysis by collecting the k most popular channels. We focus on values of k in $\{10, 20, 50\}$. Please recall that there are around 5,000 simultaneous online channels for Twitch and 400 for YouTube Live. These top channels represent a small fraction of all online channels. Overall, for each month, we gathered more than 8,500 different lists (one list every five minutes) of top-k channels.

We show that a small number of distinct channels appears in these top-channel lists over the whole months. Table 5.2 shows the data over the months. On Twitch only 975 channels (0.06% of the total) have appeared in the top-10 channels in the three months. It means that 975 channels have occupied the over 85,000 "spots" that were available. The same aspect is detected on YouTube Live traces. Only 1167 (0.4%) channels have appeared on top-10.

As pointed out in our analysis the live services can reach peaks of outgoing bandwidth traffic of 1 Tbps. This amount of bandwidth can justify the need for interfacing the delivery system with a CDN.

We measure the popularity of the top channels and calculate their footprint on the overall delivery bandwidth traffic. For Twitch we used the available bit

rates, while for YouTube we considered, as previously discussed, the average 2Mbps. Thus, we extrapolate an approximation of the total bandwidth used by the delivery systems. This information is depicted in Figure 5.1. First, the popularity follows Zipf's law, as expected the popularity of top channels decreases fast. The gap between top-10 and top-50 channels is relatively small in respect to the overall traffic. By adding in the top list 40 more channels we have only 8% more created delivery traffic in average for YouTube Live. Thus, the peak of global popularity can be exclusively credited to the most 10 (or even less) popular channels. There is a direct correlation between peak of global system popularity and peak in popular channels.



(a) Twitch (b) YouTube Live

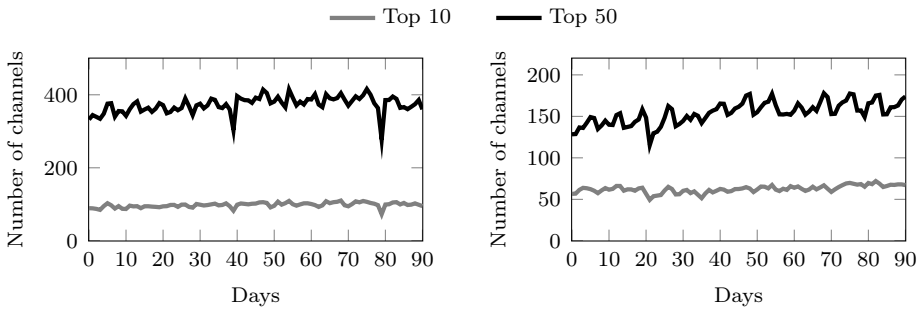
Figure 5.1: Bandwidth usage ratio (from the delivery bandwidth for all channels) on each hour for most 10 and 50 popular channels for the first 7 days of our traces.

The most popular channels have high resolutions.

The video resolutions collected from the most popular channels are associated to high-definition quality (720p and 1080p), as discussed in Chapter 3. To deliver the video content of higher resolutions more bandwidth is consumed. This stress the gain when usage of hybrid-delivery with data centers and CDNs. Integrating CDNs to deliver the popular channels will diminish the burden of such high consuming bandwidth channels from the data center infrastructure.

The number of simultaneously online popular channels is stable.

Is interesting to have a stable set of channels on each part of the delivery infrastructure (CDN and data center) in order to avoid extra costs of migration between parts. We explore the number of popular channels that are *simultaneously* online at a moment in time. We measure the number of online channels out of the overall population of top-k channels, every five minutes. We present the average number of simultaneous popular channels by day in Figure 5.2.



(a) Twitch (b) YouTube Live

Figure 5.2: Average number of simultaneously online popular channels.

The number of simultaneously online popular channels out of the entire population of channels is both stable and small. Typically for the set of around thousand channels that occurred at least once in the top-10 channels, the average number of online channels is 98 for Twitch and 62 for YouTube Live. These channels could be specifically assigned by the delivery algorithm and handled to the CDN. To conclude, we believe that live streaming platforms can easily interface with a hybrid DC-CDN model because a small and stable population of channels is responsible of the traffic peaks.

5.5 Evaluation

In the evaluation of our model we perform comparisons of the DC-CDN composition. This composition, by definition, tries to minimize costs by using the advantages of both CDN and DC components.

We implement the formalization model as a linear problem and solved it with the generic solver IBM ILOG CPLEX [IBM14]. We perform evaluations on the first week of our real traces of Twitch and YouTube Live described at Chapter 3.

To evaluate the scenario we implemented other four simple and intuitive live strategies that could be used on real time by any live service. The strategies do not consider the usage of multiple components for a given channel at the same moment in time. They are responsible for deciding if a channel will be delivery by DC or CDN, exclusively. The four strategies are:

Top-i migration. This strategy consists in delegating to a CDN the i most viewed channels. For example, with $i = 1$ we choose the most viewed channel of each five minutes to be delivered by a CDN. In the case of $i = 2$, we select the two most viewed channels of each five minutes.

The chosen channels will be migrated from one component to the other accordingly to their popularity. For example, if a channel becomes unpopular it can be assigned back to DC.

Top-i history. This strategy also consists in delegating to a CDN the i most viewed channels. Additionally channels that have once been delegated to CDN in the past will directly be assigned to the CDN independently of their popularity. In this strategy there is no migration of channels back from CDN to DC. And only once channels are migrated to CDN.

Threshold-j migration. This strategy transfers to CDN channels that popularity is more than, or equals to, a defined value j. For example, if $j = 10,000$, a channel that arrives to, or more than, $10,000$ simultaneous viewers at determined moment in time will be reallocated to CDN. Migration is allowed from one component to the other accordingly to their popularity.

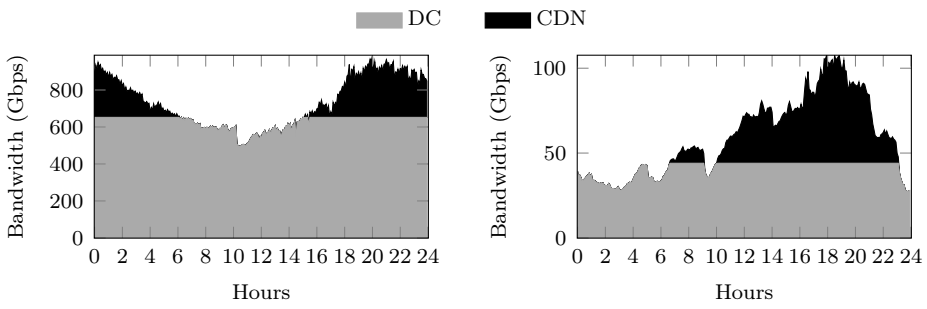
Threshold-j history. As the strategy threshold-k, it delegates to CDN the channels which popularity is more than, or equals to, a defined value j, and additionally it keeps in CDN all the channels that have ever been in CDN. In this strategy there is no migration of channels back to the DC and only once channels are migrated to CDN.

To provide the comparison between the optimal model result and the intuitive strategies we evaluate both total costs of the last day of the chosen week (January 12, 2014). Since the strategies *top-i history* and *threshold-j history* require the past information of the channels, we used the first days of the week as training data to the algorithms. As we previously discussed in this chapter, the most popular channels are stable and within few days of traces the strategies can accumulate enough information about them.

Settings		
Input	YouTube Live	Twitch
P_{vt}	Traces (1st week - January 6, 2014 to January 12, 2014)	
b_{vt}	2 Mbps	
I	[Data Center, CDN]	
r_i	[0,0]	
u_i	[44 Gbps, 4 Tbps]	[652 Gbps, 4 Tbps]
c_i	[0.0017,0.1]	
s_i	[0.1,0.2]	
d	0.1	

Table 5.3: Evaluation settings

Table 5.3 shows the settings we used for the evaluation with Twitch and YouTube Live traces. These values were chosen based on real services pricing such as Amazon Simple Storage Service (Amazon S3[1]) and Twitch information [Hof12]. We performed an evaluation of the strategies and retrieved the optimal solution with the linear problem. Figure 5.3 shows the optimal result achieved with CPLEX for the bandwidth distribution between DC and CDN.

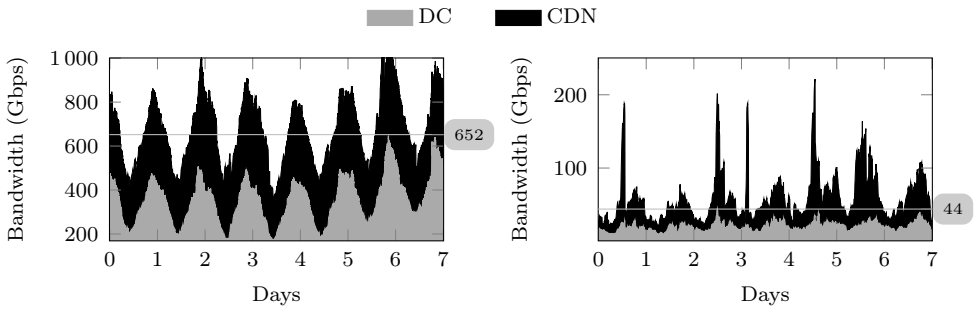


(a) Twitch (b) YouTube Live

Figure 5.3: Optimal bandwidth usage on hybrid DC-CDN delivery composition.

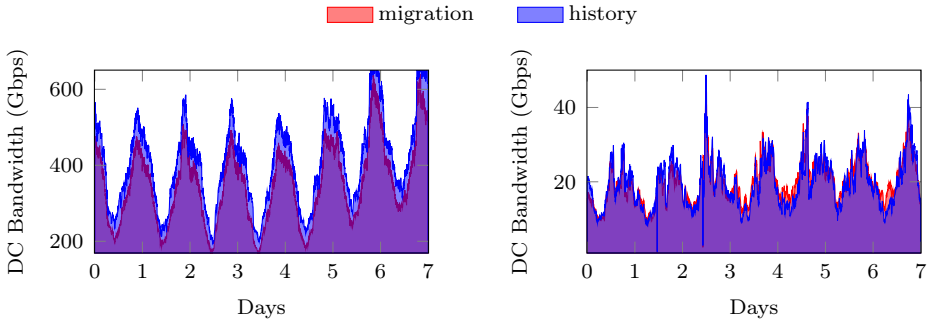
On the optimal solution the usage of the DC bandwidth is always maximized, as expected. We defined the DC bandwidth capacity as the average utilization of the services, 652 Gbps for Twitch and 44 Gbps for YouTube Live. The cost for using the DC bandwidth is smaller than for the CDN's one. Therefore, deliver strategies should try to maximize the usage of the DC

[1]http://aws.amazon.com/s3/



(a) Twitch (b) YouTube Live

Figure 5.4: An example of DC and CDN bandwidth usage for the *top-10* strategy over the week. We highlight the averages of total bandwidth usage for Twitch (652 Gbps) and YouTube Live (44 Gbps).

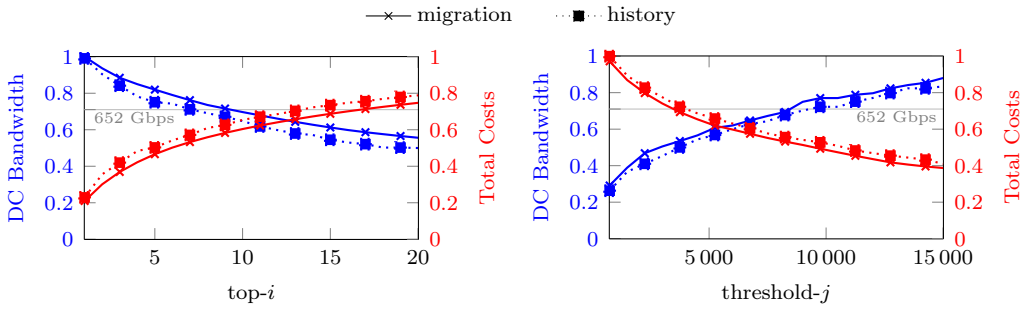


(a) Twitch (b) YouTube Live

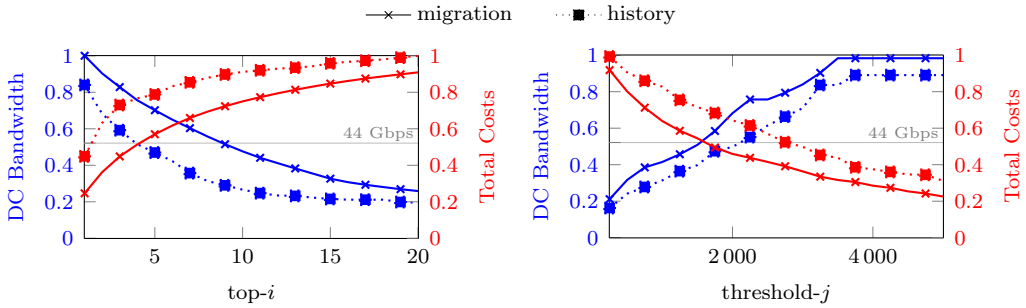
Figure 5.5: Comparison of DC bandwidth usage for the *top-10 migration* and *top-4 history* strategies over the week.

but also respect the its maximum capacity. Depicted on Figure 5.4 is one example of the bandwidth usage on DC and CDN for the *top-10 migration* strategy. The usage of bandwidth on of the services is highly heterogeneous. Peaks of usage are orders of magnitude bigger than the valleys, around 1 time for Twitch and 4 times for YouTube Live. This simple strategy successfully attributed the peak of bandwidth usage to CDN, specially on the traces of YouTube Live.

Comparisons between a strategy (top) with migration and history are presented by Figure 5.5. More specifically we compare *top-10 migration* and *top-4 history* strategies. The migration of sessions from one component to another can have extra costs and affect the quality of experience of the viewers. These results indicate that a history strategy can achieve a similar DC bandwidth usage in comparison to the migration strategy with a lower selection of



(a) Twitch



(b) YouTube Live

Figure 5.6: Normalized bandwidth and costs comparison of different channel assignment strategies for hybrid DC-CDN delivery.

channels, for example $i = 10$ for migration and $j = 4$ for history.

At Figure 5.6 we show the normalized costs comparison between strategies for Twitch and YouTube Live traces. We present the strategies regarding migration and history with different line and marker types. The different colors on each side of the axis are related to: on left side is the normalized DC bandwidth; and on the right side is the normalized total costs. The normalized bandwidth and costs were calculated by dividing the original values by the maximum found on the analyzed day (January 12, 2014). We vary *top-i* and *threshold-j*. We picked i from 1 to 20 on both traces. For Twitch we selected j from 750 to 15,000, with steps of 750, totaling 20 different js. For YouTube Live j ranges from 250 to 5,000, with steps of 250, also totaling 20 numbers. We indicate the average bandwidth capacity for the services with an horizontal line.

The results on Figure 5.6 indicate the correlation between DC bandwidth usage and total costs. The costs are lower when the the load on the DC is big-

ger. However the dimensioning of the DC must respect two conditions. First, the DC normally has a fixed capacity (pre-paid cloud or physical constraints) and the peaks of bandwidth usage should not excess its total capacity. Second, the DC dimension should avoid valleys of usage and waste of resource, in our formulation we do not consider the costs of having part of the DC unused. Additionally, we have more evidence that costs of migration can be avoided by the usage of history strategies that take into account the information of channels past sessions.

5.6 Conclusion

In this chapter we formalized the delivery of live streaming channels. We presented the results of the formal model as well as the comparison of other four intuitive strategies. We show that simple strategies can successfully assign peaks of bandwidth usage to CDN on a hybrid delivery composition. We also detected that migration costs of sessions between the different components of the delivery can be avoided using the past channel information.

As future works we aim to explore other comparisons, including other compositions of hybrid delivery, for example P2P-CDN and DC-P2P, considered in our formalization. Also, we obtained different results from both system traces. Such system particularities could be considered in the future in order to define better delivery strategies.

Chapter 6

Transcoding for Adaptive Streaming

Contents

6.1 Introduction

In the recent years, video providers have deployed Adaptive Bit Rate (ABR) streaming to cope with the heterogeneity of devices and network connections, for instance mobile phones with 3G/4G network, tablets on WiFi, laptops and TVs with ADSL and fiber. This technique consists in delivering multiple video representations and besides being applied to live streams is also adopted by the online services of TV companies [Wei14]. However, as discussed at Chapter 2, the implementation of ABR in live streaming services is limited to a small number of video channels in dedicated servers. In the case of massive live streaming platforms such as Twitch, both the large number of concurrent channels and the use of commodity servers in data centers bring new issues.

Implementing ABR in massive live streaming platforms yields some benefits and costs. The costs mainly come from the fact that for each video

channel, the raw live video stream should be transcoded into multiple live streams at different resolutions and bit rates. The consumption of computing resources needed for the transcoding induces significant costs, especially with regards to the large number of concurrent video channels. On the other hand, the benefits include the improvement of the QoE for the end-users and a reduction of the delivery bandwidth costs.

In this chapter, we study the trade-off between benefits and costs for the implementation of ABR in live streaming video platforms. In our simulations we target specifically Twitch because YouTube Live does not provide information about sessions bit rate, which is an information needed to evaluate the implementation of ABR. Also in our live sessions data set no specific information about the viewer individual information is available. For this ABR study we have a need for such detailed information, which lead us to explore a third-party data set presented in this chapter.

Our live sessions data set contains only the total number of viewers watching a channel. To better understand how viewers could benefit from ABR solutions, and evaluate the gain of transcoding operations, we introduce a more grained data for the users. We adopted the data set provided by [BSMM14]. This data set is a collection of DASH sessions from thousands of users geographically distributed. To have a more realistic scenario in the study, an association of each viewer from the live sessions data set to a given user of this data set was made. By doing that we were able to evaluate the end-to-end (from live service provider to final viewers) benefits and trade-offs of ABR solutions and video transcoding. This data set is used in two studies, this current chapter and further on the Appendix C.3.

The chapter is organized as following. First we describe the DASH sessions data set. Next, we highlight some characteristics of Twitch in Section 6.3. We highlight that one key problem is to decide which video channels should be broadcasted as usual (i.e. by directly forwarding to the viewers the raw video received from the broadcaster) and which channels should be delivered with ABR streaming. We then present two strategies, according to whether the decision of delivering a given channel by ABR can be taken while this channel is online (*on-the-fly strategies*) or only when the channel starts broadcasting (*at-the-startup strategies*). In Section 6.4 we compare both strategies in a realistic scenario based on our live sessions traces from Twitch, detailed at Chapter 3, and the DASH sessions data set for the viewers population.

6.2 DASH Sessions Data Set

The viewers bandwidth distribution is a fundamental information to create realistic scenarios for the ABR problems we target in our work. A data set that captures the characteristics of a real population of viewers, and in particular its heterogeneity, was needed. The APIs of the service providers does not provide any individual information about the viewers. In order to retrieve this information we analyzed the public available data set provided by [BSMM14]. This data set contains thirty seconds DASH sessions information, with chunks of two seconds, of thousands of users geographically distributed. It was collected by a DASH module built on top of Neubot. Neubot is a research project based on an open source program that runs in the background on thousands of Internet clients and periodically performs tests, for example the one in the DASH module. Figure 6.1 shows examples of three sessions of different users.

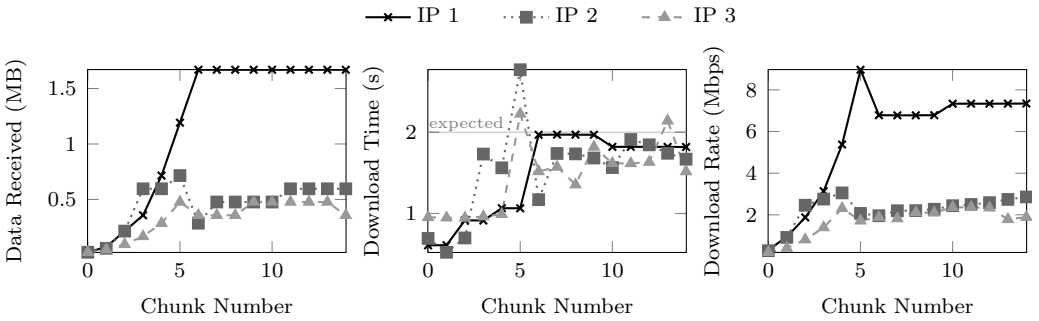


Figure 6.1: Example of three sessions from the viewers data set. The figure on the left represents the amount of data received for each chunk. In the middle figure, the time to download each chunk is illustrated. The right figure shows the download rate of users, calculated by the amount of data received (left) divided by the download time (middle).

Different metrics are available in the data set, for example: server time stamp; program version; client Operating System (OS);, we selected the ones described by Table 6.1a.

In this data set a unique user can connect from different places, for example at work and at home. Therefore, this user will have very inconsistent bandwidth rates among its sessions. To avoid such inconsistency, we decided to consider a user as its identification and IP address combined. In this case, for our simulations, a user accessing from home and from work will be considered as two different users. The amount of users found in the data set is described at Table 6.1b.

Name	Description
client_uuid	user identification
client_real_address	user IP address
client_iteration	chunk number inside a session
client_received	data received in bytes
client_elapsed	time in seconds to download the chunk

(a) Used fields

Selection	Total Number of Viewers
unique IDs	1 911
unique IPs	22 330
unique IDs+IPs	23 028

(b) Unique users

Table 6.1: Summary of information and amount of unique viewers provided by viewers data set

Another consistency filter that we considered was regarding the DASH adaption impact on the user's downloading rates. As we can see in Figure 6.1, the initial chunks of each session are far too small to provide an accurate download rate. Accordingly, to consider the stable bandwidth and to avoid any impact of the data set DASH adaption algorithm we assume as noise the initial five chunks of each session by excluding these points in our simulations. Our simulation sessions have 10 chunks each, totaling 20 seconds sessions.

Each one of the 23 028 users have a different amount of sessions in the data set. The 5^{th} plot in the Figure 6.2 presents the users CDF for the number of sessions found in the data set. Is worth to note that, more than 30% of unique users (7 221) have ten or more sessions.

As input for our simulations we decided to consider a population of 500 users. We selected the users with most high number of sessions and with 75 percentile of the download rate lower or equal to 8 Mbps (from the entire number of users, 19 359 attend this condition). Figure 6.2 illustrates different metrics over the download rate (bandwidth) of the 500 users we selected and the overall collection of users. From this collection of 500 users we collected 100 sessions each and thus obtain 500,000 samples of realistic download rates.

Scripts, made in Python and R, for parsing and extracting statistical information from this data set are public available.[1]

6.3 Which Channels to Transcode

We recall in the following four main characteristics from the Twitch data set presented in Chapter 3 that are relevant for this chapter.

Delivery Needs. We evaluate the overall bandwidth needed to deliver video channels to the viewers (we do not take into account the bandwidth required

[1] https://github.com/karinepires/neubot-dash-parser

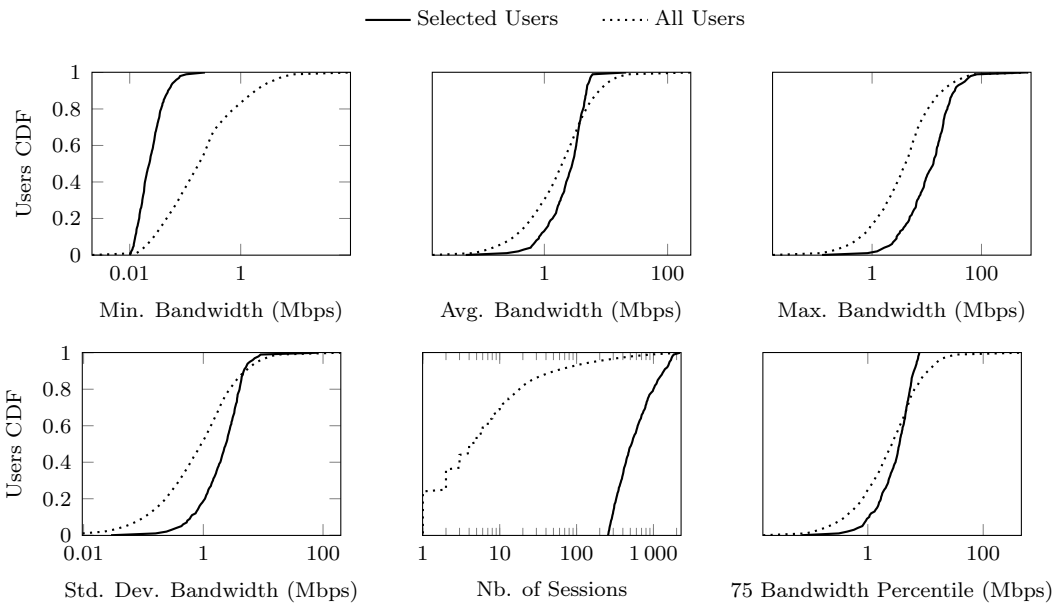


Figure 6.2: Different statistical metrics of all and selected users

from the broadcasters to Twitch data center). Twitch is a regular Over-The-Top (OTT) service with unicast transmission to viewers, so we sum up the bit rates of each session multiplied by the number of viewers for this session. We presented in Chapter 3 that the daily bandwidth peak is often more than 1.5 Tbps with a peak at more than 2 Tbps. Moreover, the delivered traffic is sustained with minimum daily bandwidth always above 400 Mbps.

Computing Needs. Another infrastructure cost is the data center, which, among others, processes the incoming raw video from broadcasters and prepares the streams to be delivered. We consider the average number of concurrent online channels as a metric for estimating the data center dimensions. Between 4,000 and 8,000 concurrent sessions always require data center processing in Twitch. Such sustained incoming traffic requires a computing infrastructure that, to our knowledge, is unique in the area of live streaming.

Channel Popularity. We confirmed in Chapter 3 that Twitch popularity follows a Zipf law. What is important here is the high value of the Zipf coefficient α for Twitch when compared to other services. Although α is often lower than 1 in other UGC platforms, it is always larger than 1.3 over the three months, and even sometimes above 1.5. Such a large α coefficient characterizes both a sharp difference between the most popular channels and the others, and a long tail of unpopular channels.

Raw Videos. The raw live stream is the video encoded at the broadcaster side and transmitted to the data center of Twitch. This video can be en-

coded in various resolutions and bit rates. Our data set analysis confirms the compromise between quality and popularity of the sessions. As described at Chapter 3, sessions with videos at a resolution lesser than 720p represent 40% of the total amount of sessions but they attract only 8% of the total viewers. While considering the bit rates found in Twitch data set, the ones of 720p and 1080p channels are significantly higher than for 480p channels. For example, only half of the video sessions at both 720p and 1080p have a bit rate lower than 2Mbps although such a bit rate is larger than 90% of the bit rates of 480p channels.

From these characteristics we extract three main ideas. First ABR streaming is required to cope with high delivery cost and with the inaccessibility of the most popular channels for a fraction of the population. Second, applying ABR to all channels requires a significant amount of computing resources because the number of concurrent sessions is high. Third, all channels should not be treated equally since only a few ones are popular. In the following, we introduce the problem of deciding the subset of channels to which ABR should apply. We then present two strategies.

6.3.1 Trade-off and Problem Definition

We introduced in Chapter 2 two types of process when the raw live video of an online channel is received by live streaming services. First the *traditional process* that consists of preparing the raw live stream and then delivering it directly to the viewers requesting it. Second the *transcoding process* that consists of transcoding the raw live stream into multiple live video streams and of using ABR streaming to deliver the sessions to the viewers.

We envision that only a fraction of online channels should use ABR streaming: the popular channels with high bit rate and resolution. For a given channel, ABR streaming generates two main benefits.

First, it improves the QoE for all the viewers having a downloading rate inferior to the raw video bit rate. We call them the *degraded viewers*. Without ABR, the degraded viewers experience video buffering at a frequency that depends on the difference between their downloading rate and video bit rate. Such a bad QoE causes churn and degrades the reputation of the platform.

Second, ABR streaming reduces the overall needed bandwidth for serving a population. Without ABR, Twitch should deliver the raw live video to all viewers (including the degraded viewers since the delivery of the overall video session is delayed in time but the overall amount of data to be transferred is roughly the same). On the contrary with ABR, the degraded viewers are served with a video stream at a lower bit rate than the raw live stream.

The problem is to decide which process should apply for every online chan-

nel at a given time with respect to the trade-off between extra-cost induced from the transcoding process and the gains. To make it harder, switching from one process to another *while the channel is online* is not trivial. A possible implementation is to use ABR streaming technologies for both decisions. The *representation set* of a channel in the transcoded process contains the multiple live video streams while it contains only the raw live stream for channels in the traditional process. Thus, switching from one process to another *on-the-fly* requires a revision of the ABR *manifest* with the new set of video representations and the notification of all users about the manifest revision. Both actions are not always possible with respect to the ABR streaming technology.

Due to this uncertainty, we distinguish:

On-the-fly strategies, where the decision of whether an online channel should be transcoded or not can be taken at anytime during a session. For example, the session 1 in Figure 6.3 starts at time t_1 and ends at t'_1. An on-the-fly strategy can decide to transcode or not the channel at anytime between t_1 and t'_1.

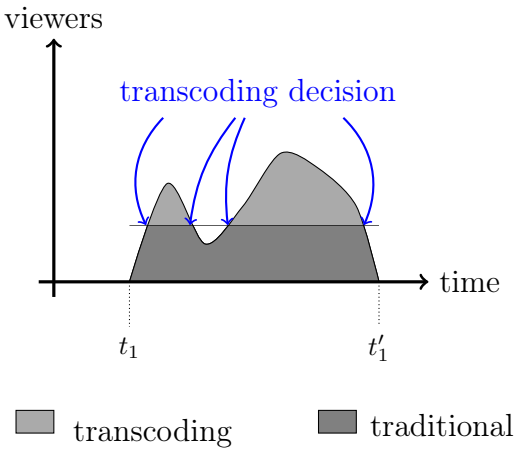


Figure 6.3: On-the-fly strategy

At-startup strategies, where the decision of whether an online channel should be transcoded or not can be taken only when the session starts. An example is depicted by Figure 6.4. The decision at start time t_1 applies for the whole session 1.

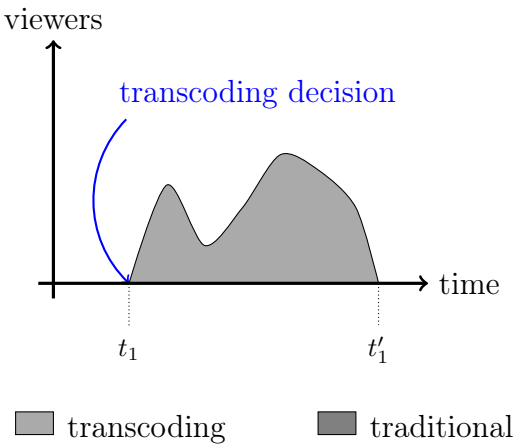


Figure 6.4: At-startup strategy

In the following, we present a simple implementation for each strategy type. One of our goals is to keep the strategy as simple as possible for a fast implementation.

6.3.2 An On-the-Fly Strategy

An on-the-fly strategy is supposed to perform well since it is reactive to any event, including unexpected events like flash crowd on a video channel or abrupt disinterest on another. That is, it is possible to build a strategy that is close to the optimal (with respect to the ability of the platform provider to quantify the QoE of degraded viewers). Our goal is not to define the best strategy but rather to highlight the performances of on-the-fly strategies in general. We thus design a simple strategy as follows.

First, we filter the broadcasters so that only broadcasters with a raw live video at resolutions 480p, 720p, and 1080p are considered as candidates for ABR streaming. Then, we define a value j which is the threshold on the number of viewers. Every five minutes, all candidate broadcasters with more than j viewers are selected to be transcoded. We call this strategy *threshold-j*.

6.3.3 An At-Startup Strategy

An at-startup strategy requires to foresee the popularity of a channel in the near future. Predicting video popularity has become an important research topic with connection to massive data treatment, artificial intelligence, and social network observations. However, with the same motivations as for the on-the-fly strategy, we rather stay simple in this work and left for future works the design of sophisticated efficient strategies.

To predict the popularity of channels, we only use the history. The strategy comes from the observation that a channel that has been popular in the past will be popular in the future. To estimate how popular was a past session with respect to the context at that time, we focus on a simple measure: the top-k channels, *i.e.* the k most popular channels. Every five minutes, we collected the k most popular channels. Overall, for each month, we gathered more than 8,500 different sets of top-k channels. We analyzed these sets and observed that the number of *distinct* broadcasters is low. Typically for top-10 sets, around 500 broadcasters occupy the over 85,000 "spots" that are available every month.

We derive from this observation an at-startup strategy, which we call *top-k*. On a periodic basis, we get the k most popular broadcasters and we insert them to a list of candidate broadcasters. When a new session starts, the decision to apply the transcoding process to the broadcaster is taken if the broadcaster is in the list of candidates and if the raw live video has a resolution of 480p, 720p, or 1080p. Notice that we do not implement any mechanism for removing broadcasters from the candidate list after a while, so this list continuously inflates with time. We expect to implement such a mechanism in future works.

6.4 Evaluation

We now evaluate the performances of both threshold-j and top-k strategies and also show the feasibility of the implementation of ABR for Twitch.

6.4.1 Settings

We use two real data sets for the setting of our evaluation. The first data set is our Twitch data set as described in Chapter 3. One time unit is five minutes, as the time needed to refresh the API at Twitch. We use the information of sessions (video bit rate, video resolution) and broadcasters (number of viewers). Simulations were done with three months of data and the figures present results of the last month.

Population Download Rate Settings. The second data set comes from [BSMM14], which was previously presented. Since the Twitch API does not provide any information about the viewers (neither their geographic positions, nor the devices and the network connections), we need real data to set the download rates for the population of viewers. The data set presented in [BSMM14] gives multiple measurements over a large number of 30s-long DASH sessions from thousands of geographically distributed IP addresses.

From their measurements, we infer the download rate of each IP address for every chunk of the session and thus obtain 500,000 samples of realistic download rates. After filtering this set to remove abnormal rates, we randomly associate one download rate for every viewer.

Multi Bitrate Video Transcoding. When a session is selected for a delivery by ABR streaming, the raw video is transcoded into multiple live streams. We consider the creation of one stream per resolution, only in smaller resolutions than the one of the raw video. The bit rate of each live stream depends on the bit rate of the raw video, as given in Table 6.2. For example, let the raw video have a bit rate of 2,000 kbps and resolution 720p. The transcoded streams have a bit rate of 1,400 kbps for the 480p stream, of 1,000 kbps for the 360p, and of 600 kbps for the 224p.

		input resolution		
		480p	720p	1080p
output res.	224p	$0.5 \times n$	$0.3 \times n$	$0.25 \times n$
	360p	$0.7 \times n$	$0.5 \times n$	$0.3 \times n$
	480p	n	$0.7 \times n$	$0.5 \times n$
	720p		n	$0.7 \times n$
	1080p			n

Table 6.2: Bitrate of the streams transcoded from a raw video stream having bit rate n kbps

Strategies. We evaluate the *top-k* and *threshold-j* strategies against two naive and extreme strategies: the *none* strategy with no ABR implementation, and an *all* strategy where all online channels are delivered by ABR. The former represents the current state of live game streaming platforms while the latter is an upper-bound of an implementation of ABR streaming. For *top-k* and *threshold-j*, we set $k = 50$ and $j = 100$ by default.

6.4.2 Evaluations

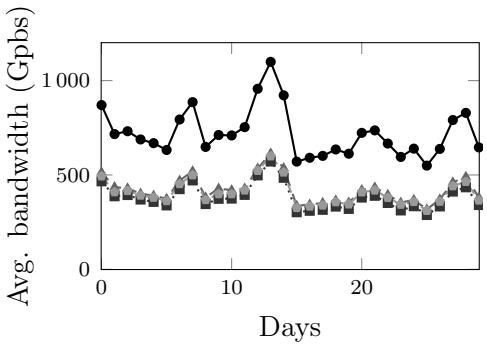


Figure 6.5: Bandwidth needed for streams delivery

In our evaluation we adopt three metrics (bandwidth, viewers QoE and computing needs) that highlight the trade-off related to the implementation of ABR on live streaming services. In Figure 6.5, we measure the average daily bandwidth that is needed to deliver the video channels. The first observation is that implementing ABR generates a non-negligible reduction of the delivery. The aggregation of bandwidth savings on every degraded viewers can nearly halve the bandwidth between both extremes *none* and *all* strategies. The *top* and *threshold* strategies are close to the *all* strategy, which validates our strategy of implementing ABR streaming only for the most popular channels.

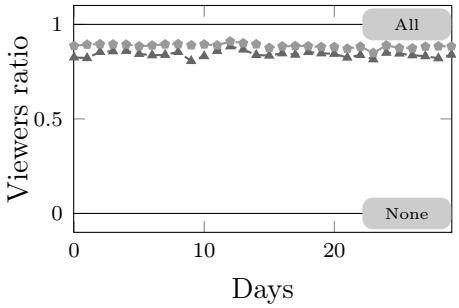


Figure 6.6: Ratio of addressed degraded viewers

In Figure 6.6, we measure the gain in QoE for the viewers. As this being our initial study, we focus on only one simple metric, which is the ratio of degraded viewers that can find in ABR streaming a video stream with a bit rate lower than their download rate. In the next chapter we introduce a more precise metric. We observe in the data set that, even with ABR, some degraded viewers have still a download rate lesser than the bit rate of the 224p stream. What we measure here is the ratio of the "addressable" degraded

viewers. Again the *all* strategy gives an upper-bound, and both *top* and *threshold* strategies allow most of these viewers to enjoy a video stream that fits with their network connection.

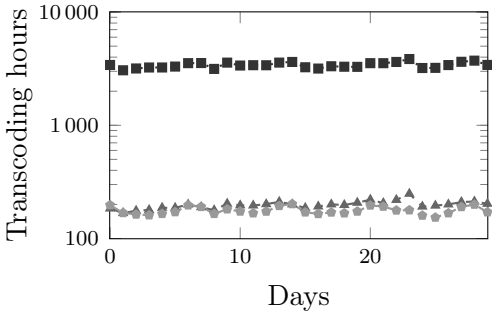


Figure 6.7: Transcoding hours

Finally in Figure 6.7, we measure the computing needs with a metric in *transcoding hours* inspired from the current practice in commercial cloud transcoding offers (*e.g.* Zencoder[2]). Both input and output resolutions are used to determine the amount of transcoding hours. For example, a given stream with 1 hour length and resolution 480p transcoded to 360p and 224p is counted as 3 hours of transcoding (1 input + 2 output). High-definition resolutions (720p and 1080p) double the transcoding hours. In Figure 6.7, the main observation is that we have to use a log scale on the y-axis to keep it readable. It means that the transcoding hours of the *all* strategy is one order of magnitude larger than for the *top* and *threshold* strategies.

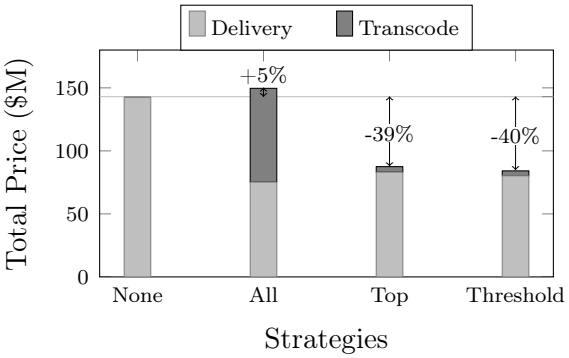


Figure 6.8: Estimation of the total infrastructure costs

To summarize our results in a more practical way, we estimate the overall infrastructure cost for the four strategies. We use the publicly available prizes

[2] http://zencoder.com/

of Zencoder and Amazon CloudFront[3] to estimate the cost of the transcoding hours and the delivered bandwidth, respectively. We present in Figure 6.8 the synthesis. We see on the *all* strategy that the gain in delivery are unfortunately counter-balanced by the costs in transcoding. Both *top* and *threshold* strategies find a better trade-offs with a significant reduction of delivery cost at a negligible transcoding cost. Finally, we show that the performance gap between *top* and *threshold* strategies does not necessarily justify the implementation of *on-the-fly* strategies. Sophisticated *at-startup* strategies are expected to even reduce the gap.

6.4.3 Playing with Strategies Parameters

We evaluate the impact of parameters k and j for the *top* and *threshold* strategies respectively. In Figure 6.9 we represent the total estimation costs for different values of k and j. What is interesting here is that these parameters allow Twitch to adjust the trade-off between delivery and transcoding according to any external constraint (*e.g.* price variation, data center load and maintenance operation).

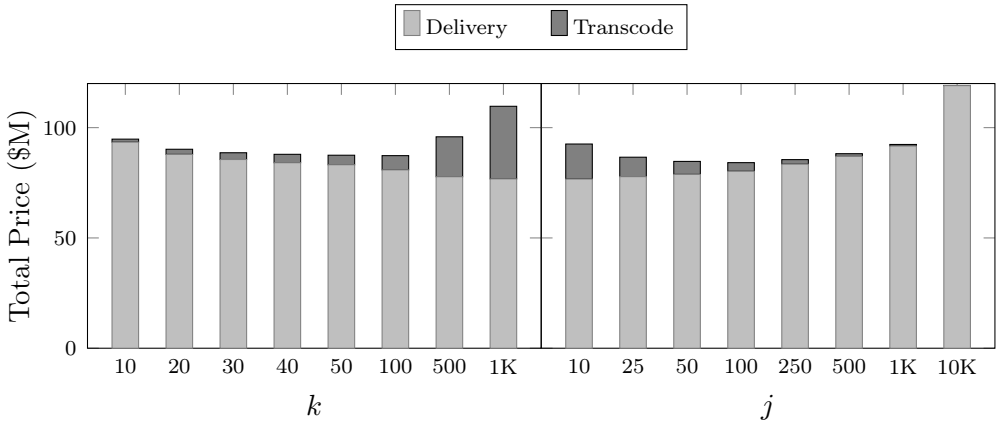


Figure 6.9: Estimation of the total infrastructure costs for different values of k and j in *top* and *threshold* strategies

6.5 Conclusion

This chapter presents our work in the more general context of implementing interactive multimedia services at a massive scale. We introduce an infrastructure management problem and we reveal that some simple strategies can

[3] http://aws.amazon.com/cloudfront/

significantly cut the overall infrastructure costs while increasing the QoE of end-users.

Many future works can derive from this study. The first derivation is presented in Appendix C.3. In this appendix we evaluate a formalization of the transcoding problem with our live sessions data set.

Other works like more efficient strategies can be designed. We present here simple strategies but more sophisticated strategies are expected to yield better performances, especially for at-startup strategies. A more comprehensive analysis of the levers that make a session become popular as well as recent statistical approaches to deal with popularity forecasting represent appealing research. As well, the integration in practical platforms and real implementation also bring additional difficulties, from a technical perspective with, for example, the integration of standard, but also from a business perspective. Typically, Twitch has recently started offering ABR for some "partner" premium broadcasters.

Chapter 7

Conclusion

Contents

The popularization of devices capable of capturing videos, the continuing efforts to increase network quality and the success of the live streaming services have lead these services to be responsible for an important share of the Internet traffic. In this thesis we have explored approaches to improve live video streaming services. Our objectives are to diminish resource waste, such as server computation and network bandwidth, and to improve the quality experienced by users using these services.

Live streaming services stresses the actual Internet infrastructure. Twitch had to increase the delay in their platform and limit the adaptive bit rate delivery to partners. YouTube Live restricted the service access to a selection of users for 2 years. These decisions limiting the services emphasize the demand for improvements. In our work we focus on two main challenges raised specifically by the massive amount of content produce by users the *delivery* and *transcoding* of live streaming systems.

7.1 Synthesis

We explored and developed solutions for the delivery of the massive content. One of the challenges is the huge variation in the total number of viewers and the great heterogeneity among streams popularity, which generally implies over-provisioning and consequently lead to an important resource waste. In this thesis, we show that there is a trade-off between the number of servers involved to broadcast the streams and the bandwidth usage among the servers. We also stress the importance to predict streams popularity in order to efficiently place them on the servers. We propose three solutions that address different aspects concerning the delivery of live sessions.

We target the difficulties concerning transcoding of live streams. The transcoding operations over streams are computational intensive and are key operations on adaptive bit rate streaming. We show that adaptive bit rate streaming is able to reduce the delivery bandwidth cost and to increase viewer quality of experience at the cost of computing resources for transcoding purposes. To address the trade-off between benefits and costs, we formulate two management problems. The main contributions of this thesis are: (*i*) live streaming sessions data set; (*ii*) approaches for cloud delivery; (*iii*) hybrid delivery solutions; (*iv*) analysis of ABR on live streaming; (*v*) data set application on CDN delivery and cloud transcoding.

7.1.1 Live Sessions Data Set

First, we created a live streaming sessions data set. To study the aspects we target of live streaming systems we need to characterize them and evaluate the proposed solutions with relevant input traces. Therefore our first contribution is a data set and its analysis. The data set contains three months traces of two UGC live streaming services. We made it freely available for the community. It has millions of live sessions and hundreds of thousands broadcasters. To the best of our knowledge this data set of Twitch and YouTube Live is the first attempt to understand the behavior of UGC live streaming videos systems with comparison of two largely used systems. We bring with our study useful insights on what can be expected of a massive UGC live streaming video service. We show that 1 Tbps of bandwidth is commonly achieved in order to deliver the contents of such system. We also point out differences between Twitch and YouTube Live, especially regarding the broadcaster behavior. Broadcasters on Twitch are clearly more engaged on producing live streams than the ones found in YouTube Live. We detected many characteristics that indicates the maturity of Twitch environment. Compared to VoD service, live streaming video services broadcasters are more active. Aside from

other systems, the popularity found on live systems is highly heterogeneous. We then use the findings of our analysis to conceive our solutions and the real data set information as input to our simulations, which provides a close to real simulation scenario. By making the data set public available we hope to enrich the community and help to improve live streaming systems.

7.1.2 Cloud Delivery

This solution is designed for platforms such as clouds or distributed virtual machines and uses popularity predictions to dynamically provision servers, and map live sessions on the servers. The user-generated content live streaming platforms have to face a high number of simultaneous video sessions that they have to collect and distribute to an even higher number of viewers. We show that taking into account an estimation of the future popularity of incoming sessions is important while mapping them to a set of servers. This may help to prevent session partition across multiple servers, inefficient in terms of inter-server network cost. We base our study on real data sets collected on the YouTube Live and the Twitch platforms. We study the number of servers versus inter-server bandwidth usage trade-off. Our approach, POPS, uses popularity peak estimation of broadcasters while placing new sessions on broadcasting servers.

7.1.3 Hybrid Delivery

This solution is assisted delivery involving proprietary servers and Content Delivery Network (CDN). We first formalized the delivery of live streaming channels. We have then presented the results of the formal model as well as the comparison of this theoretical optimal with other four intuitive strategies. The results of our study reveal a real need for exploring and designing smart strategies for the delivery of live streaming videos.

7.1.4 DASH on Live Streaming

We design two strategies to decide which online channels should be delivered by adaptive bit rate streaming. We focus in the more general context of implementing interactive multimedia services at a massive scale. We introduce an infrastructure management problem and we reveal that some simple strategies can significantly cut the overall infrastructure costs while increasing the QoE of end-users.

7.1.5 Data Set Applications

One of our presented contributions is a live session data set. This data set opens opportunities to introduce real traces on simulations regarding live streaming systems. In Appendix C we describe two collaborative works, which we applied the live session data set.

First, we discuss the economics aspects related to CDN based delivery. We propose a model to analyze of the policy that a profit-driven CDN should implement when delivering live video content. This model is especially significant with regard to the multiple recent debates about network neutrality. To the best of our knowledge, this is the first attempt to model CDN from an economic standpoint with the ambition to understand the impact of CDN on the content delivery market.

Second, we present an integer linear program to maximize the average user quality of experience and a heuristic algorithm that can scale to large number of videos and users. We study the management of new live adaptive streaming services in the cloud from the point of view of streaming providers using cloud computing platforms. All the simulations we conducted make use of real data from three data sets covering all the actors in the system. The study is focused on the interactions between the optimal video encoding parameters, the available CPU resources and the QoE perceived by the end-viewers. We use an Integer Linear Program (ILP) to model this system and we compare its optimal solution to current industry-standard solutions, highlighting the gap between the two. Due to the ILP computational limitations, we propose a practical algorithm to solve problems of real size, thanks to key insights gathered from the optimal solution. This algorithm finds representations beating the industry-standard approaches in terms of the trade-off between viewers QoE and CPU resources needed. Furthermore, it uses an almost-constant amount of computing resources even in the presence of a time varying demand.

7.1.6 Additional Contributions

Additionally to the main contributions, presented in this thesis and the appendix, we have also collaborated with minor contributions to other two related challenges to video streaming alongside this thesis studies.

We utilized the DASH session data set, presented in this thesis, to evaluate the proposed solution performance in [TAP+14]. First an integer linear program that maximizes users' average satisfaction is formulated, taking into account network dynamics, type of video content, and user population characteristics. The solution of the optimization is a set of encoding parameters

corresponding to the representations set that maximizes user satisfaction. We evaluate this solution by simulating multiple adaptive streaming sessions characterized by realistic network statistics, the DASH session data set, showing that the proposed solution outperforms commonly used vendor recommendations, in terms of user satisfaction but also in terms of fairness and outage probability. The simulation results show that video content information as well as network constraints and users' statistics play a crucial role in selecting proper encoding parameters to provide fairness among users and to reduce network resource usage. Finally a few theoretical guidelines are proposed and can be used, in realistic settings, to choose the encoding parameters based on the user characteristics, the network capacity and the type of video content.

We participated to a work concerning the enforcement of average bitrate through CDN Service Level Agreement (SLA) contracts in [SBP+15] to achieve better user's quality of experience. In this work, WiseReplica is introduce, an adaptive replication scheme for peer-assisted VoD systems that enforces the average bitrate for Internet videos. Using an accurate machine-learned ranking, WiseReplica saves storage and bandwidth from the vast majority of non-popular contents for the most watched videos. Simulations using YouTube traces suggest that this approach meets users expectations efficiently. Compared to caching, WiseReplica reduces the required replication degree for the most-watched videos by two orders of magnitude, and under heavy load, it increases the average bitrate by roughly 85%.

7.2 Perspectives

We describe three main research efforts that should, in our opinion, advance the current state of our proposed solutions for live streaming systems: (i) model extension, (ii) statistical and learning mechanisms, and (iii) middleware integration.

7.2.1 Model Extension

Several models we presented are simple representations of our studied target scenario. We believe that these studies open new perspectives and consequently the need of extensions for the presented models.

For example, our hybrid delivery model, presented in Chapter 5, considers that all viewers have enough bandwidth to download the video content, neglecting its heterogeneity. We could extend the model and include the heterogeneity by using the information provided by DASH sessions data set, as we applied on the model in [APSB15].

7.2.2 Statistical and Learning Mechanisms

The prediction of session popularity is an important aspect in the presented solutions for both delivery (Chapters 4 and 5) and transcoding (Chapter 6). Therefore, enhanced mechanisms could be used to try to predict the popularity evolution before and during the life of a session. More specifically, we could consider two mechanisms: more sophisticate statistical models and machine learning.

Statistical models, such as time series analysis, have been implemented on VoD systems in order to forecast the videos popularity. For example, ARIMA and GARCH time series techniques was studied by [NXLZ12] in order to save bandwidth and storage costs on a VoD system running on the cloud. We believe that more sophisticated strategies are expected to yield better performances, especially for the transcoding at-startup strategies presented at Chapter 6.

Different types of machine learning mechanisms exists. In [BSA+13] they show that a delivery infrastructure that uses their machine learning model to choose CDN and bitrates can achieve more than 20% improvement in overall user quality of experience compared to naive approach. We believe that these mechanisms could improve our results as well, and we discuss more specifically the application of classification and clustering.

Classification learning involves a machine learning from a set of pre-classified (also called pre-labeled) examples, from which it builds a set of classification rules (a model) to classify unseen examples. In our solutions we could classify sessions and broadcasters into popular or not. With this distinction we could for example, in the hybrid delivery in Chapter 5, attribute to the CDN all sessions labeled popular.

Clustering is the grouping of instances that have similar characteristics into clusters, without any prior guidance. Similarly to the classification, for our delivery solutions we would aim to the construction of two groups: popular and unpopular. For the solutions of transcoding we could profit from different popular groups.

7.2.3 Middleware Integration

The solutions conceived in this thesis could be integrated to real systems. Such an integration can be performed by the implementation of middleware. With this implementation we could perform realistic experiments on test beds such as PlanetLab[1], moreover we would provide the community the means to integrate our proposed solutions to their systems. Also, we could benefit

[1] https://www.planet-lab.org/

from the real implementation by validating our simulation models, which have typically considered simplified scenarios.

For example, our strategies proposed to solve the transcoding at Chapter 6 neglect the assignment of transcoding jobs to machines. Would be interesting to make the integration of the Virtual Machine (VM) assignment policy into middleware such as OpenStack.[2]

Another aspect that we could explore with the middleware integration is the impact of VM live migration on the cloud. OpenStack offers three types of live migration: shared storage-based live migration (both VMs have access to shared storage), block live migration (no shared storage is required) and volume-backed live migration (VMs are backed up by hard disks rather than RAM and no shared storage is required).

All these options of live migration provide extreme versatility of management, although it comes at a price of degraded service performance during migration. Such degradation of quality of service when performing a live migration in the cloud has been studied on the scenario of web content delivery [BKR10]. We could improve our work on Chapter 4 by similarly evaluating the live migration at the end of viewer peaks, reducing the total costs. One important difference regarding our scenario is that, contrary to static web content, live video delivery is highly impacted by the migration downtime.

[2]http://www.openstack.org/

Appendix A

Algorithms in Pseudo-code

In this section we give the pseudo-code of all the elements of the transcoding heuristic algorithm proposed in Chapter C.3.

Algorithm 1: Main routine

```
   Data: channelsSet: Channels metadata (e.g. number of viewers, id) sorted by decreasing
         channel popularity.
   Data: totalCPU: total CPU Budget in GHz.
 1 representations ← emptySet()
 2 foreach channel ∈ channelsSet do
 3     cpu ← channel.viewersRatio * totalCPU
 4     cpu ← min(cpu, MAX_CPU)
 5     w_video ← getVideoTypeWeight(channel.video)
 6     w_resol ← getResolutionWeight(channel.resolution)
 7     cpu ← max(cpu * (1 + w_video + w_resol), 0)
 8     representations.append(findReps(channel, cpu))
 9 return representations
```

Algorithm 2: findReps Find the channels representations set meeting a budget

```
   Data: channel: Channel metadata (e.g. number of viewers, id).
   Data: CPU: calculated channel CPU Budget in GHz.
 1 representations ← emptySet()
 2 freeCPU ← CPU
 3 repeat
 4     newRep ← false
 5     foreach resolution ≤ channel.resolution ∈ resolutionsSet do
 6         foreach bitrate ∈ bitratesSet[resolution] do
 7             thisRep ← (resolution, bitrate)
 8             if bitrate ≤ channel.bitrate and thisRep ∉ representations then
 9                 thisRep.cpu ← getCPU(thisRep, channel)
10                 if thisRep.cpu ≤ freeCPU then
11                     reps ← representations + thisRep
12                     thisRep.qoe ← getQoE(reps, channel)
13                     if not newRep or newRep.qoe < thisRep.qoe then
14                         newRep ← thisRep
15     if newRep then
16         representations.append(newRep)
17         freeCPU− = newRep.cpu
18 until not newRep
19 return representations
```

Algorithm 3: getQoE Obtain an estimation of Peak Signal-to-Noise Ratio (PSNR) for a given representation set of one channel

```
   Data: channel: Channel metadata (e.g. number of viewers, id).
   Data: repSet: Set of representations for a given channel.
1  totalPSNR ← 0
2  foreach rep ∈ repSet do
3      ranges ← getResolutionsRanges(rep)
4      foreach viewersRange ∈ ranges do
5          v_ratio ← getViewersRatio(viewersRange)
6          v_resol ← getViewersResolution(viewersRange)
7          partialPSNR ← calcPSNR(rep, channel, v_resol)
8          totalPSNR+ = v_ratio ∗ partialPSNR
9  return totalPSNR
```

Appendix B

Résumé Étendu en Français

Contents

B.1 Introduction

Nous sommes actuellement témoins de l'émergence de deux phénomènes : la popularisation des appareils capables de capturer des vidéos, et l'accroissement de la qualité des réseaux engendrant un nombre toujours croissant d'utilisateurs d'Internet. En effet, de nos jours, chaque ordinateur portable ou même téléphone possède une caméra. Les appareils en résolution haute définition sont devenus accessibles, incluant leur version mobile. Cela implique qu'une large portion de la population a le moyen nécessaire pour faire de la vidéo en direct.

En ce qui concerne les réseaux, nous avons vu dans ces dernières années un afflux d'utilisateurs et une meilleure couverture du réseau. Parallèlement, la qualité du réseau en termes de latence et bande passante a été améliorée de manière significative en particulier avec l'arrivée des réseaux à très haut débit comme la fibre optique. De plus, les frais de connexions sont devenus abordables, et la grande majorité des internautes possèdent un accès illimité. La couverture du réseau mobile est également large et de bonne qualité. La

vitesse de progression de la couverture du réseau mobile (3G/4G) suggère que la plus grande partie de la population aura un accès quasi permanent à un réseau de qualité à un prix raisonnable.

L'accroissement de la qualité du réseau a permis aux utilisateurs de consommer plus de contenu vidéo par internet. Netflix, un service de diffusion VoD (vidéo à la demande), a dépassé 50 millions d'abonnés mensuels au second trimestre de l'année 2014 avec un revenu de 1.34 millions de dollars [Sha14]. Twitch, un service de distribution en direct de vidéo, est devenu la quatrième source génératrice de bande passante internet aux Etats-Unis en Février 2014 et a été racheté par Amazon la même année pour près d'un milliard de dollars [FW14]. En 2013 les contenus vidéos représentaient plus de la moitié du trafic internet [Inc14].

La combinaison de ces phénomènes et la tendance des utilisateurs à exposer des portions de leur vie a conduit à l'augmentation de la popularité des services de diffusion de vidéo en direct. Dans cette thèse nous étudions de nombreux défis créés par ces systèmes. Les services de diffusion de vidéos en direct sont décrits dans la section suivante. Ensuite, nous présentons nos contributions dans ce domaine. Enfin, nous conclurons ce résumé.

B.2 Service de diffusion directe de vidéo en ligne

De nombreux acteurs sont impliqués dans les UGC (contenus générés par l'utilisateur) diffusés en direct. Cet environnement est illustré par la Figure B.1. Les services de diffusion en direct sont constitués non seulement d'utilisateurs qui consomment et produisent du contenu, mais également des fournisseurs de services et leurs procédés de conversion/livraison du contenu à grandes échelles. Nous décrivons ci-dessous les aspects importants des systèmes de diffusion en direct.

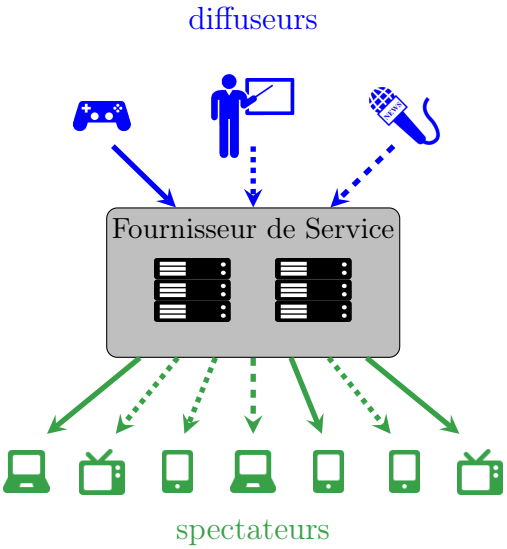


Figure B.1: L'architecture simplifié de diffusion de vidéos en direct

User-Generated Content (UGC) Les UGC correspondent aux contenus crées par un utilisateur, que ce soitent des blogs, des tweets, des images, des vidéos, etc. Ces contenus sont souvent ensuite mis à disposition sur internet par le biais de type réseaux sociaux. Les UGC sont à l'origine de nouveaux phénomènes émergeants comme le journalisme civil, où les civils capturent et font des rapports sur les actualités locales avec leurs appareils connectés. Les UGC sont au cœur de plusieurs grands services Internet tel que Youtube[1] et Wikipedia[2]. Ce contenu est en général créé en dehors de tout cadre professionnelle.

Diffuseur Les diffuseurs sont les utilisateurs responsables de créer des UGC et de les diffuser en direct sur Internet. Normalement ces utilisateurs sont authentifiés sur la plateforme par laquelle ils diffusent leurs contenus et possèdent une « chaine » qui leur est propre.

Session Une session est un événement en direct émis par un diffuseur. Dans une plateforme de diffusion de direct en ligne, une chaine peut être en ligne ou hors ligne. Chaque période durant laquelle la chaine est restée en ligne peut être archivée ultérieurement. Dans nos travaux, nous ne nous intéressons uniquement aux sessions en direct.

Spectateur Un spectateur est un utilisateur qui n'est pas forcément enregistré sur le service et va regarder les sessions créées par les diffuseurs.

[1]https://www.youtube.com/
[2]https://www.wikipedia.org/

Le nombre d'utilisateurs regardant une session peut évoluer au cours du temps. Le nombre total de spectateurs d'une chaine à un instant donné est appelé la *popularité de la chaine*. La figure B.2 montre l'évolution de la popularité d'une chaine sur une durée. Cette chaine contient deux sessions.

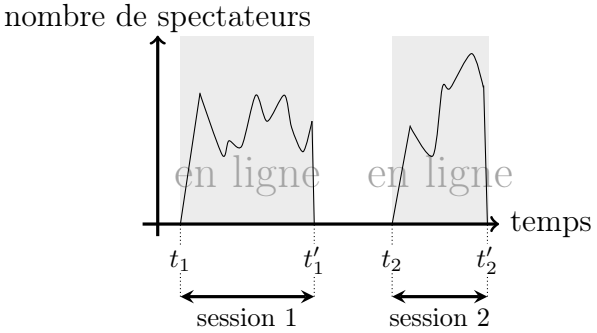


Figure B.2: La popularité d'une chaine avec deux sessions en direct pendant une période de temps.

Service Provider Le « service provider » (fournisseur de service) est la plateforme sur laquelle le diffuseur peut mettre en ligne son contenu UGC. Le fournisseur de service est ensuite responsable du processus de livraison du contenu aux utilisateurs et plus particulièrement aux spectateurs. En règle générale les fournisseurs de services possèdent un moyen de monétiser leur plateforme, souvent sous forme de publicités présentes au début de session.

CDN Un CDN (réseau de distribution de contenu) est un immense système distribué de serveurs disposés un peu partout dans la toile. Ces serveurs sont normalement déployés dans les bordures d'Internet, chez les fournisseurs d'accès (ISP) et proches des utilisateurs. Le but d'un CDN est de délivrer du contenu aux utilisateurs tout en assurant une forte disponibilité et performance. Les fournisseurs de services payent les CDN pour délivrer leurs contenus aux utilisateurs. Le CDN paye ensuite les ISP pour héberger des serveurs dans leur centres de données.

Adaptive Bit Rate (ABR) Le débit adaptatif des flux en direct est une technique employée à la diffusion de la vidéo. En utilisant l'ABR la vidéo diffusée est encodée en différentes versions chacune à un débit différent. Les versions sont nommées représentations. Un spectateur peut ensuite utiliser son lecteur de contenu pour choisir la qualité souhaitée en fonction de son appareil et de la capacité de sa bande passante. Plus précisément, les implémentations actuelles, comme DASH (un standard

de conversion créée par MPEG) et HLS (implentation faite par Apple) , fonctionnent normalement au dessus d'Hypertext Transfer Protocol (HTTP). Les débits multiples sont segmentés en plusieurs parties, typiquement de deux à dix secondes. Un résumé contient l'information nécessaire concernant les différents débits et segments disponibles. Au début, le client demande les segments de débits les plus faibles. Si le client possède suffisamment de bande passante pour télécharger un segment plus volumineux, son lecteur va alors faire une requête à ce segment dès que possible. Si les conditions se détériorent ensuite, le lecteur vidéo fera la requête au segment de plus faible débit. ABR permet ainsi de limiter les temps de mise en tampon et d'améliorer la QoE (qualité de l'expérience) des utilisateurs.

Encodage L'encodage est la tâche qui transforme une source vidéo d'un format donné vers une représentation différente qui sera une version alternative de la vidéo possédant une résolution et/ou un débit différent. L'encodage est un procédé essentiel à l'ABR.

La manière d'absorber la charge imposée par la transmission de ces sessions diffère entre les fournisseurs de services. Pour la diffusion en direct, il y a en général deux types de processus.

Processus traditionnel Il consiste à préparer le flux direct brut (par exemple pour vérifier la cohérence et une meilleure intégration à la page web) et ensuite le délivrer directement aux spectateurs. La figure B.3 illustre ce processus. La vidéo brute est produite par le diffuseur et est envoyée au système par les serveurs. Le système est ensuite chargé de délivrer le contenu de cette vidéo à des spectateurs hétérogènes (avec des appareils et des conditions d'accès différentes).

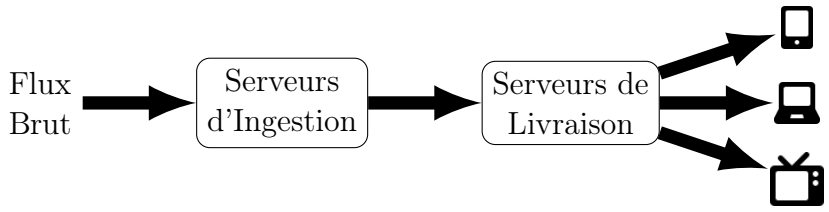


Figure B.3: Processus traditionnel de livraison de flux video en direct.

Processus d'encodage Illustré par la Figure B.4, le processus d'encodage consiste à transformer le flux brut dans de multiples flux vidéo en utilisant ABR pour la livraison des sessions aux spectateurs. Cette livraison

ce fera en utilisant une technologie telle que DASH [Sto11]. Pendant le processus d'encodage le système est capable de fournir différentes versions du flux brut à chaque spectateur, en fonction de la taille de leur écran ou de la capacité de leur bande passante.

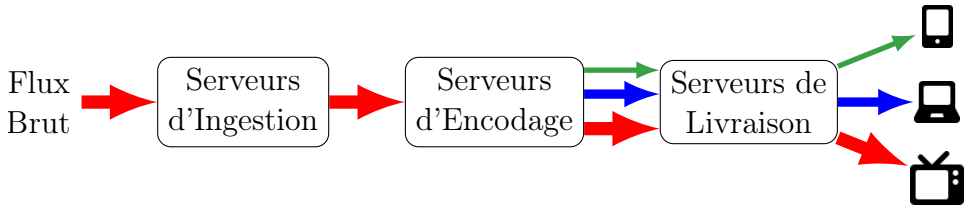


Figure B.4: Processus d'encodage et livraison de flux video en direct.

B.3 Contributions

Les contributions principales de cette thèse sont les suivantes : (*i*) la collecte et l'analyse d'un ensemble de données de flux vidéo en direct ; (*ii*) la définition de nouvelles approches pour le distribution du contenu en utilisant le cloud ; (*iii*) des solutions hybrides de distribution de contenu ; (*iv*) une analyse de la technique de débit adaptatif de flux (ABR) ; (*v*) une application de l'ensemble de données à la distribution par le cloud et l'encodage.

B.3.1 L'ensemble de données des sessions en direct

Premièrement nous avons rassemblé un ensemble de données de sessions en direct. Ces traces ont ensuite étaient utilisées pour évaluer nos solutions. Ainsi, notre première contribution la collecte et l'analyse de cet ensemble de données, contenant trois mois de traces de deux services majeurs d'UGC en ligne. Nous avons rendu disponible en ligne cette trace gratuitement pour la communauté qui contient des millions de sessions directes et des centaines de milliers de diffuseurs. À notre connaissance ce travail est le premier à fournir non seulement une telle quantité de données mais également la comparaison et la compréhension des deux services majeurs de diffusion en ligne YouTube et Twitch. Nous apportons par notre étude des éléments précieux de compréhension de ce que peuvent être ces services à grande échelle. Nous montrons qu' 1 Tbps de bande passante est souvent atteint de manière à délivrer ce type de contenu. Nous montrons également les différences entre YouTube et Twitch , particulièrement en ce qui concerne les habitudes de leurs diffuseurs. Les diffuseurs sur Twitch sont plus engagés sur la plateforme pour produire des contenu en direct que ceux de YouTube. Nous avons détecté de nombreuses

caractéristiques qui indiquent la maturité de l'environnement Twitch. Comparé à un service VoD , les diffuseur des services en ligne en direct sont plus actifs. À la différence des autres systèmes, la popularité des contenu au sein de ces systèmes est très hétérogène. Ces informations sont utilisés dans nos simulations pour fournir un environnement proche du réel. En rendant l'ensemble des données publiquement disponible nous espérons enrichir la communauté et aider à améliorer les plateformes de diffusion de contenu en direct.

B.3.2 Livraison par le nuage

Cette solution est basée sur des systèmes comme le nuage ou des machines virtuelles distribuées et utilise les prédictions de popularité pour héberger les flux sur les serveurs. Les services de distributions de flux en direct d'UGC doivent gérer un grand nombre de sessions simultanées qu'il faut collecter puis distribuer à un nombre encore plus grand d'utilisateurs. Nous montrons qu'en prenant en compte la popularité future des sessions nous pouvons allouer les flux plus efficacement sur les serveurs. Cela faut également réduire le partitionnement des flux entre les différents serveurs et limiter ainsi les transferts intra-serveurs qui génèrent au final une perte de bande passante. Nous basons notre étude sur les ensembles de données précédemment présentés. Nous nous intéressons particulièrement au nombre de serveurs nécessaire pour la diffusion les sessions ainsi qu'au nombre d'échanges intra-serveurs. Notre approche, POPS, utilise une estimation du pic de popularité des sessions au moment de les placer dans les serveurs de diffusion.

B.3.3 Livraison hybride

Cette solution est basée sur une livraison assistée mettant en pratique non seulement les serveurs privés de la compagnie de diffusion mais également les CDN. Nous avons en premier formalisé la livraison des chaines de flux en direct. Ensuite, nous analysons ce modèle formel et comparons quatre stratégies intuitives à l'optimal théorique. Les résultats de notre étude montrent un besoin d'explorer et de trouver des solutions intelligentes pour la livraison des sessions vidéo.

B.3.4 Diffusion de flux avec ABR

Nous avons conçu deux stratégies pour décider quelles chaines devraient être délivrées par ABR. C'est un travail préliminaire qui vise une intégration dans le contexte d'un service multimédia à grande échelle. Nous introduisons un problème lié à la gestion de l'infrastructure et nous proposons des stratégies

simples qui peuvent facilement réduire les couts globaux de l'infrastructure tout en améliorant la QoE des utilisateurs.

B.3.5 Applications à l'ensemble de données collectées

L'ensemble de données que nous avons collecté; nous offre l'opportunité de pouvoir introduire des traces réelles dans nos simulations. Nous décrivons deux travaux en collaboration, dans lesquels nous avons appliqué cet ensemble.

En premier, nous discutons des aspects économiques liés à la distribution basée sur les CDN. Nous proposons un modèle pour analyser la politique qu'un CDN reposant sur le profit. Ce modèle s'est montré particulièrement efficace. À notre connaissance, nous sommes les premiers à modéliser un CDN d'un point de vue économique avec pour ambition de comprendre l'impact qu'un CDN peut avoir sur le marché de la distribution.

Dans un deuxième temps, nous présentons une optimisation linéaire en nombres entiers pour maximiser la qualité moyenne que les utilisateurs obtiendront et proposons une heuristique visant à obtenir un résultat analogue tout en permettant de passer à l'échelle en nombre d'utilisateurs. Nous étudions la gestion de nouveaux services de diffusion de flux en direct dans le nuage en considérant que les fournisseurs utilisent le nuage comme plateforme de calcul. Toutes les simulations que nous avons conduites utilisent des données réelles issues des trois acteurs majeurs dans ce domaine. L'étude se concentre sur les interactions entre les paramètres d'encodage optimaux, l'utilisation CPU et la qualité perçue par les utilisateurs. Nous utilisons une Programmation Linéaire en Nombres Entiers (PLNE) pour modéliser ce système et nous comparons cette solution optimale aux solutions actuelles de l'industrie, mettant en évidence l'écart entre les deux. Dû au temps de calcul du PLNE, nous proposons un algorithme alternatif pour résoudre des problèmes de taille réelle. Cet algorithme trouve des représentations capables d'améliorer les approches standards aussi bien en termes de qualité que d'usage CPU. En plus, il utilise un taux de ressources quasi constant même en présence de variance de la demande.

B.3.6 Contributions additionnelles

En plus de nos contributions principales, présentées dans cette thèse et dans les appendices, nous avons également collaboré dans des contributions visant deux autres grands challenges de la diffusion en direct.

Nous avons utilisé l'ensemble de sessions ABR, présenté précédemment dans cette thèse, pour évaluer la solution proposée dans [TAP+14]. Dans un premier temps nous avons formulé un PLNE maximisant la satisfaction moyenne des utilisateurs, prenant en compte les dynamiques réseaux, les types

de vidéos, et la population visée. La solution de l'optimisation est un ensemble de représentations correspondant à la maximisation de la satisfaction des utilisateurs. Nous avons ensuite évalué cette solution en simulant de multiples sessions ABR reprenant les statistiques réseaux réelles. Les résultats ont montré que la solution proposée est meilleure que les recommandations actuelles des fournisseurs de services aussi bien pour la qualité moyenne, le partage équitable des ressources et la résistance aux éventuelles pannes. La simulation a montré que les informations apportées par les vidéos et les statistiques concernant les utilisateurs sont des paramètres cruciaux à prendre à compte pour choisir les paramètres d'encodage. Ceci permet d'offrir non seulement une équité entre utilisateurs mais également de réduire le cout total en bande passante. Enfin, des instructions théoriques sont données et peuvent être utilisées dans des conditions réelles afin de choisir les paramètres d'encodage basés sur la capacité réseau et le type de contenu vidéo.

Nous avons participé dans [SBP+15] à la discussion à propos du renforcement des Service-Level Agreements (SLAs) en utilisant un accord avec un CDN afin d'améliorer la qualité perçue par l'utilisateur. Nous avons introduit WiseReplica, un système de réplication SLA pour des systèmes VoD assistés par les pairs du réseau. Utilisant un algorithme d'apprentissage, WiseReplica économise de l'espace de stockage mais également de la bande passante pour la grande majorité des contenus non populaires et l'utilise ensuite pour les vidéos les plus regardées. Les simulations utilisant les traces de YouTube suggèrent que cette approche satisfait les utilisateurs de manière efficace. Comparé à un système de cache, WiseReplica réduit le degré de réplication pour les vidéos les plus regardées de deux ordres de grandeur et sous forte charge WiseReplica permet une augmentation du débit moyen obtenu de 85%.

B.4 Conclusion

Dans cette thèse nous avons exploré des solutions pour améliorer les services de diffusion de flux vidéo en direct. La popularisation d'appareils capables de capturer des vidéos, les efforts continus pour améliorer la qualité des réseaux et le succès des services en ligne de direct ont conduit ces derniers à être responsables d'une grande partie du trafic internet. Nos objectifs sont la diminution des ressources utilisées, du temps de calcul et de la bande passante afin d'améliorer l'expérience des utilisateurs.

Nous avons exploré et développé des solutions pour délivrer des contenus massifs produits par ces plateformes. Un des défis est la grande variation du nombre total de spectateurs et la grande hétérogénéité de la popularité des flux, qui implique généralement un surprovisionnement des ressources. Dans

cette thèse nous montrons qu'il y a un équilibre entre le nombre de serveurs employés pour diffuser le contenu et la bande passante utilisée pour diffuser le contenu aux utilisateurs pour chaque serveur. Nous mettons aussi l'accent sur l'importance de prédire la popularité des flux afin de placer ces derniers de manière efficace sur les serveurs. Nous proposons trois solutions qui abordent les différents aspects concernant la livraison des sessions en direct.

Nous avons également étudié les difficultés concernant l'encodage des flux. L'encodage des flux est gourmand en temps de calcul. Nous montrons qu'un codage adaptatif (ABR) est capable de réduire le cout en bande passante et d'améliorer la qualité pour les utilisateurs en échange d'un cout en termes de CPU pour l'encodage. Pour étudier les compromis entre bénéfices et couts, nous avons formulé en réponse deux PLNE.

Appendix C

Live Sessions Data Set Applications

Contents

C.1 Introduction

In Chapter 3 we presented a live session data set. This data set open opportunities to introduce real traces on simulations regarding live streaming systems. In this appendix we describe two collaboration works, which we applied the live session data set.

C.2 CDN Fairness on Live Delivery

C.2.1 Introduction

The delivery model we explored in this section is entirely formulated with CDN, as first introduced at Chapter 2. The term CDN refers to both an *infrastructure* designed to deliver content at large scale over an underlying network, and the *economic actor* providing that service. Previously we discussed the infrastructure aspect but in this section we also include the economic aspect.

CDN have a huge economic weight (the annual revenues of Akamai, the CDN leading company, are over two billion dollars), and a growing impact on the Internet ecosystem: *i)* CDN activities affect the traffic exchanged between network providers, and consequently their economic relationships [Kov12, Sau12]; *ii)* on many aspects (per-volume charging, connectivity service) CDN actors compete with transit providers, which explains why some major transit network operators such as Level 3 have shifted a fraction of their activities to CDN; and *iii)* other actors in the value chain of content delivery have started developing a CDN activity, including ISPs, content providers, and equipment vendors [Bon10, Sch13]. This fast-moving and business-driven environment exacerbates the concerns among user and regulation communities regarding service quality and economic fairness, epitomized by the net neutrality debate [Cro11, LME06, MRT12, Wu03].

The scientific literature provides models and analyses of the interactions between content providers and ISPs in order to address network neutrality, and sometimes to propose regulation remedies [ALX11, CMT12, CMT13, Fri07, NOSMW10], but the role of CDNs is barely mentioned. To the best of our knowledge, the only official report mentioning CDNs is from the Norwegian regulator [Sor], where it is stated that "*the ordinary use of CDN servers is not a breach of net neutrality*". In this section, we show that CDNs that are "normally" managed (i.e., by rational actors) can nevertheless lead to differences in the perceived Quality of Service (QoS) among content providers, which goes against neutrality principles.

More generally, the performance analysis community has barely considered the economics of CDN actors so far. Among the few notable works, we can mention [HKSC04, Hos08, HCKS08] where the authors consider a single CDN over a time period of interest. The best pricing strategy is studied, but the complex relationships between actors, and their consequences on fairness and social welfare are ignored.

We focus on the management problems faced by a CDN having to dimension and optimally use its infrastructure, sharing it among its clients (content/service providers) so as to maximize its revenue. We propose a model

to analyze the behavior of a profit-maximizing CDN, and assess the impact of a CDN policy on the quality perceived by users and on the fairness among content providers. We illustrate these theoretical results with an analysis based on our real data from Twitch and YouTube Live, presented at Chapter 3, which we artificially make compete for the resources of a CDN. We show that a CDN implementing a revenue-maximizing policy tends to favor incumbent content providers, but at an extent that is not dramatic, even if the said incumbent tries to take advantage of the profit-driven CDN policy by over-paying for a better service.

The work presented in this section has been published on CNSM 2014 [MPST14]. While I participate to the discussion with the co-authors for the mathematical model and theoretical results, my main contribution is the application of the model to the real data with the traces described at Chapter 3 and the analysis of service providers competition for the CDN.

This section is organized as follows. Next the mathematical model is presented. Further, we describe a theoretical analysis of the model. Then we have an application of the model to data from our real traces. Lastly, we conclude this section.

C.2.2 Model

A CDN is a *multi-tenant* infrastructure: its resources are shared among multiple Service Providers (SPs). For simplicity reasons, we consider here two SPs (referred to as SP1 and SP2) but the model can be extended to more SPs. We depict the configuration/topology in Figure C.1.

We distinguish two classes of CDN resources: some privileged resources that are located close to the clients in the ISP, and the remaining resources, which often correspond to the origin data centers. Since the resources that are the most often offered by CDN are storage, we will hereafter call *cache* the privileged resources in the ISP and we will use other wording related to storage management. Note however that the services that are offered by today's CDNs extend to other types of resources, typically computing. Figure C.1 represents only one ISP, but multiple ISPs can be considered, each one being studied independently.

The economic flows involving the CDN, depicted at Figure C.2, are as follows:

Revenues. Each SP subscribes to the CDN service to reach its customers. The CDN charges the SPs a different price per unit of data volume delivered to users, according to whether users are served from the cache

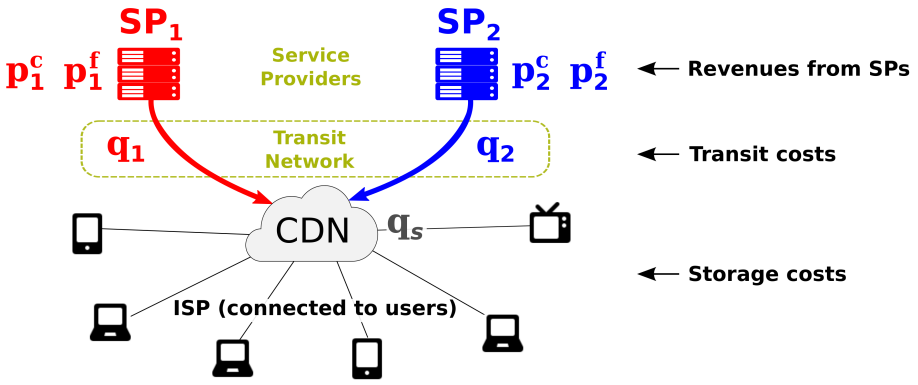


Figure C.1: Costs and revenues for a CDN located within an ISP's network.

server (unit price p_i^c for SPi, $i = 1, 2$) or from the SP origin data center, hence with lower QoS (unit price $p_i^f < p_i^c$ for SPi).

Costs. The CDN is responsible for sending the data to users (those covered by the considered ISP). There are two cases. If the data are taken from the origin data center of SPi, the cost is the transit cost q_i per unit of volume for the CDN (which can be low if the CDN owns the transit network, but large otherwise). Remark that those transit costs can differ among SPs (i.e., $q_1 \neq q_2$) since the path to reach the ISP of interest may differ. The second case is when the data are delivered by the CDN cache, the quality experienced by users is better, and no transit costs are incurred. On the other hand, storage in the CDN cache incurs a unit cost q_s.

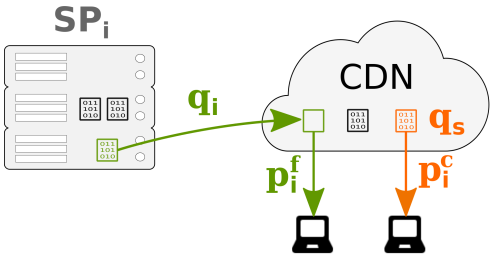


Figure C.2: Economic flows involving the CDN. A flow that the requested data is in the CDN (cache hit) will have storage cost q_s and delivery revenue p_i^c. A flow that the requested data is not cached in the CDN (cache miss) will have transit cost SPi $\rightarrow$ CDN q_i and delivery revenue p_i^f.

If we consider those prices fixed (from long-term contracts), the decision variables of the CDN, illustrated at Figure C.3, are:

- The **capacity** C of the server in the ISP.
- The implemented **caching strategy**, i.e., the management of the storage space in the cache. With two service providers, the only decision variable for the CDN regards the choice of whose content to favor in the cache, summarized by the volume $C_1 \leq C$ of cached SP1 content (the volume of SP2 content cached being $C_2 = C - C_1$).

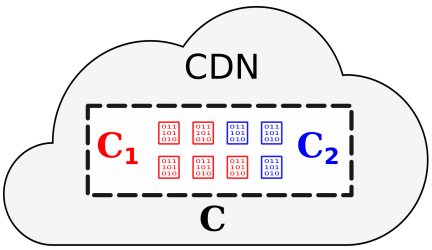


Figure C.3: Decision variables of the CDN. The capacity **C** is the CDN cache storage capacity in the ISP. The caching strategy, involving C_1 and **C** (C_2 = **C** – C_1), manages the storage space in the cache.

We do not deal with the extensive literature on the subject that involves time variations of the download frequency of content items. We rather consider a static problem, with content popularity values as *constant and known to the CDN operator*. Let us denote by $F_i(x), i = 1, 2$ the minimum download frequency (number of requests per time unit) for the x most popular units of content of SPi, and assume that F_i is continuous and strictly decreasing. Knowing the popularity values, the CDN stores the content of each provider that yields the largest revenues, which may result in an "unfair" strategy with respect to SPs.

The *incomes* of the CDN from SP payments equal

$$\begin{aligned} & \sum_{i=1}^{2} \left(p_i^c \int_{x=0}^{C_i} F_i(x)\mathrm{d}x + p_i^f \int_{x=C_i}^{V_i} F_i(x)\mathrm{d}x \right) \\ = & \sum_{i=1}^{2} \left(p_i^f \bar{G}_i + (p_i^c - p_i^f) G_i(C_i) \right) \end{aligned}$$

where V_i is the total volume of content proposed by SPi ($i = 1, 2$), $G_i(y) := \int_{x=0}^{y} F_i(x)\mathrm{d}x$ is the cumulated user download throughput from requests for the volume y of the most popular content from SPi, and $\bar{G}_i = G_i(V_i)$ is the total user download throughput of SPi content. Without loss of generality, we

ignore content with no demand, so that we can assume $F_i(x) > 0$ for all $x < V_i$ (i=1,2).

Storage costs equal $q_s C$. We assume $C \leq \min(V_1, V_2)$, i.e., the CDN cannot cache all the content from any SP.

For the *transit costs*, we neglect the one-shot costs for the content stored in the CDN cache: therefore transit costs only correspond to content that is not in the cache, and for each SP they are proportional to the aggregated download rate for that content. Since $C_2 = C - C_1$, the total transit costs equal

$$q_1(\bar{G}_1 - G_1(C_1)) + q_2(\bar{G}_2 - G_2(C - C_1)).$$

Overall, the net revenue of the CDN per time unit is

$$R(C, C_1) = \sum_{i=1}^{2} \left(r_i^f \bar{G}_i + (p_i^c - r_i^f) G_i(C_i) \right) - q_s C \tag{C.1}$$

with $r_i^f := p_i^f - q_i$ for $i = 1, 2$. We limit ourselves to $r_i^f > 0 \ \forall i$ to ensure that the CDN makes some non-negative revenue.

Finally, let us denote the quality experienced by users by Q_c for the cached content, and by $Q_{f,i} < Q_c$ for content retrieved from SPi. The average user experienced quality is then

$$Q^{\text{tot}} = \frac{1}{\bar{G}_1 + \bar{G}_2} \sum_{i=1}^{2} \left(Q_c(G_i(C_i) + Q_{f,1}(\bar{G}_i - G_i(C_i)) \right).$$

C.2.3 Maximizing the CDN revenue

In our analysis, we first focus on the caching strategy, i.e., determine the revenue-maximizing sharing of the storage space C (treated as fixed) among SP1 and SP2 content. Then we discuss the optimal value of the cache capacity C.

C.2.3.1 Whose content to cache?

We assume the total storage capacity C is fixed, and look for the best caching strategy decision (the value C_1^{opt} of C_1 maximizing the net revenue in (C.1), where $C_2 = C - C_1$).

By construction, each function G_i $(i = 1, 2)$ is continuously differentiable (with derivative F_i), strictly increasing and strictly concave on $[0, V_i]$, hence $R(C, C_1)$ is a strictly concave function of C_1 for C fixed. The first-order optimality condition is thus sufficient, and the optimal C_1 equals:

- C if $(p_1^c - r_1^f)F_1(C) \geq (p_2^c - r_2^f)F_2(0)$
- 0 if $(p_1^c - r_1^f)F_1(0) \leq (p_2^c - r_2^f)F_2(C)$
- the unique solution in $(0, C)$ of

$$\frac{F_1(x)}{F_2(C-x)} = \frac{p_2^c - r_2^f}{p_1^c - r_1^f} \qquad \text{otherwise.} \tag{C.2}$$

It therefore exists and is unique. Remark that for given popularity distributions, the optimal C_1 then only depends on the ratio $\frac{p_2^c - r_2^f}{p_1^c - r_1^f} = \frac{p_2^c + q_2 - p_2^f}{p_1^c + q_1 - p_1^f}$. Due to the decreasingness in x of the left-hand term in (C.2), the optimal C_1 decreases with the value of that ratio.

Notice also that when prices are fixed, the solution of (C.2) strictly increases with C: take $\tilde{C} > C$, the corresponding optimal values C_1^{opt} and $\tilde{C}_1^{\text{opt}}$ of C_1 must satisfy

$$F_1(C_1^{\text{opt}})F_2(\tilde{C} - \tilde{C}_1^{\text{opt}}) = F_1(\tilde{C}_1^{\text{opt}})F_2(C - C_1^{\text{opt}}). \tag{C.3}$$

Assuming $\tilde{C}_1^{\text{opt}} \leq C_1^{\text{opt}}$ leads to $F_1(C_1^{\text{opt}}) \leq F_1(\tilde{C}_1^{\text{opt}})$ and $F_2(\tilde{C} - \tilde{C}_1^{\text{opt}}) < F_2(C - C_1^{\text{opt}})$, contradicting (C.3).

Example. Following the literature on content popularity [AH02], let us consider a Zipf or power-law distribution of the request rates among pieces of content: $F_i(x) = A_i x^{-\alpha}$ with $\alpha > 0$, for $x > x_{i,\min} > 0$. The values $x_{i,\min}$ indicate the domain of validity of the power law, and we assume they are small enough, so that $\frac{F_2(C-x_{1,\min})}{F_1(x_{1,\min})} < \frac{p_1^c - r_1^f}{p_2^c - r_2^f} < \frac{F_2(x_{2,\min})}{F_1(C-x_{2,\min})}$, and thus the optimal value of C_1 is in $(x_{1,\min}, C - x_{2,\min})$. Solving (C.2) then leads to C_1 being the solution x of

$$\frac{x}{C-x} = \left(\frac{A_1}{A_2}\frac{p_1^c + q_1 - p_1^f}{p_2^c + q_2 - p_2^f}\right)^{1/\alpha}, \quad \text{which gives}$$

$$C_1^{\text{opt}} = \frac{C}{1 + \left(\frac{A_2}{A_1}\frac{p_2^c + q_2 - p_2^f}{p_1^c + q_1 - p_1^f}\right)^{1/\alpha}}. \tag{C.4}$$

C.2.3.2 Dimensioning the cache

We can similarly determine the optimal storage capacity C for the CDN, by differentiating $R(C, C_1^{\text{opt}})$ in terms of C, with C_1^{opt} a function of C. Rewriting the conditions not to end up with (C.2):

$$F_1(C) \geq kF_2(0) \quad \text{or} \quad F_2(C) \geq \frac{1}{k}F_1(0)$$

with $k = (p_2^c - r_2^f)/(p_1^c - r_1^f)$, we remark that none is satisfied (since the F_i are decreasing functions) when

$$C > \max(F_2^{-1}(kF_1(0)), F_1^{-1}(F_2(0)/k)), \tag{C.5}$$

in which case the solution of C_1^{opt} is inside $(0, C)$. For C smaller, it may happen that the optimal value C_1^{opt} is 0 or C. Assuming (C.5), the envelope theorem yields

$$\begin{aligned} \frac{\partial R(C, C_1^{\text{opt}}(C))}{\partial C} &= (p_2^c - r_2^f)F_2(C - C_1^{\text{opt}}(C)) - q_s \\ &= (p_1^c - r_1^f)F_1(C_1^{\text{opt}}(C)) - q_s, \end{aligned}$$

where the last equality comes from (C.2). From that last expression and due to the strict increasingness of C_1^{opt} in C, the revenue is a strictly concave function of C for C sufficiently large. From (C.1), it is also strictly concave when $C_1^{\text{opt}} \in \{0, C\}$. Since $R(C, C_1^{\text{opt}}(C))$ is differentiable for all C (its "interior" derivative when $C_1^{\text{opt}}(C)$ tends to 0 or C being equal to the derivative with a fixed C_1^{opt}), it is then strictly concave over the whole interval $[0, C]$.

Hence, since the derivative of R gets negative for C sufficiently large, there exists a unique cache capacity C maximizing revenue (that is strictly positive if q_s is not too large, *i.e.*, if $q_s < \max_{i=1,2}(p_i^c - r_i^f)F_i(0)$).

Example. Considering again the case of power-law distributions, the above derivative is

$$(p_2^c - r_2^f)A_2 \left(C - \frac{C}{1 + \left(\frac{A_2}{A_1}\frac{p_2^c - r_2^f}{p_1^c - r_1^f}\right)^{1/\alpha}} \right)^{-\alpha} - q_s,$$

which gives the optimal dimensioning of the storage space

$$C = \frac{((p_1^c - r_1^f)A_1)^{1/\alpha} + ((p_2^c - r_2^f)A_2)^{1/\alpha}}{q_s^{1/\alpha}}.$$

C.2.4 Analysis

We restrict the numerical analysis of our model to one situation closely linked to the problems of fairness (and neutrality) of CDNs, where we apply our model to popularity distributions obtained from real traces, and consider two SPs competing for one CDN.

We study two *user-generated live video aggregators*. These service providers offer a service such that anybody can become a *broadcaster*, who

uploads a video session to the aggregator, which is then in charge of preparing and delivering the video to a potentially wide population. Two main players compete: (*i*) an *incumbent*, namely *Twitch*, which has been a well-established service for years, with a stable population of engaged broadcasters (more than five thousands simultaneously broadcasting at any time), and (*ii*) a *challenger*, namely YouTube Live, which has released this feature to the all regular YouTube users in December 2014. Recall that the population of broadcasters of YouTube Live is one order of magnitude smaller than Twitch but, at peak hours, both services have approximately the same population of viewers. In the following, SP2 refers to Twitch while SP1 is YouTube Live. For both services, we consider the traces of the activities described at Chapter 3. In the following, we study one randomly chosen date and we abusively consider that both services use the same CDN to deliver their live streams (for such service, the resources that the CDN offers are transcoding and delivering in the access network).

Our goal is to highlight the role of CDN in three representative scenarios: (*i*) both service providers pay the same price for the CDN service. The transit costs are the same. In this *regular* scenario, the main question is whether the dominance of the incumbent prevents the growth of the challenger. (*ii*) the incumbent player deploys an aggressive strategy where it pays ten times what its competitors pays for the CDN service. It is one of the most critical question in the net neutrality debate: can a well-established player prevents one competitor from growing? Finally (*iii*) the challenger is now the one that is aggressive. Regarding the parameters adopted in the scenarios, we extracted from the traces the information about the videos, meaning V_i, G_i and $\bar{G}_i$. Based on CDNs and Amazon pricing[1], the remaining parameters are: $p_i^f = 0.005$, $p_i^c = 0.5$, $q_s = 0.0000053$, $q_i = 0.94$, $Q_c = 2$, $Q_{f,i} = 0.5$ and $C = 50$. For scenarios (*ii*) and (*iii*), we defined $p_1^c = 5$ and $p_2^c = 5$ respectively.

We show in Figure C.4 the QoE of end-users from both service providers with regard to the evolution of the ratio of the cache that is filled with SP1 content (recall that SP1 is the challenger YouTube). The QoE of SP1 users is represented by Q_1 (black line), while users on SP2 by Q_2 (gray line). We show with thin vertical lines the optimal values of $\frac{C_1}{C}$ for the three considered scenarios.

Our main observation is that, due to the heterogeneity of video popularity, the impact of aggressive strategies is limited in all cases. By choosing to maximize its revenues, the CDN serves more content from SP2 in the regular scenario, which in turn leads to a better overall QoE for users of SP2. But the QoE remains excellent for SP1 as well (more than 0.9 of the best possible).

[1] http://is.gd/CArKkn, http://aws.amazon.com/s3/pricing/

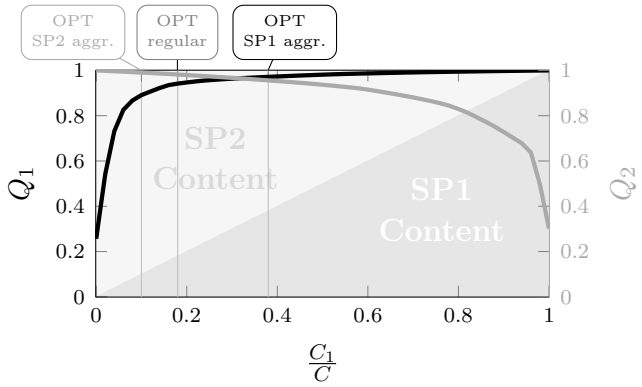


Figure C.4: Normalized quality of experience for both service providers SP1 and SP2 according to the ratio of the cache filled with content from SP1

More interestingly, both aggressive policies are not worth the price. In both cases, the CDN adjusts the ratio $\frac{C_1}{C}$ accordingly to maximize its revenues, but in both cases, the overall QoE of the competitor is not significantly affected. Even when the incumbent player pays ten times the price paid by the challenger, the users of the latter service have a QoE which is more than 0.8 of the best possible.

C.2.5 Conclusion

We propose in this section a model to analyze of the policy that a profit-driven CDN should implement when delivering live video content. This model is especially significant with regard to the multiple recent debates about network neutrality. To the best of our knowledge, this is the first attempt to model CDN from an economic standpoint with the ambition to understand the impact of CDN on the content delivery market.

This study opens perspectives. Fairness can be further analyzed thanks to our model. We would also like to study more generic versions of this model with multiple ISPs, players and resources within the CDN. The competition among players, and ways to regulate it toward the benefit of the whole population, are among the very first studies that we envision. We would also like to integrate more complex monetary agreements between service providers and CDN, including Service Level Agreement (SLA). In addition, we plan to significantly extend the study based on real data that is introduced in Section C.2.4.

C.3 Transcoding Live Adaptive Video Streams in the Cloud

C.3.1 Introduction

The management of live video services is a complex task due to the demand for specialized resources and to real-time constraints. In the previous chapter we concluded that the optimization problem requires a formal definition and analysis. Previously we neglect many details in our formulation in order to provide the global picture. However a more formal and accurate formulation should include a more precise estimation of the QoE gain for the degraded viewers, a better model for the transcoding computing needs, and the management of different hardware computing resources.

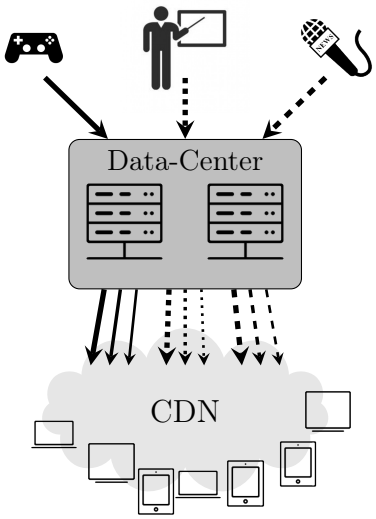


Figure C.5: Live streaming in the cloud

To guarantee the QoE for end-users, *live streaming service providers* (*e.g.*, TV operators and multimedia broadcasters) have traditionally relied on private data centers (with dedicated hardware) and private networks. The widespread availability of *cloud computing platforms*, with ever decreasing prices, has changed the landscape [Red14]. Significant economies of scale can be obtained by using standard hardware, VM, and shared resources in large data centers. As illustrated in Figure C.5, live streaming providers use these services in combination with widely available CDN to build an elastic and scalable platform that can adapt itself to the dynamics of viewer demand. The only condition is to be able to use the standardized cloud computing platforms to prepare the video for delivery.

The emergence of cloud computing platforms has enabled some new trends, including: (*i*) the adoption of ABR streaming technologies to address the heterogeneity of end-users. ABR streaming requires encoding multiple video representations, and thus increases the demand for hardware resources. Modern cloud computing platforms can meet this demand. And (*ii*) the growing diversity of live video streams to deliver. The popularity of services like Twitch [DiP14] illustrates the emergence of new forms of live streaming services, where the video stream to be delivered comes from non-professional sources (*e.g.*, gamers, teachers, witnesses of public events). Instead of a few high-quality well-defined video streams, live streaming providers have now to deal with many low-quality unreliable video streams.

The preparation of a given video channel includes deciding the number of representations to encode, setting the encoder parameters, allocating the transcoding jobs to machines, and transcoding each raw video stream into multiple video representations. Even though solutions have already been proposed for these subproblems, as we discussed at Chapter 2, it is non-trivial to combine them to form a single solution and there is no guarantee that a combination of optimal solutions of each subproblem is an optimal and feasible solution of the global problem. For example, selecting the available representations (resolution and bit rate) without considering the available computing resources is likely to lead to unfeasible solutions.

In this chapter we are interested in maximizing the average user QoE by selecting the optimal encoding parameters under given computing and CDN capacity constraints. More specifically, we present two contributions.

First, we formulate an optimization problem for the management of a data center dealing with a large number of live video streams to prepare for delivery at Section C.3.4. The goal is to maximize the QoE for the end-users subject to the number of available machines in the data center and the CDN delivery budget. With this problem, they highlight the complex interplay between the popularity of channels, the required computing power for video transcoding, the satisfaction of end-users, and the delivery bandwidth. The problem is formulate as an ILP. Then they use a generic solver to compare the performances of standard stream preparation strategies (where all the channels use the same encoding parameters for the transcoding operation) to the optimal. The results highlight the gap between the standard preparation strategies and the optimal solution.

Second, we propose a heuristic algorithm for the preparation of live ABR video streams at Section C.3.5. This algorithm can decide on-the-fly the encoding parameters. Our results show that our proposal significantly improves the QoE of the end-users while using almost constant computing resources even in the presence of a time varying demand.

C.3.2 Current Industrial Strategies

Today's live service provider have to implement a strategy for stream preparation. To the best of our knowledge, no provider has yet implemented an optimal strategy. Typically one of the following two options is implemented. In the first one, used by Twitch, ABR streaming is only offered to some premium broadcasters. That is, only a small subset of channels is transcoded into multiple representations. For the other broadcasters, the raw video is forwarded to the viewers without transcoding. The problem of this solution is that many viewers of standard broadcasters cannot watch the stream because their downloading rate is too low. This problem was discussed in Chapter 6.

The second option consists in delivering all channels with ABR streaming. This is the option we study in this chapter. To the best of our knowledge, the live streaming providers apply the same transcoder settings for all channels although it has been shown in [TAS+14] that such a strategy is sub-optimal. Therefore, we consider two possible strategies.

Full-Cover Strategy. This corresponds to a strategy with one representation per resolution smaller than or equal to the resolution of the source. The bit rate is chosen such as to be the lowest possible for this resolution (100 kbps for low resolutions and 1000 kbps for the high resolutions). With this strategy, viewers with a display size smaller than or equal to the source resolution are guaranteed to find one representation in their display resolution. Moreover, since the Central Processing Unit (CPU) requirements are low for low bit rates, this strategy is the least CPU-hungry possible strategy (among the strategies with at least one representation per resolution).

Zencoder Strategy. We follow here the recommendations of one of the main cloud transcoding providers, namely Zencoder. The recommendations are given on their public website.[2] We give in Table C.1 the characteristics of the set of representations. Again, only representations with a bit rate and a resolution smaller than or equal to the video source are produced.

Video Resolution	Bitrates (in kbps)
224p	200, 400, 600
360p	1000, 1500
720p	2000
1080p	2750

Table C.1: Zencoder encoding recommendations for live streaming (adapted to our bit rate ranges).

[2] http://zencoder.com/en/hls-guide

C.3.3 Transcoding CPU and PSNR Data Set

As part of the responsibilities considered for live streaming services is the transformation of incoming raw video into a multimedia object that can be delivered to a large number of users. This phase is called *preparation*. The preparation comprises multiple tasks, such as content sanity check, implementation of the Digital Rights Management (DRM) policies and transcoding. This thesis explores problematics associated to the transcoding operations performed by service providers. This task is CPU consuming and the live scenario adds even more challenge. The transcoding task prepares the raw video stream into a set of ABR video streams for delivery. By focusing on the transcoding task we consider for each session the following twofold service provider goal:

- Decide the set of video representations to be transcoded. The decision includes the number of representations, and, for each representation, the bit rate and the resolution;
- Assign the transcoding *jobs* to the machines of the data center.

A key information for the study is the amount of CPU cycles that are required to transcode one raw video into a video stream in a different format. This quantity depends on various parameters, but mostly on (*i*) the bit rate and the resolution of the source, (*ii*) the type of the source, and (*iii*) the bit rate and the resolution of the target video stream. To obtain a realistic estimate, we have performed a set of transcoding operations from multiple types of sources encoded at different resolutions and rates to a wide range of target resolutions and rates. For each one of the transcoding operations, there is a QoE estimation of the transcoded video, and measured the CPU cycles required to perform it. This data set is publicly available.[3] In the following paragraphs we describe each one of the many parameters considered for this data set.

Source Types. Four types of video content are considered, corresponding to four test sequences available at [Xip14]. Each of these four test sequences corresponds to a representative video type as given in Table C.2.

Source Encoding. In current live streaming systems, the encoding of the source is done at the broadcaster side. As shown in Chapter 3, the raw video that is emitted by the broadcaster can be encoded with different parameters. Based on the analysis of the Twitch data set, only four resolutions, from 224p to 1080p were considered. Also the video bit rates were restricted to be in

[3] http://dash.ipv6.enstb.fr/dataset/transcoding/

Video Type	Video Name
Documentary	Aspen, Snow Mountain
Sport	Touchdown Pass, Rush Field Cuts
Cartoon	Big Buck Bunny, Sintel Trailer
Video	Old Town Cross

Table C.2: Test videos and corresponding type.

ranges covering 90% of the sources observed in the Twitch data set. Table C.3 have more details on the resolutions and bit rates evaluated.

Target Videos. The format of the target videos depends on the source video. For each input video all the resolutions that are smaller than, or equal to, the input video are considered and for each resolution all the rates that are smaller than, or equal to, the rate of the input video. Again, Table C.3 lists all the specifics combinations for target videos.

Input and Output Rates. Only the rates covering 90% of the sources observed in the Twitch live sessions data set were considered, as detailed in Table C.3. For low resolutions (224p and 360p), the set of bit rates ranges from 100 kbps up to 3000 kbps with steps of 100 kbps, while for high resolutions (720p and 1080p), the set of bit rates ranges from 1000 kbps up to 3000 kbps with steps of 250 kbps. Thus, each video sequence from Table C.2 can be encoded into 78 different combinations of rates and resolutions. To obtain these 78 sources, the original full-quality decoded videos were considered from Table C.2 and then encoded into each of the 78 videos that was considered as possible raw videos.

Resol.	Width x Height	Min–Max Rates	Rate Steps
224p	400 x 224	100–3000 kbps	100 kbps
360p	640 x 360	100–3000 kbps	100 kbps
720p	1280 x 720	1000–3000 kbps	250 kbps
1080p	1920 x 1080	1000–3000 kbps	250 kbps

Table C.3: Resolutions and ranges of rates for the raw videos.

Transcoding. The transcoding operations were performed on a standard server, similar to what can be found in most public data centers. The debate about whether GPU can be used in a public cloud is still relevant today. Those who do not believe in a wide availability of GPU in the cloud emphasize the poor performance of standard virtualization tools on GPU [SL13] and the preferences of the main cloud providers for low-end servers (the so-called *wimpy* servers) in data centers [BCH13]. On the other hand, new middleware have been developed to improve GPU sharing and VM-GPU matching in data

centers [Gon13], so it may be possible to envision a wider deployment of GPU in a near future. Nevertheless, in this data set, a conservative position was adopted, which is the choice made by today's live streaming service providers, and only the availability of CPU in the servers are considered.

As for the physical aspect of the CPU cycles measurements, no performance impact by virtualization was considered, *i.e.* a transcoder running in a VM on a shared physical machine is as fast as if the same transcoder ran directly on the physical machine. The server used was an Intel Xeon CPU E5640 at 2.67GHz with 24 GB of RAM using Linux 3.2 with Ubuntu 12.04.

The transcoding operation performed is summarized in Figure C.6. This operation has been done 12,168 times in total. This corresponds to 4 (the number of video types) multiplied by 78 (the number of possible sources) multiplied by 39 (the average number of possible target videos). Recall that, for each input video, were produced only videos with resolutions and bit rates lower than or equal to those of the input. That is, only a subset of the 78 possible representations are created from a given raw video.

For the transcoding `ffmpeg` was used with the same parameters as in [BCF14], which is a study conducted by the leading developers of the popular GPAC video encoder. The command is

```
ffmpeg -i source_name -vcodec libx264 -preset
ultrafast -tune zerolatency   -s target_resolution -r 30 -b
target_rate -an target_name
```

Measuring CPU cycles. As discussed above, many transcoding operations were performed using an Intel Xeon CPU E5640 at 2.67GHz with 24 GB of RAM using Linux 3.2 with Ubuntu 12.04. To measure the number of used CPU cycles, the `perf` tool[4], a profiler for Linux 2.6+ based systems was used, with the next command:

```
perf stat -x -e CYCLES
```

This command provides access to the counter collecting the number of CPU cycles at the Performance Monitoring Unit (PMU) of the processor. Then, this number is divided by the duration in s of the video sequences to obtain the frequency of CPU (in GHz) required to perform the transcoding during a running time equal to the play time of the video, that is, the frequency of CPU required to do a live transcoding.

Figure C.7a shows the experimental results for all the target videos generated from a source of type "movie," 1080p resolution and encoded at 2,750 kbps. The empirical CPU cycles measurements are depicted as circles. Overall, 588 curves similar to these ones were prepared to cover the 12,168 transcoding operations. For the sake of brevity, here only these four

[4] https://perf.wiki.kernel.org

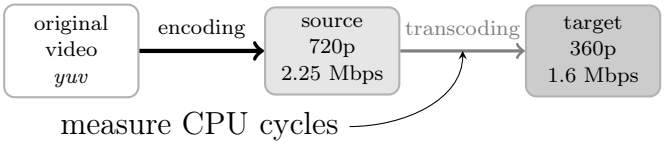
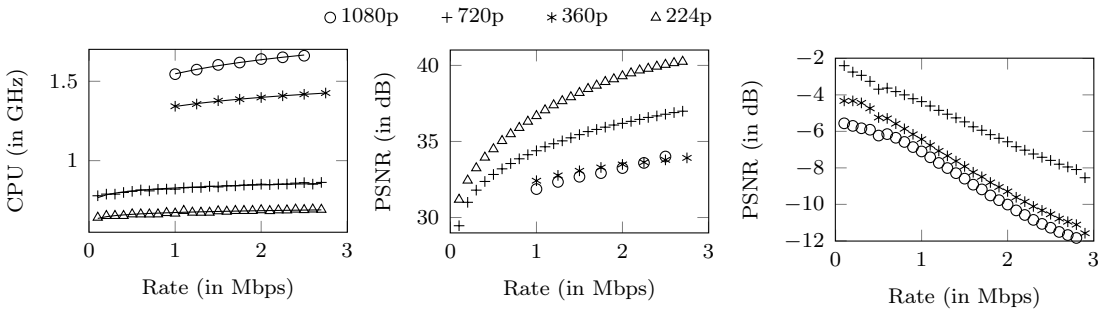


Figure C.6: Measuring the CPU cycles for the transcoding of any source to any target video. Here an example with a source at 720p and 2.25 Mbps and a target video at 360p and 1.6 Mbps.



(a) CPU cycles (b) Viewers satisfaction (c) Up-scaling penalties

Figure C.7: Results obtained from the transcoding operations performed. Satisfaction of viewers and CPU cycles needed to transcode from source 1080p, 2750 kbps, and type *movie* to various target videos. Up-scaling penalties curves from the source video 224*p*, and type *movie*.

are shown. The full set of curves are publicly available for consult with the data set.

Estimating QoE. The QoE was evaluated by means of the Peak Signal to Noise Ratio (PSNR) score [SYDN+10], which is a full-reference metric commonly used due to its simplicity. We apply the PSNR filter[5] provided by `ffmpeg` in two different cases illustrated in Figure C.8.

The first case, depicted on top of Figure C.8, corresponds to the scenario where a target (transcoded) video at a given spatial resolution is watched on a display of the same size. The PSNR filter compares the target video against a reference video. The reference is the source encoded at the same resolution as the target but with the largest encoding bit rate considered in the study (3,000 kbps). The measurement was repeated as many times as target videos, *i.e.*, 12,168 times. As in the case of the live transcoding CPU curves, only one example corresponding to the PSNR curves is depicted in Figure C.7b. All other set of curves are available at the public site hosting the data set.

[5] https://www.ffmpeg.org/ffmpeg-filters.html#psnr

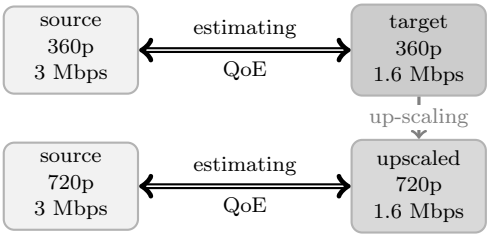


Figure C.8: Estimating the QoE for a target video. On top, a target video at 360p and 1.6 Mbps watched on a 360p display. On the bottom, the same target video upscaled to be watched on a 720p display.

The second scenario, shown on bottom of the Figure C.8, refers to the situation when a target (transcoded) video at a given resolution need to be upscaled to be watched on a display with a higher size. This up-scaling introduces a penalty on the final QoE for the viewer. To estimate these penalties, a new battery of transcoding operations was carried out, using the same `ffmpeg` command as before, but the input and output video are the target and the upscaled video, respectively. The upscaled video is compared against a reference with an encoding rate of 3,000 kbps but with the *same resolution as the upscaled target*. The penalty, using the example of up-scaling from 360p to 720p in Figure C.8, can simply be computed by subtracting from the PSNR measure on top of the Figure the PSNR measurement on the bottom. In Figure C.7, the up-scaling penalties for a 224p source of type movie is depicted.

C.3.4 Optimizing Stream Preparation

We first address the problem of live video stream preparation with an optimization approach. As previously said, the preparation includes both the decision about the encoding parameters of the video representations and the assignment of transcoding jobs to the machines. Our goal is to maximize the QoE of viewers subject to the availability of hardware resources. In the following we first provide a formal formulation of the problem, and then we present the ILP model that we use to solve the optimization problem. Finally, we compare the performance of the industry-standard strategies with the optimal.

C.3.4.1 Notations

Let $\mathcal{I}$ be the set of raw video streams encoded at the broadcaster side. Each video stream $i \in \mathcal{I}$ is characterized by a type of video content $v_i \in \mathcal{V}$, an encoding bit rate $r_i \in \mathcal{R}$ and a spatial resolution $s_i \in \mathcal{S}$, where $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{S}$ are the sets of video types, the set of encoding bit rates (in kbps) and the set

of spatial resolutions, respectively. We have shown in Chapter 2 the diversity of raw videos.

Let $\mathcal{O}$ be the set of the possible video representations that are generated from the source by transcoding jobs. Each representation $o \in \mathcal{O}$ corresponds to a triple $(v_o,\ r_o,\ s_o)$, that is, to a video representation of type of content $v_o \in \mathcal{V}$ encoded at the resolution $s_o \in \mathcal{S}$ and at the bit rate $r_o \in \mathcal{R}$.

Let $\mathcal{M}$ be the set of physical machines where the transcoding tasks should be performed. Each machine $m \in \mathcal{M}$ can accommodate transcoding jobs up a maximum CPU load of P_m GHz.

To reduce the size of the problem, and make it more tractable, we introduce the notion of *viewer type*. Let $\mathcal{U}$ be the set of viewers types. All viewers in a given viewer type $u \in \mathcal{U}$ have the same display resolution (i.e., the spatial resolution at which the video is displayed on the device) $s_u \in \mathcal{S}$, request the same video type $v_u \in \mathcal{V}$, and use an Internet connection with the same minimum bandwidth of at least c_u kbps. However, viewers of the same type u watch different channels. We denote by d_{iu} the number of viewers of type u watching a given channel i. Note that a viewer of a given type u can play segments encoded at resolutions lower than its display size s_u by performing spatial up-sampling before rendering.

A viewer from viewer type u watching a video representation o transcoded from a stream i experiences a satisfaction level of f_{iou}, which is an increasing function of the bit rate r_o. Based on the transcoding CPU and PSNR data set presented in Section C.3.3, we know that the satisfaction function depends on the video content type v_o, the resolution s_o and the original raw video stream i. As previously said, the satisfaction f_{iou} also depends on whether the video should be up-scaled or not, since up-scaling introduces a penalty on the final satisfaction value. We incorporate this up-scaling penalty into the satisfaction computation by the following definition of the satisfaction f_{iou}:

$$f_{iou} = \begin{cases} f_{io}, & \text{if } s_o = s_u \\ f_{io} - q_{ou}, & \text{if } s_o < s_u \end{cases} \quad i \in \mathcal{I}, o \in \mathcal{O}, u \in \mathcal{U} \tag{C.6}$$

where f_{io} is the satisfaction level when the display resolution and the target video resolution match, and q_{ou} is the penalty of the up-scaling process from resolution s_o to the viewer display size s_u. Table C.4 summarizes the notation used throughout the chapter.

Name	Description
$f_{iou} \in \mathbb{R}^+$	Satisfaction level for a representation o transcoded from a stream i watched on a display of size s_u
$f_{io} \in \mathbb{R}^+$	Satisfaction level for a representation o transcoded from a stream i l when display size s_u and target resolution s_o match
$q_{ou} \in \mathbb{R}^+$	Up-scaling penalty from resolution s_o to the viewer display size s_u
$d_{iu} \in \mathbb{Z}^+$	Number of viewers of type u watching a stream i
$r_o \in \mathbb{R}^+$	Value in *kbps* of the encoding bit rate of the representation o
$c_u \in \mathbb{R}^+$	Maximum Internet connection capacity in *kbps* of viewer type u
$v_u \in \mathbb{V}$	Video stream requested by viewer type u
$s_u \in \mathbb{S}$	Display size (spatial resolution) of viewer type u
$N \in \mathbb{Z}^+$	Overall number of viewers
$R \in [0, 1]$	Minimum fraction of viewers that must be served
$p_{io} \in \mathbb{R}^+$	CPU requirement to perform the live transcoding from stream i to representation o in GHz
$P_m \in \mathbb{R}^+$	CPU capacity of a virtual machine m in GHz

Table C.4: ILP notation.

C.3.4.2 Integer Linear Program Model

We now describe the ILP. The decision variables in the model are:

$\alpha_{iou} \in \mathbb{Z}_{\geq 0}$: Number of viewers of type u watching a representation o transcoded from a stream i.

$$\beta_{iom} = \begin{cases} 1, & \text{if machine } m \text{ transcodes stream } i \text{ into} \\ & \text{representation } o \\ 0, & \text{otherwise.} \end{cases}$$

With these definitions, the optimization problem can be formulated as shown in (C.7).

Integer Linear Programming formulation

$$\max_{\{\boldsymbol{\alpha},\boldsymbol{\beta}\}} \sum_{i\in\mathcal{I}}\sum_{o\in\mathcal{O}}\sum_{u\in\mathcal{U}} f_{iou}\cdot\alpha_{iou} \tag{C.7a}$$

$$\text{s.t. } \alpha_{iou} \le N\cdot\sum_{m\in\mathcal{M}}\beta_{iom}, \qquad i\in\mathcal{I}, o\in\mathcal{O}, u\in\mathcal{U} \tag{C.7b}$$

$$\sum_{m\in\mathcal{M}}\beta_{iom} \le \sum_{u\in\mathcal{U}}\alpha_{iou}, \qquad i\in\mathcal{I}, o\in\mathcal{O} \tag{C.7c}$$

$$\alpha_{iou} \le \begin{cases} N, & \text{if } s_u \ge s_o \\ 0, & \text{otherwise} \end{cases} \qquad i\in\mathcal{I}, o\in\mathcal{O}, u\in\mathcal{U} \tag{C.7d}$$

$$\sum_{o\in\mathcal{O}}\alpha_{iou} \le d_{iu}, \qquad i\in\mathcal{I}, u\in\mathcal{U} \tag{C.7e}$$

$$\sum_{i\in\mathcal{I}}\sum_{o\in\mathcal{O}}\left(r_o - c_u\right)\cdot\alpha_{iou} \le 0, \qquad u\in\mathcal{U} \tag{C.7f}$$

$$\sum_{i\in\mathcal{I}}\sum_{o\in\mathcal{O}}\sum_{u\in\mathcal{U}}\alpha_{iou} \ge R\cdot N, \tag{C.7g}$$

$$\beta_{iom} \le \begin{cases} 1, \text{ if } (v_i = v_o \;\&\; s_i = s_o \;\&\; b_i > b_o) \;\|\; (v_i = v_o \;\&\; s_i > s_o \;\&\; b_i \ge b_o) \\ 0, \text{ otherwise} \end{cases} \qquad i\in\mathcal{I}, o\in\mathcal{O}, m\in\mathcal{M} \tag{C.7h}$$

$$\sum_{m\in\mathcal{M}}\beta_{iom} \le 1, \qquad i\in\mathcal{I}, o\in\mathcal{O} \tag{C.7i}$$

$$\sum_{i\in\mathcal{I}}\sum_{o\in\mathcal{O}} p_{io}\cdot\beta_{iom} \le P_m, \qquad m\in\mathcal{M} \tag{C.7j}$$

$$\alpha_{iou} \in [0, N], \qquad i\in\mathcal{I}, o\in\mathcal{O}, u\in\mathcal{U} \tag{C.7k}$$

$$\beta_{iom} \in \{0, 1\}, \qquad i\in\mathcal{I}, o\in\mathcal{O}, m\in\mathcal{M} \tag{C.7l}$$

The objective function (C.7a) maximizes the average viewer satisfaction. The constraints (C.7b) and (C.7c) set up a consistent relation between the decision variables α and β. The constraint (C.7d) establishes that a viewer of type u can play only the transcoded representations o with spatial resolutions equal or smaller than the viewer display size s_u, that is, those susceptible to experience an up-sampling operations at the rendering. The constraints (C.7e) ensures that the sum of all the viewers of type u watching any representation o transcoded from a given stream i does not exceed the number of viewers

of type u originally watching the stream i. The constraint (C.7f) limits the viewer link capacity. The constraint (C.7g) force us to serve at least a certain fraction R of viewers. The constraint (C.7h) forces that only transcoding operations defined over the same video content type are allowed and it forbids senseless transcoding operations, like transcoding to higher bit rates or higher resolutions or transcoding to the same rate-resolution pair. The constraint (C.7i) guarantees that a given transcoding task (i, o) is performed in one unique machine m. Finally, (C.7j) sets the CPU capacity of each machine m.

C.3.4.3 Settings for Performance Evaluation

To find the exact solution of the optimal problem, we use the generic solver IBM ILOG CPLEX [IBM14] on a set of instances. Unfortunately, this approach does not allow solving instances as large as the ones that live service providers face today. Thus, we have built problem instances based on the data sets introduced in Chapter 3 and Section C.3.3 but of a smaller size.

Incoming Videos from Broadcasters. We restrict the size of the set of sources by picking only the 50 most popular channels from the Twitch data set. More precisely, we take 66 snapshots from the data set, corresponding to those ones extracted every 4 hours along 11 days since April 10, 2014 at 00:00. For each snapshot, we use the channel information (bit rate and resolution), which we modify slightly to match the spatial resolutions and bit rates from Table C.3. Each channel is randomly assigned to one of the four video types given in Table C.2.

QoE for Target Videos. We use the transcoding CPU and PSNR data set presented in Section C.3.3 to obtain the QoE (estimated as a PSNR score) f_{io} of a target video o obtained from transcoding a source i. The up-scaling penalties q_{ou} are fixed using PSNR measures from the situation shown on bottom of the Figure C.8 (target resolution lower than display one).

CPU for the Transcoding Tasks. Still to reduce the size of the instances, and thus the complexity of the problem, we fit an exponential function to the set of CPU measurements:

$$p = a \cdot r^b \tag{C.8}$$

where p is the number of GHz required to transcode a source into a target, a and b are the parameters used in the curve fitting and the parameter r is the bit rate in Mbps of the target video. The values of the parameters a and b depend on (i) the source video (content type, bit rate and resolution), and (ii) the resolution of the target video. The fitting curves are identified by

continuous lines in Figure C.7a. Table C.5 gives the parameters a and b used in the curves shown in Figure C.7a.

Target Resol	a	b
224p	0.673091	0.024642
360p	0.827912	0.033306
720p	1.341512	0.060222
1080p	1.547002	0.080571

Table C.5: Parameters of the fitting model of the transcoding CPU curves. Source stream: 1080p, 2,750 kbps, movie

Viewers. The viewers set $\mathcal{U}$ is based on the DASH sessions data set [BSMM14] presented in Chapter 6. However, the number of viewers is too large and we implement the concept of user type. To build the types, we divide the range of bandwidth into *bins*, whose limits are selected so that each bin contains an equal number of viewers. A viewer type corresponds to a bin, with a display spatial resolution set according to the lower bandwidth in the bin, and the downloading rate of the viewer type is equal to the higher bandwidth in the bin. The number of viewers d_{iu} watching a raw video i is proportionally set up according to the popularity of the channel in the Twitch data set.

C.3.4.4 Numerical Results

We now show the results of our analysis. Our motivation is to determine how far from the optimal are current industry-standard strategies. In Figure C.9, we represent the average QoE, expressed as the PSNR in dB, as a function of the number of machines. The line represents the results obtained from solving the optimization problem with CPLEX. We show with gray pins the results for both industry-standard strategies. The results are the average over all the snapshots we took from the Twitch data sets.

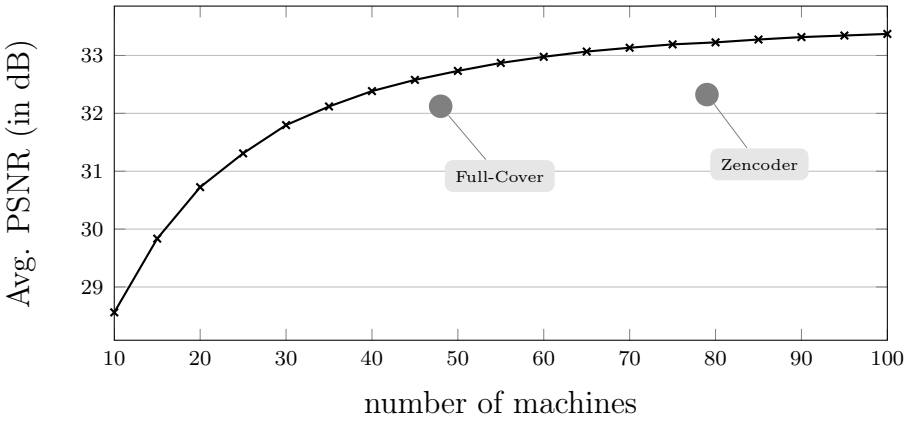
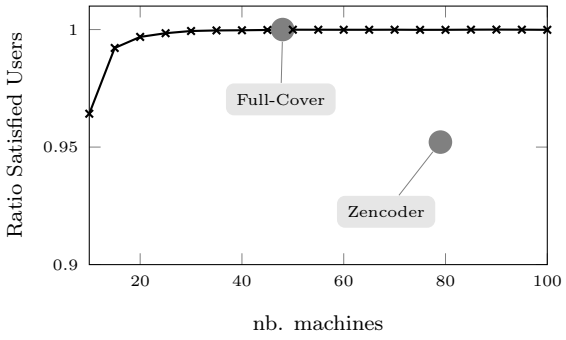


Figure C.9: Optimal average QoE for the viewers vs. the number of machines that are used in the data centers. The 50 most popular channels from several snapshots of the Twitch data set are transcoded.

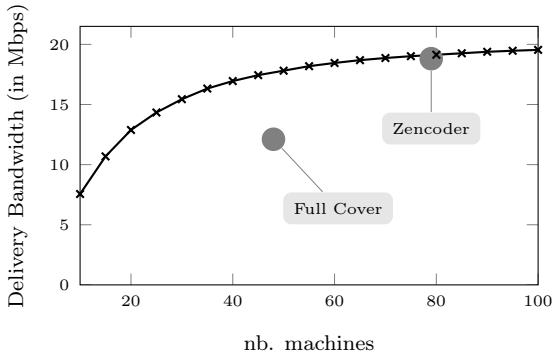
We first emphasize that the amount of hardware resources in the data center has a significant impact on the QoE for the viewers. The difference of PSNR reaches 4dB between 10 and 100 machines. This highlights the need of being able to reserve the right amount of resources in the data center. However, the ability to forecast the load and to reserve the resources is not trivial for elastic live streaming services such as Twitch.

Our second main observation is that, on our data set, the Full-Cover strategy is more efficient than the Zencoder one in terms of trade-off QoE-CPU. The Full-Cover strategy is close to the optimal, and thus represents an efficient implementation with respect to its simplicity. Note however that Full-Cover needs 48 machines, while there exists a solution with the same QoE but with only 35 machines. Therefore, a significant reduction of resources to reserve can be obtained. The Zencoder strategy is outperformed by the Full-cover one, as it consumes nearly twice the CPU cycles for a tiny increase of the QoE. For a similar amount of CPU, the QoE gap between the Zencoder strategy and the optimal is more than 0.9 dB, which is significant.

To complete this study, we provide another view of the choices to be taken in Figure C.10. Here, we show the ratio of served users and the amount of delivery bandwidth that is required to serve the users. In our ILP, we optimize the average QoE so the solutions that are found by CPLEX are not optimal on other aspects. In Figure C.10b, we see that the delivery bandwidth of the optimal solution is significantly higher than the Full-Cover, which may annihilate the gains obtained by using fewer machines. Please note that both parameters of Figure C.10 can also be the objective of the ILP. In the same vein, the ILP can also be re-written so that the parameter to be optimized is the amount of CPU needed, subject to a given QoE value.



(a) Ratio of satisfied users



(b) Delivery Bandwidth

Figure C.10: Other views on the optimal solution of Figure C.9

C.3.5 A Heuristic Algorithm

We now present and evaluate an algorithm for massive live streaming preparation. Our goal is to design a fast, light, adaptive algorithm, which can be implemented in production environments. This algorithm should in particular be able to *absorb* the variations of the streaming service demand while using a fixed data center infrastructure.

C.3.5.1 Algorithm Description

The purpose of the algorithm is to update the set of transcoded representations with respect to the characteristics and the popularity of the incoming raw videos. The algorithm is executed on a regular basis (for example every five minutes to stick to the Twitch API) by the live streaming service provider in charge of the data center. You can find in Appendix A the pseudo-code of the algorithms.

In our algorithm we process each channel iteratively in a decreasing order of their popularity. For a given channel, the algorithm has two phases: First,

we decide a CPU budget for this channel. Second, we determine a set of representations with respect to the CPU budget computed during the first phase.

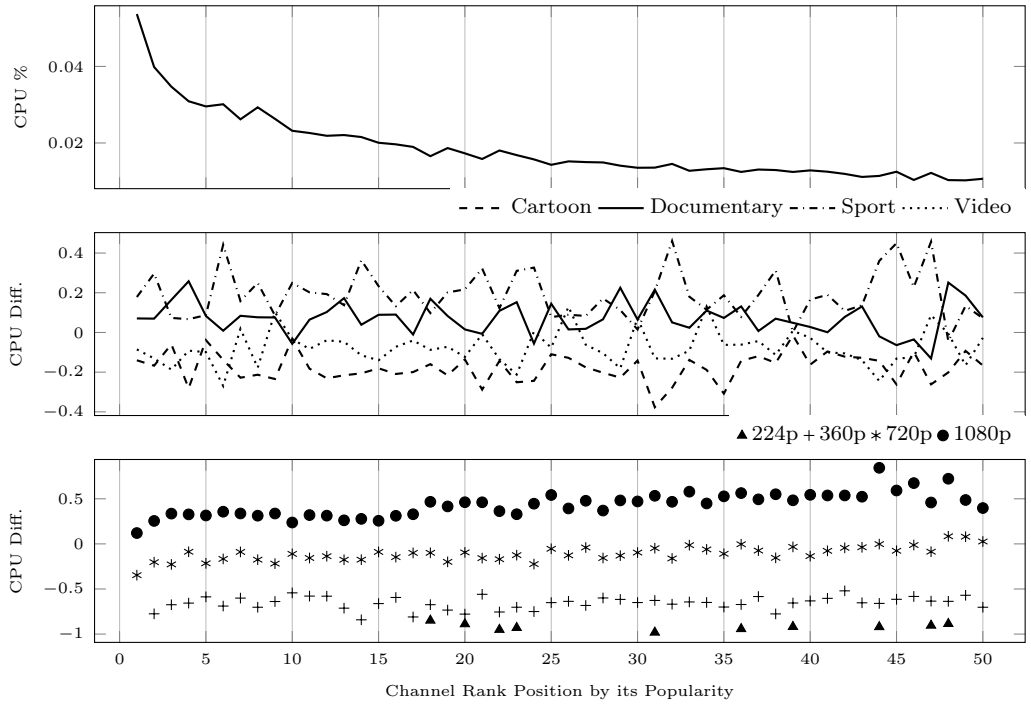


Figure C.11: CPU information for a given channel popularity rank. Data comes from the average optimal solution with 100 machines. From the average total CPU % according to the rank (top figure), we derive the difference between distinct video types (middle figure) and between distinct channel input resolutions (bottom figure).

Set a CPU Budget Per Channel. We base our algorithm on the observations of the optimal solutions found by CPLEX. Four main observations are illustrated in Figure C.11: (*i*) on average, the ratio of the overall CPU budget of a given channel is roughly proportional to the ratio of viewers watching this channel; (*ii*) the CPU budget per channel is less than 10 GHz; (*iii*) on average, some video types (*e.g.*, sport) require more CPU budget than others (*e.g.*, cartoon); and (*iv*) the higher is the resolution of the source, the bigger the CPU budget.

We derive from these four observations the algorithm shown in Algorithm 1. We start with the most popular channel. We first set a *nominal* CPU budget according to the ratio of viewers and the maximal allowable budget. Then we adjust this nominal budget based on a *video type weight* and a *resolution weight* to obtain a CPU budget for this channel.

The values we used for both *video type weight* and *resolution weight* are given in Tables C.6. These values correspond the average of the curves plotted in Figure C.11, which we obtain from the optimal solutions computed by CPLEX on the 50 most popular channels and 100 machines. The x-axis is the rank of the channels according to their popularity. On the top we show the average distribution of CPU budget per channel. On the bottom, we show the average difference between the average CPU budget when a channel from a given type (respectively resolution) is at a given rank and the average CPU budget for channels at this rank. This difference allows us to compute the adjustment of CPU budget according to the video type and the resolution.

Video Type	Weight
Cartoon	-0.176
Documentary	0.072
Sport	0.190
Video	-0.076

Resolution	Weight
224p	-0.917
360p	-0.657
720p	-0.108
1080p	0.432

Table C.6: Video type and resolution weights

Decide the Representations for a Given CPU Budget. The pseudo-code is detailed in Algorithm 2. This algorithm builds the set of representations by iteratively adding the best representation. At each step, the needed CPU budget to transcode the chosen representation should not exhaust the remaining channel budget. To decide among the possible representations, we need to estimate the QoE gain that every possible representation can provide if it is chosen. To do so, we estimate the assignment between the representations and the viewers in Algorithm 3. In short, this algorithm requires a basic knowledge on the distribution of downloading rates in the population of viewers. (In our simulations, we have considered that the service provider has no information, so it considers a uniform distribution of downloading rates in the range between 100 kbps and 3,000 kbps). The idea is then to assign subsets of the population to representations and to evaluate the overall QoE.

To estimate the QoE gain when choosing one representation, we need to consider all the assignments representations-viewers. In Algorithm 3, we evaluate the representations in the set in descending order of their bit rates. At each iteration, we identify the fraction of viewers whose bandwidth is between the rate of the considered representation and the closest representation with superior bit rate. Then, we also have to take into account the display sizes of the viewers (it also depends on the knowledge of the service provider; we again considered it a minimal knowledge of the population). Therefore, this

fraction of viewers is again split into one or more sub-fractions corresponding to their display resolutions. A different value of PSNR is then computed for each sub-fraction. This value is multiplied by the ratio of viewers belonging to the sub-fraction. When all the representations have been assessed, the sum of the PSNR contributions of all the sub-fractions of viewers is returned as the estimated QoE of the set.

C.3.5.2 Simulation Settings

Our simulator is based on the data sets presented in Chapter 3 and Chapter 6, and the extra settings given in Section C.3.4.3. However, in contrast to the ILP, our heuristic is expected to scale. Therefore we evaluate the heuristic and the aforementioned industry-standard strategies on the complete data set containing all online broadcasters at each snapshot. Regarding the viewers, we consider now each viewer, to which we randomly assign a bandwidth value of the DASH session data set and a display resolution accordingly. We use the actual number of viewers watching channels according to the Twitch data set.

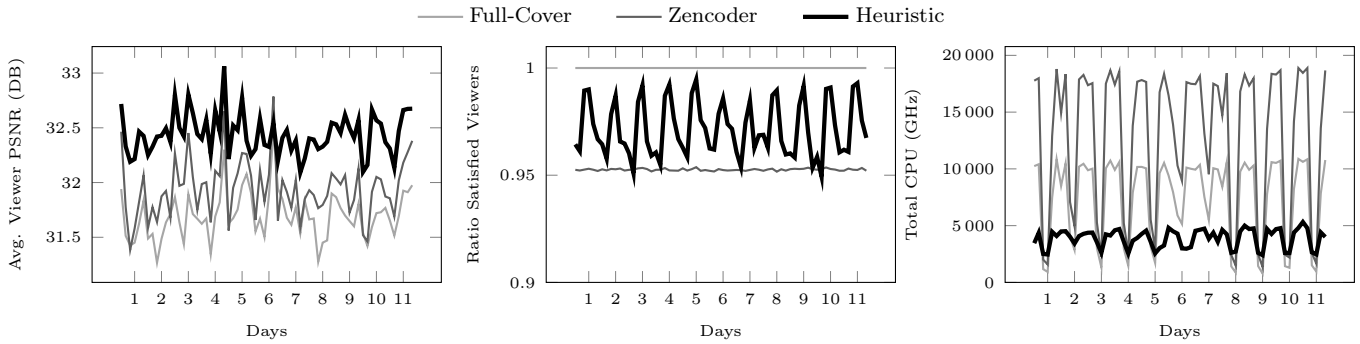
Please note also that we focus here on the decision about the representations (number of representations and transcoding parameters), and we neglect the assignment of transcoding jobs to machines. This does not impact the evaluation since all tested strategies (our heuristic and both industry-standard strategies) can be evaluated without regard to this assignment. We let for future works the integration of the VM assignment policy into middleware such as OpenStack.[6]

C.3.5.3 Performance Evaluations

In the following, we present the same set of results from two different perspectives; first, we show how the performances evolve throughout the eleven days we consider. Then, we present the results in order to highlight the main features of the algorithms.

In Figure C.12 we show the three metrics during our 11-days data set. The combination of the three figures reveals important characteristics of the strategies. The main point we would like to highlight is that our heuristic keeps a relatively constant, low CPU consumption without regard to the traffic load in input. Our heuristic also succeeds in maintaining a high QoE. To achieve this excellent trade-off, our heuristic adjusts with the ratio of served viewers. Yet, this ratio maintains a high value since it is always greater than 95%. Our heuristic thus demonstrates the benefits from having different representation

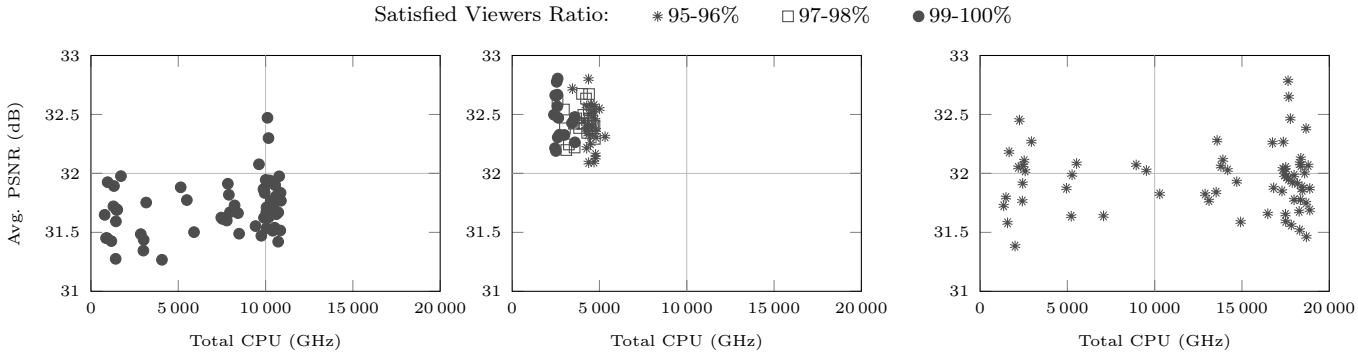
[6] http://www.openstack.org/



(a) Average Viewer PSNR (b) Ratio of satisfied viewers (c) Total CPU needs

Figure C.12: Different metric results over time for the distinct solutions: Full-Cover strategy, Zencoder encoding recommendations and our Heuristic.

sets for the different channels according to their popularity. The industry-standard strategies are less capable of absorbing the changing demand. In particular, the CPU needs of the Zencoder strategy ranges from 1,000 GHz to 18,000 GHz while the average QoE is always lower than for our heuristic.



(a) Full-Cover (b) Heuristic (c) Zencoder

Figure C.13: Total required CPU (in GHz) by the perceived QoE for strategies Full-Cover, Zencoder encoding recommendations and Heuristic. The markes shape indicates the percentage of satisfied viewers. Each point corresponds to one of the 66 snapshots.

To highlight the relationship between CPU needs and QoE for the population, we represent in Figure C.13 a cloud of points for each snapshot. The Full-Cover has most points in the southwest area of the Figure, which corresponds to a low CPU utilization but also a low QoE. We also note that the distance between two points can be high, which emphasizes an inability to absorb load variations. This inability is even stronger for the Zencoder strategy,

for which the points are far from each other, covering all areas of the Figure. On the contrary, our heuristic absorbs well elastic services, with points that are concentrated in the northwest part of the Figure, which means low CPU and high QoE.

C.3.6 Conclusion

This chapter studies the management of new live adaptive streaming services in the cloud from the point of view of *streaming providers* using *cloud computing platforms*. All the simulations we conducted make use of real data from three data sets covering all the actors in the system. The study is focused on the interactions between the optimal video encoding parameters, the available CPU resources and the QoE perceived by the end-viewers. We use an ILP to model this system and we compare its optimal solution to current industry-standard solutions, highlighting the gap between the two. Due to the ILP computational limitations, we propose a practical algorithm to solve problems of real size, thanks to key insights gathered from the optimal solution. This algorithm finds representations beating the industry-standard approaches in terms of the trade-off between viewers QoE and CPU resources needed. Furthermore, it uses an almost-constant amount of computing resources even in the presence of a time varying demand.

C.4 Appendix Conclusion

In this appendix we have discussed works related to the application of our live session data set. With our data set we are able to create scenarios that are based on real traces for two studies.

On the first work, the data set was fundamental in order to have relevant inputs for the competition among service providers over the CDN delivery resources. Moreover, we analyzed different aspects in the competition, by identifying one service, Twitch, as an incumbent, well-established service with thousands of broadcasters, and another one, YouTube Live, as a challenger. We show that even if the bigger competitor tries to monopolize the CDN resources, the CDN remains relatively neutral in the delivering process.

On the second study, we analyze the benefits of the a heuristic algorithm with our data set and the possibility of usage of this algorithm on large scale systems present on UGC live streaming service. The ILP relies on multiple generalizations of the real scenario to cope with the scale. The proposed heuristic algorithm is able to handle all the inputs from our data set. Also it is able to provide better QoE for viewers by wisely choosing which channels to be delivered with ABR.

We hope that other works can derive or benefit from our data set and its analysis, which are publicly available for community.

Bibliography

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010. (Cited on page 21.)

[AGH+12] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *INFOCOM*. IEEE, 2012. (Cited on pages 19 and 20.)

[AH02] L. A. Adamic and B. A. Huberman. Zipf's law and the Internet. *Glottometrics*, 3:143–150, 2002. (Cited on pages 30, 38 and 115.)

[AJCZ12] Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen, and Zhi-Li Zhang. Vivisecting youtube: An active measurement study. In *INFOCOM*. IEEE, 2012. (Cited on pages 2 and 19.)

[AJZ10] Vijay Kumar Adhikari, Sourabh Jain, and Zhi-Li Zhang. Youtube traffic dynamics and its interplay with a tier-1 ISP: an ISP perspective. In *IMC*. ACM, 2010. (Cited on page 13.)

[Ale12] Kris Alexander. Fat client game streaming or cloud gaming. Akamai Blog, Aug. 2012. https://blogs.akamai.com/2012/08/part-2-fat-client-game-streaming-or-cloud-gaming.html. (Cited on page 18.)

[ALX11] E. Altman, A. Legout, and Y. Xu. Network non-neutrality debate: An economic analysis. In *Proc. of IFIP Networking*, 2011. (Cited on page 110.)

[Ama14] Inc. Amazon.com. Amazon.com to acquire twitch. BusinessWire, 2014. http://www.businesswire.com/news/home/20140825005820/en/Amazon.com-Acquire-Twitch. (Cited on page 31.)

[APSB15] Ramon Aparicio-Pardo, Karine Pires, Gwendal Simon, and Alberto Blanc. Transcoding live adaptive video streams at a massive scale in the cloud. In *ACM MMSys*, 2015. (Cited on pages 6 and 93.)

[ASV11] Micah Adler, Ramesh K. Sitaraman, and Harish Venkataramani. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55(18):4007–4020, 2011. (Cited on page 3.)

[AZL+12] Paarijaat Aditya, Mingchen Zhao, Yin Lin, Andreas Haeberlen, Peter Druschel, Bruce Maggs, and Bill Wishon. Reliable client accounting for p2p-infrastructure hybrids. In *NSDI*. USENIX, 2012. (Cited on page 19.)

[BCF14] Nassima Bouzakaria, Cyril Concolato, and Jean Le Feuvre. Overhead and performance of low latency live streaming using MPEG-DASH. In *IEEE IISA*, 2014. (Cited on pages 3, 25 and 124.)

[BCH13] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2013. Second Edition. (Cited on page 123.)

[BH09] Luiz André Barroso and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 4(1):1–108, 2009. (Cited on page 17.)

[BKR10] David Breitgand, Gilad Kutiel, and Danny Raz. Cost-aware live migration of services in the cloud. In *SYSTOR*. ACM, 2010. (Cited on page 95.)

[Bon10] V. Bonneau. Evolution of the CDN market. In *CDN World Summit*, 2010. (Cited on page 110.)

[BSA+13] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. In *ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013*, pages 339–350. ACM, 2013. (Cited on page 94.)

[BSMM14] Simone Basso, Antonio Servetti, Enrico Masala, and Juan Carlos De Martin. Measuring DASH streaming performance from the end users perspective using neubot. In *ACM MMSys*, 2014. (Cited on pages 4, 76, 77, 83 and 131.)

[BVB10] Ruben Van Den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *IEEE CLOUD*. IEEE, 2010. (Cited on page 22.)

[BYGJ+09] Ricardo A. Baeza-Yates, Aristides Gionis, Flavio Junqueira, Vassilis Plachouras, and Luca Telloli. On the feasibility of multi-site web search engines. In *Proc. of ACM CIKM*, 2009. (Cited on page 21.)

[CJW09] Hyunseok Chang, Sugih Jamin, and Wenjie Wang. Live streaming performance of the zattoo network. In *IMC*. ACM, 2009. (Cited on page 14.)

[CKR+07] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue B. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *IMC*. ACM, 2007. (Cited on page 15.)

[CKR+09] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue B. Moon. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Trans. Netw.*, 17(5):1357–1370, 2009. (Cited on page 38.)

[CMT12] P. Coucheney, P. Maillé, and B. Tuffin. Network Neutrality Debate and ISP Inter-Relations: Traffic Exchange, Revenue Sharing, and Disconnection Threat. Technical report, Inria, 2012. (Cited on page 110.)

[CMT13] P. Coucheney, P. Maillé, and B. Tuffin. Impact of Reputation-Sensitive Users and Competition Between ISPs on the Net Neutrality Debate. *IEEE Trans. on Network and Service Management*, 2013. (Cited on page 110.)

[Cro11] P. Crocioni. Net neutrality in Europe: Desperately seeking a market failure. *Telecommunications Policy*, 35(1):1 – 11, 2011. (Cited on page 110.)

[CSF10] Gloria Chatzopoulou, Cheng Sheng, and Michalis Faloutsos. A first step towards understanding popularity in youtube. In *IEEE INFOCOM Workshops*, 2010. (Cited on page 38.)

[CWLC14] Rui Cheng, Wenjun Wu, Yihua Lou, and Yongquan Chen. A cloud-based transcoding framework for real-time mobile video conferencing system. In *IEEE MobileCloud*, 2014. (Cited on page 3.)

[CWSR14] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Syst.*, 20(5):503–519, 2014. (Cited on page 17.)

[DDH+11] Yuan Ding, Yuan Du, Yingkai Hu, Zhengye Liu, Luqin Wang, Keith W. Ross, and Anindya Ghose. Broadcast yourself: understanding youtube uploaders. In *IMC*. ACM, 2011. (Cited on page 15.)

[DiP14] Matthew DiPietro. Twitch.tv: 2013 retrospective. Twitch Blog, Jan. 2014. http://twitch.tv/year/2013. (Cited on page 120.)

[Fie12] FierceOnlineVideo. Streaming the olympics: A game-changer for online video. FierceOnlineVideo, Aug. 2012. http://is.gd/TNjEKB. (Cited on page 3.)

[FMM+11] Alessandro Finamore, Marco Mellia, Maurizio M. Munafò, Ruben Torres, and Sanjay G. Rao. Youtube everywhere: impact of device and infrastructure synergies on user experience. In *IMC*. ACM, 2011. (Cited on pages ix, 13 and 14.)

[Fri07] R. Frieden. Internet Packet Sniffing and its Impact on the Network Neutrality Debate and the Balance of Power between Intellectual Property Creators and Consumers. Technical report, SSRN, 2007. http://is.gd/FG978B. (Cited on page 110.)

[FW14] Drew Fitzgerald and Daisuke Wakabayashi. Apple Quietly Builds New Networks. Wall Street Journal, Feb. 2014. http://is.gd/MXc2b7. (Cited on pages 1 and 100.)

[GB06] Jiani Guo and L.N. Bhuyan. Load balancing in a cluster-based web server for multimedia applications. *IEEE Transactions on Parallel and Distributed Systems*, 17(11):1321–1334, November 2006. (Cited on pages 3 and 22.)

[GKMS13] Fabrice Guillemin, Bruno Kauffmann, Stephanie Moteau, and Alain Simonian. Experimental analysis of caching efficiency for youtube traffic in an isp network. In *IEEE ITC*, 2013. (Cited on page 40.)

[Gon13] Carlos Reaño Gonzalez. CU2rCU: A CUDA-to-rCUDA Converter. Master's thesis, Universitat Politecnica de Valencia, 2013. (Cited on page 124.)

[HCKS08] K. Hosanagar, J. Chuang, R. Krishnan, and M.D. Smith. Service adoption and pricing of content delivery network (CDN) services. *Management Science*, 54(9):1579–1593, 2008. (Cited on page 110.)

[HKSC04] K. Hosanagar, R. Krishnan, M. Smith, and J. Chuang. Optimal pricing of content delivery network (CDN) services. In *Proc. of the Annual Hawaii Int. Conf. on System Sciences*, 2004. (Cited on page 110.)

[HLC+05] Zhihai He, Yongfang Liang, Lulin Chen, I Ahmad, and Dapeng Wu. Power-rate-distortion analysis for wireless video communication under energy constraints. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(5):645–658, 2005. (Cited on page 23.)

[HLL+07] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, 2007. (Cited on page 16.)

[HMLW11] Zixia Huang, Chao Mei, Li Li, and T. Woo. CloudStream: Delivering high-quality streaming videos through a cloud-based SVC proxy. In *IEEE INFOCOM*, 2011. (Cited on pages 3, 22 and 23.)

[Hof12] Todd Hoff. Gone fishin': Justin.tv's live video broadcasting architecture. High Scalability blog, Nov. 2012. http://is.gd/5ocNz2. (Cited on pages 2, 19, 21, 29 and 71.)

[Hos08] K. Hosanagar. CDN pricing. In R. Buyya, M. Pathan, and A. Vakali, editors, *Content Delivery Networks*, volume 9 of *Lecture Notes Electrical Engineering*. Springer, 2008. (Cited on page 110.)

[IBM14] IBM. Ibm ilog cplex optimization studio cp optimizer user's manual, 2014. http://is.gd/7Y3TtZ. (Cited on pages 69 and 130.)

[Inc] Akamai Inc. Akamai facts & figures. http://www.akamai.com/html/about/facts_figures.html. (Cited on page 18.)

[Inc14] Cisco Inc. Cisco visual networking index: Forecast and methodology, 2013 - 2018. June, 2014. (Cited on pages 1, 3 and 100.)

[IP11] Sunghwan Ihm and Vivek S. Pai. Towards understanding modern web traffic. In *IMC*. ACM, 2011. (Cited on page 13.)

[JAL+13] F. Jokhio, A Ashraf, S. Lafond, I Porres, and J. Lilius. Prediction-based dynamic resource allocation for video transcoding in cloud computing. In *Euromicro PDP*, 2013. (Cited on pages 3 and 22.)

[JDMP11] Adele Lu Jia, Lucia D'Acunto, Michel Meulpolder, and Johan A. Pouwelse. Modeling and analysis of sharing ratio enforcement in private bittorrent communities. In *Proc. of IEEE ICC*, 2011. (Cited on pages 3 and 17.)

[JdV14] Vinay Joseph and Gustavo de Veciana. NOVA: qoe-driven optimization of dash-based video delivery in networks. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 82–90. IEEE, 2014. (Cited on page 25.)

[KBZ13] Dilip Kumar Krishnappa, Divyashri Bhat, and Michael Zink. Dashing youtube: An analysis of using DASH in youtube video service. In *38th Annual IEEE Conference on Local Computer Networks, Sydney, Australia, October 21-24, 2013*, pages 407–415. IEEE Computer Society, 2013. (Cited on pages ix, 24 and 26.)

[Kov12] A.M. Kovaks. Internet peering and transit. Technical report, Technology Policy Institute, 2012. (Cited on page 110.)

[KSC+12] Mehdi Kaytoue, Arlei Silva, Loïc Cerf, Wagner Meira Jr., and Chedy Raïssi. Watch me playing, i am a professional: a first study on video game live streaming. In *ACM WWW Conf.*, 2012. (Cited on pages ix, 15, 16 and 29.)

[LLSY11] Jimmy Leblet, Zhe Li, Gwendal Simon, and Di Yuan. Optimal network locality in distributed virtualized data-centers. *Computer Communications*, 34(16):1968–1979, 2011. (Cited on page 21.)

[LME06] T.M. Lenard and R.J. May (Eds.). *Net Neutrality or Net Neutering: Should Broadband Internet Services be Regulated*. Springer, 2006. (Cited on page 110.)

[LMT12] Stefan Lederer, Christopher Müller, and Christian Timmerer. Towards peer-assisted dynamic adaptive streaming over http. In *Packet Video Workshop (PV), 2012 19th International*, pages 161–166, May 2012. (Cited on page 25.)

[LSRT14] Jiayi Liu, Gwendal Simon, Catherine Rosenberg, and Géraldine Texier. Optimal delivery of rate-adaptive streams in underprovisioned networks. *IEEE Journal on Selected Areas in Communications*, 2014. (Cited on page 25.)

[LSYR11] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. Inter-datacenter bulk transfers with netstitcher. In *SIGCOMM*. ACM, 2011. (Cited on page 21.)

[LZG12] Feng Lao, Xinggong Zhang, and Zongming Guo. Parallelizing video transcoding using map-reduce-based cloud computing. In *IEEE ISCAS*, 2012. (Cited on pages 3, 22 and 23.)

[LZY+13] Song Lin, Xinfeng Zhang, Qin Yu, Honggang Qi, and Siwei Ma. Parallelizing video transcoding with load balancing on cloud computing. In *IEEE ISCAS*, 2013. (Cited on pages 3, 22 and 23.)

[MCAB12] Pietro Michiardi, Damiano Carra, Francesco Albanese, and Azer Bestavros. Peer-assisted content distribution on a budget. *Computer Networks*, 56(7):2038–2048, 2012. (Cited on page 19.)

[MPST14] Patrick Maillé, Karine Pires, Gwendal Simon, and Bruno Tuffin. How neutral is a CDN? an economic approach. In *CNSM*. IEEE, 2014. (Cited on pages 6 and 111.)

[MRT12] P. Maillé, P. Reichl, and B. Tuffin. Internet governance and economics of network neutrality. In A. Hadjiantonis and B. Stiller, editors, *Telecom. Eco. - Selected Results of the COST Action IS605 EcontTel*, pages 108–116. Springer Verlag, 2012. (Cited on page 110.)

[MSZ14] He Ma, Beomjoo Seo, and Roger Zimmermann. Dynamic scheduling on video transcoding for MPEG DASH in the cloud environment. In *Multimedia Systems Conference 2014, MMSys '14, Singapore, March 19-21, 2014*, pages 283–294. ACM, 2014. (Cited on page 26.)

[MWN+13] Usama Mir, Houssein Wehbe, Loutfi Nuaymi, Aurelie Moriceau, and Bruno Stevant. The zewall project: Real-time delivering of events via portable devices. In *VTC-Spring*. IEEE, 2013. (Cited on page 29.)

[NOSMW10] P. Njoroge, A. Ozdaglar, N. Stier-Moses, and G. Weintraub. Investment in two sided markets and the net neutrality debate. Technical Report DRO-2010-05, Columbia Univ., Decision, Risk and Operations Working Papers Series, 2010. (Cited on page 110.)

[NXLZ12] Di Niu, Hong Xu, Baochun Li, and Shuqiao Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *INFOCOM*. IEEE, 2012. (Cited on page 94.)

[ope12] Netflix open connect, 2012. `https://openconnect.itp.netflix.com/`. (Cited on page 19.)

[Pas12] Andrea Passarella. A survey on content-centric technologies for the current internet: CDN and P2P solutions. *Computer Communications*, 35(1):1–32, 2012. (Cited on pages 3, 17, 18 and 63.)

[PMS14a] Karine Pires, Sébastien Monnet, and Pierre Sens. POPS : service de diffusion de flux vidéos live prenant en compte la popularité. In *French Conference ComPAS*, 2014. (Cited on page 5.)

[PMS14b] Karine Pires, Sébastien Monnet, and Pierre Sens. POPS: a popularity-aware live streaming service. In *ICPADS*. IEEE, 2014. (Cited on page 5.)

[Pop14] Ben Popper. Justin.tv, the live video pioneer that birthed twitch, officially shuts down. The Verge, 2014. `http://is.gd/BzxLtr`. (Cited on page 31.)

[PS14] Karine Pires and Gwendal Simon. Dash in twitch: Adaptive bitrate streaming in live game streaming platforms. In *VideoNext CoNEXT Workshop*. ACM, 2014. (Cited on page 6.)

[PS15] Karine Pires and Gwendal Simon. Youtube live and twitch: A tour of user-generated live streaming systems. In *MMSys Data Sets*. ACM, 2015. (Cited on page 5.)

[QGL+09] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Jun Jim Xu, and Qi Zhao. Modeling user activities in a large iptv system. In *IMC*. ACM, 2009. (Cited on page 14.)

[R C14] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. (Cited on page 38.)

[Red14] Harish Reddy. Adapt Or Die: Why Pay-TV Operators Must Evolve Their Video Architectures, July 2014. Videonet white paper. (Cited on page 119.)

[RSW07] J Reichel, H Schwarz, and M Wien. Joint scalable video model 11 (jsvm 11). *Joint Video Team, Doc. JVT-X202*, 2007. (Cited on page 23.)

[Sau12] H. Saunders. IP interconnection: trends and emerging issues, 2012. (Cited on page 110.)

[SBP+15] Guthemberg Silvestre, David Buffoni, Karine Pires, Sébastien Monnet, and Pierre Sens. Boosting streaming video delivery with wisereplica. *TLDKS*, 2015. (Cited on pages 93 and 107.)

[Sch13] B. Schwarz. Content delivery networks 3.0. Technical report, CTOiC White paper, 2013. (Cited on page 110.)

[SdDF+12] Srikanth Sundaresan, Walter de Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Measuring home broadband performance. *Commun. ACM*, 55(11):100–109, 2012. (Cited on page 43.)

[Sha14] Samantha Sharf. Netflix eyes global streaming domination as it crosses 50 million subscriber mark. Forbes, 2014. `http://is.gd/9fTRND`. (Cited on pages 1 and 100.)

[SI11] Siqi Shen and Alexandru Iosup. XFire online meta-gaming network: Observation and high-level analysis. In *MMVE Workshop*, 2011. (Cited on page 15.)

[SKS09] Jan Seedorf, Sebastian Kiesel, and Martin Stiemerling. Traffic localization for p2p-applications: The alto approach. In *Proc. of IEEE Peer-to-Peer Computing (P2P)*, 2009. (Cited on page 18.)

[SL13] Ryan Shea and Jiangchuan Liu. On GPU pass-through performance for cloud gaming: Experiments and analysis. In *ACM Netgames*, 2013. (Cited on page 123.)

[SLW+09] Li Su, Yan Lu, Feng Wu, Shipeng Li, and Wen Gao. Complexity-constrained h.264 video encoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(4):477–490, April 2009. (Cited on page 23.)

[Sor] F. Sorensen. The norwegian model for net neutrality. `http://is.gd/HtQuuJ`. (Cited on page 110.)

[SSSL12] Michael Smit, Mark Shtern, Bradley Simmons, and Marin Litoiu. Partitioning applications for hybrid and federated clouds. In *CASCON*. IBM / ACM, 2012. (Cited on page 22.)

[Sta13] Twitch Staff. Service update: Major video system update and prague is online. Twitch: The Official Blog, Dec. 2013. `http://is.gd/PdqlZI`. (Cited on pages 4 and 9.)

[Sto11] Thomas Stockhammer. Dynamic adaptive streaming over HTTP -: standards and design principles. In *Proceedings of the Second Annual ACM SIGMM Conference on Multimedia Systems, MMSys 2011, Santa Clara, CA, USA, February 23-25, 2011*, pages 133–144. ACM, 2011. (Cited on pages 13, 25 and 104.)

[SYDN+10] Hosik Sohn, Hana Yoo, W. De Neve, Cheon Seog Kim, and Yong-Man Ro. Full-reference video quality metric for fully scalable and mobile svc content. *IEEE Transactions on Broadcasting*, 56(3):269–280, Sept 2010. (Cited on page 125.)

[TAP+14] Laura Toni, Ramon Aparicio-Pardo, Karine Pires, Gwendal Simon, Alberto Blanc, and Pascal Frossard. Optimal selection of adaptive streaming representations. *TOMM*, 2014. (Cited on pages 92 and 106.)

[TAS+14] Laura Toni, Ramon Aparicio-Pardo, Gwendal Simon, Alberto Blanc, and Pascal Frossard. Optimal set of video representations in adaptive streaming. In *ACM MMSys*, 2014. (Cited on page 121.)

[twi] `http://www.twitch.tv/`. (Cited on page 47.)

[VAJ+06] Eveline Veloso, Virgílio A. F. Almeida, Wagner Meira Jr., Azer Bestavros, and Shudong Jin. A hierarchical characterization of a live streaming media workload. *IEEE/ACM Trans. Netw.*, 14(1):133–146, 2006. (Cited on page 15.)

[VGN+12] Alex Borges Vieira, Pedro Gomes, José Augusto Miranda Nacif, Rodrigo Mantini, Jussara M. Almeida, and Sérgio Vale Aguiar Campos. Characterizing sopcast client behavior. *Computer Communications*, 35(8):1004–1016, 2012. (Cited on page 16.)

[Wei14] Nicolas Weil. The State of MPEG-DASH Deployment. Technical report, Streaming Media, Apr. 2014. http://is.gd/8Pcu78. (Cited on page 75.)

[WSW+14] Zhi Wang, Lifeng Sun, Chuan Wu, Wenwu Zhu, and Shiqiang Yang. Joint online transcoding and geo-distributed delivery for dynamic adaptive streaming. In *IEEE INFOCOM*, 2014. (Cited on pages ix and 23.)

[Wu03] T. Wu. Network neutrality, broadband discrimination. *Journal of Telecom. and High Tech.*, 2003. (Cited on page 110.)

[Xip14] Xiph.org. Xiph.org video test media, 2014. http://media.xiph.org/video/derf/. (Cited on pages 4 and 122.)

[YCWF11] Ming Yang, Jianfei Cai, Yonggang Wen, and Chuan Heng Foh. Complexity-rate-distortion evaluation of video encoding for cloud media computing. In *IEEE ICON*, 2011. (Cited on page 23.)

[YLQ+09] Hao Yin, Xuening Liu, Feng Qiu, Ning Xia, Chuang Lin, Hui Zhang, Vyas Sekar, and Geyong Min. Inside the bird's nest: measurements of large-scale live vod from the 2008 olympics. In *IMC*. ACM, 2009. (Cited on page 14.)

[YLZ+09a] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *ACM Multimedia*. ACM, 2009. (Cited on pages 2 and 19.)

[YLZ+09b] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *ACM Multimedia*. ACM, 2009. (Cited on page 64.)

[You13] YouTube. Youtube live introduction. YouTube Live Streaming Guide, Dec. 2013. http://is.gd/AwOyAx. (Cited on pages 4, 9, 29 and 36.)

[ZH12] Xiangyang Zhang and Hossam S. Hassanein. A survey of peer-to-peer live video streaming schemes - an algorithmic perspective. *Computer Networks*, 56(15):3548–3579, 2012. (Cited on page 16.)

[ZJY+09] Hui Zhang, Guofei Jiang, Kenji Yoshihira, Haifeng Chen, and Akhilesh Saxena. Intelligent workload factoring for a hybrid cloud computing model. In *SERVICES I*. IEEE, 2009. (Cited on page 22.)

[ZLAZ11] Jia Zhou, Yanhua Li, Vijay Kumar Adhikari, and Zhi-Li Zhang. Counting youtube videos via random prefix sampling. In *IMC*. ACM, 2011. (Cited on page 13.)