TÉLÉCOM BRETAGNE
UNIVERSITÉ RENNES 1

T H È S E

pour obtenir une

Habilitation à Diriger des Recherches

Présentée par

Gwendal SIMON

Massive Interactive Multimedia Services Over the Internet

Soutenue le 7 avril 2015

Devant le jury composé de :

Rapporteurs :	Walid DABBOUS	-	INRIA - École Polytechnique
	Nidhi HEGDE	-	Alcatel-Lucent Bell Labs
	Carlos Enrique PALAU SALVADOR	-	Univ. Politècnica València
Examinateurs :	Gerardo RUBINO	-	INRIA
	Laurent MASSOULIÉ	-	INRIA - Microsoft Research
	Dario ROSSI	-	Télécom ParisTech

Abstract

The Internet is now a support infrastructure for the entertainment industry. In particular, the demand for interactive, multimedia applications over the Internet has significantly grown over the past ten years, i.e. since I defended my PhD thesis. As a researcher in the field of multimedia and Internet, I found the last decade especially exciting and challenging. My contributions to a better understanding and, hopefully, a better implementation of large-scale, interactive, multimedia services have taken various forms, which I classify here into three categories: measurements, optimization, and system design. This manuscript describes two representative studies in each of these categories.

As for the measurement chapter, I present two campaigns of measurement related to gaming. In the first one, the motivation is to study the latency of cloud gaming applications, where the game engine is hosted in a datacenter and the output streamed to the end-users. In the second one, a popular platform allowing gamers to broadcast themselves playing (live gameplay), namely Twitch.tv, is dissected.

The optimization chapter deals with live streaming delivery. The first study aims at minimizing the waste of resources for systems based on multiple concurrent peer-to-peer overlays, while the second study addresses the reduction of cost for Content Delivery Networks having to deliver rate-adaptive, live, video streams.

The topic of the system design chapter is related to Content Centric Networks, where the main functions of the networks are based on the content name instead of the content location. I present here our proposal to augment the CCN protocol in order to better leverage cache storage inside routers. The second study describes an evaluation of performances of this protocol based on experiments in laboratory.

The manuscript ends with a chapter about research perspectives in the field. Although I doubt that it is possible to have a clear vision of what will the research look like in a horizon of ten years, especially in my field, I present some global research axis, which, in my opinion, are worth exploring in the next future years.

Contents

Chapter 1

Preamble

I defended my PhD thesis ten years ago. At that time, my research domains included peer-to-peer systems, mobile ad-hoc networks and large-scale virtual worlds. Today, these topics hardly get any attention from the academic world. Although most papers published in the early 2000s advocated that centralized systems would never scale, today's most popular services, which are used by billions of users, rely on a centralized architecture powered by data-centers. In the meantime, the open virtual worlds based on 3D graphical representation (e.g. Second Life) fell short of users while social networks based on static text-based web pages (e.g. Twitter and Facebook) have exploded. I do not want to blame myself for having worked in areas that have not proved to be as critical as they were supposed to be. Instead, I would like to emphasize that I work in an ever-changing area, which is highly sensitive to the development of new technologies (e.g. big data middleware), of new hardware (e.g. smartphone), and of new social trends (e.g. user-generated content).

I envy the scientists who are able to precisely describe a multi-year research plan, and to stick to it. I am not one of them. But I am not ashamed to admit that my research activity is mostly driven by short-term intuition and opportunities and that the process of academic funding directly impacts my work. Indeed, despite all of the above, I have built a research work, which I retrospectively find consistent. And more importantly, I have been relatively successful in advising PhD students and managing post-docs, all of them having become better scientists to some extents.

In very short, I have developed during the past ten years a more solid expertise in (i) theoretical aspects of optimization algorithms, (ii) multimedia streaming, and (iii) Internet architecture. I have applied these triple expertise to a specific set of applications: massive multimedia interactive services. I provide in this manuscript an overview of the activities that have been developed under my lead since 2006. It is a subset of selected studies, which are in my opinion the most representative of my core activity.

I hope you will have as much fun reading this document as I had writing it.

Chapter 2

Introduction

This thesis deals with the challenge of offering massive, interactive, multimedia services over the Internet. In this chapter, our goal is to introduce the main elements of context, to fix the terminology that we will use throughout the thesis, and to introduce the general bibliography (more relevant related works are studied in other chapters). We first give a short description of the landscape: the services, the video technologies, and the delivery infrastructure architectures. Then we give a general perspective of our research with regard to the scientific communities.

2.1 Targeted Services

The object of our research is the massive, interactive, multimedia services. We first go over each of these adjectives.

Services on the Internet should essentially be *massive*, in the sense that millions of people, not to say billions, may use the service. In turn, the scalability is a permanent concern for the service designers and developers. Throughout this thesis, we will recall that the massiveness of the services brings new problems, calls for new architectural decisions, and highlights the limitations of some traditional theoretical approaches for designing the services.

Most of today's popular services also integrate multimedia aspects. When Cisco claimed in 2010 that video would represent 90% of the whole Internet traffic in 2015 [38], it was not only a shock—the Internet has not been designed for video flows—but also an alarm about the sustainability of the trend. Semesters after semesters, reports have confirmed the sustained growth of the appetite for multimedia content, which is expected to continue beyond 2015 [2, 39].

Finally, we are interested in *interactive* services, where each user can have some kind of control on the experience. Interactive services are especially challenging on two aspects: (*i*) the interactivity requires short response times, so the servers that deliver the service, wherever they are on peers or in a data-center, should be located close to the end-users, and (*ii*) the interactivity is generally associated with personalized content, so massive delivery techniques based on broadcasting are inefficient.

During the last years, we have studied three services which are representatives of massive, interactive, multimedia services. First, **cloud gaming services**, also known

as on-demand gaming. It is a new video gaming application/platform. Instead of requiring end-users to have sufficiently powerful computers to play games, cloud gaming performs the intensive game computation, including the game graphics generation, remotely with the resulting output streamed as a video back to the end-users. Second, **User-Generated Content (UGC) live streaming services** where anybody can become a TV broadcaster. This promise has seen a surge of interest in the past years, pushed by new usages like crowdsourced journalism [70] and e-sport [46]. Finally, **time-shifted on-demand TV services**, where a program broadcasted from a given time t is made available at any time from t to $t + \delta$ where δ can be potentially infinite. The popularity of TV services based on time-shifted streaming has dramatically risen with the proliferation of Digital Video Recorder (DVR) devices at the client side. For many reasons, today's service providers would like to implement these services *in the cloud*.

2.2 Targeted Video Technologies

The heterogeneity of Internet video consumers has grown on two aspects. First the devices that are used to play the video range from smartphone to smart TV. Second the network connections range from cellular networks to fiber in residential area.

To address the heterogeneity of end-users, the researchers have worked for years on *scalable video coding*, which features mechanisms to adjust video quality to the network conditions. Despite the strengths of scalable video coding, the industry has adopted another technology, called **Adaptive Bit-Rate (ABR) streaming**.

The main idea behind ABR is to enable service providers to match video stream quality with any end-user, regardless of her device/network configuration. The service provider prepares several (different) video *representations* of the raw channel stream, each representation being characterized by a different resolution and a bit rate. At the other end, end-users request the available video *segments* that are the best match according to their actual network conditions at the time they should stream the segments. For example, the Netflix High-Definition (HD) videos were encoded into up to 14 representations [4] in 2012..

Several models have been recently proposed to standardize ABR streaming, including Dynamic Adaptive Streaming over HTTP (DASH) [82, 84] and WebRTC [1]. ABR technologies were implemented and deployed in the late 2000's without prior support from the scientific community and bypassing the standardization groups. Since 2010, scientists have developed an intense research activities to model and understand these complex systems where multiple parameters interplay. Our work is a modest part of these efforts.

2.3 Targeted Delivery Infrastructure Architectures

To deal with the demand for more interactivity and more multimedia content, service providers have been forced to upgrade their infrastructure. In the meantime, the service providers have also seen the benefits they can get from virtualizing their infrastructure and from delegating the delivery of content to external companies, in

particular Content Delivery Networks (CDNs). The traditional delivery architecture where the service provider owns a set of servers and delivers the content by itself is now limited to a handful of very big players. The now dominant architecture is the combination of multiple actors tied by commercial agreements and orchestrated by the service provider through virtualization techniques.

Service provides have several options to build their delivery infrastructure. Here is a short introduction to them.

Data-Center (DC) The most common way to deliver content is to use a DC, which is basically a large set of interconnected servers [13]. The DC can be either owned or rented by the service provider. In the former case, the owner can set the DC according to its needs, but the infrastructure has some fixed capacity limitations. In the latter case, the infrastructure can scale up and down on demand but the service provider has to deal with another actor (DC provider). Although a DC is an attractive, easy-to-manage infrastructure, it does not enable low response time for a large population of users because it is a centralized infrastructure where one DC location covers a wide area. As demonstrated by CDN, a more distributed infrastructure allows the deployment of servers closer to the end-users, so better average response time.

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. 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 [73]. However, various constraints have limited the deployment of P2P systems for commercial purposes. First, firewalls and Network Address Translator (NAT) still prevent many direct connections between users [44]. Second, P2P require users to install a program on their computers. Such a "technical", requirement can prevent users from utilizing 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 programming difficulties. Third, the service provider has a low control on the Quality of Experience (QoE) of users since it does not directly control the performances. Finally, the complexity of P2P system can increase the delay. A peer usually get data from multiple other peers. Even if the direct connection between two peers has a short latency, aggregating data from multiple peers requires synchronisation and buffering, which cause extra-delay (even though many initiatives have aimed at ensuring that peers connect preferentially with the other peers that are located in the same network [79]).

CDN In the recent years, CDNs have emerged as the privileged way for large-scale content delivery. 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. For a decade, the CDN providers have met the demand of two families of

players 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 CDN providers provide both scalability and flexibility, they deal with distribution complexities and they manage multiple operator references, all of these services at a unique selling point. Works such as [28, 54] confirm that edge-servers are used not only for serving static content but also for interactive multimedia services despite the weak computational capabilities of CDN edge servers.

To get the best of the three options, service providers commonly deploy hybrid architectures. For example Google uses multiple DCs [5], Netflix uses multiple CDNs [4], LiveSky deployed a composition of P2P assisted by CDN [91], and Twitch uses a private DC assisted by CDN [37]. In this thesis, our main focus is on CDN but other architectures can appear. Our choice to progressively migrate our research activities from P2P to CDN comes from the observation that, despite the intrinsic qualities of P2P, the CDN architecture has become the best option for service providers, with an handful of players being in charge of most video traffic over the Internet [23]. Note that we have studied P2P and multi-DC architectures in other works for video applications (in particular [50, 53, 61, 65, 66]).

2.4 Targeted Research Trends

The contributions that we will present all along this thesis are related to *applied research*. Yet, we had also developed activities on more "fundamental" research trends, which try to prefigure long-term evolution of the Internet and delivery infrastructures. These works do not target short term deployment but they are exploratory proposals.

In the recent years, we have mainly contributed to a worldwide research effort toward **Information-Centric Network (ICN)**. The main motivation for ICN comes from the observation that end-users are no longer interested in *where* is the content they are looking for. They are rather interested in the content itself. This basic principle is in contradiction with the current Internet implementation where each request should have a destination host. To address this mismatch, service providers have implemented techniques to redirect requests for a given content to the closest server, especially Domain Name System (DNS) redirection. The ICN research trend aims at correcting this mismatch in a clean-slate way, *i.e.*, by considering that it is possible to design a totally new network regardless of the existing infrastructure. Among the proposals in this direction, the one that has attracted the most attention so far is Content-Centric Network (CCN) [41]. The main concepts defined in CCN are simple enough to make it a perfect toy to study the network delivery of content in a more fundamental perspective. Due to its simplicity and the availability of various research tools, our activity around ICN has exclusively been related to CCN.

2.5 Outline of the Thesis

We will present a *selection* of our works in the remaining of this thesis. This selection covers the three main activities we have had in the past years. First, we identify some of the most critical characteristics of interactive multimedia applications (see Chapter 3 related to measurement studies). Second, we study them from an optimization perspective, where we analyze their theoretical limits (see Chapter 4 related to optimization studies). Finally we propose systems to be implemented on top of some promising delivery infrastructures (see Chapter 5 related to system design). For each chapter, we focus on only two contributions. More details are given now.

Measurements Today's popular online systems are so complex that a whole scientific community focuses on studying these systems and identifying their most prominent characteristics. We consider especially useful to contribute in this area with specific studies on emerging multimedia services. We highlight here two key studies.

- Our first contribution is a measurement campaign related to network latency. This work targets cloud gaming but it can be generalized to any massive interactive service. It deals with whether current cloud deployment can serve a population with fast response time.
- Our second contribution is a measurement campaign related to the popularity of UGC live streaming systems. The motivation is to characterize the contributors of user-generated live video services, and to identify the challenges that these services have to address.

Optimization Combinatorial optimization is the main theoretical tool we have used during the past ten years. We have dealt with optimization problem formulation and integer linear programs. We have especially applied combinatorial optimization to the development of ABR streaming systems. We present here two studies:

- Our first contribution is about the setting of video encoding parameters for massive video services. This study targets the phase before the delivery in itself, when the multimedia content is prepared to be delivered.
- Our second contribution is about minimizing the delivery cost of video streams in CDN. In this work, we aim at reducing the footprint of adaptive streaming systems, especially when infrastructure is underprovisioned.

System design Designing new services requires, among others, the definition of protocols, large-scale simulation campaigns, laboratory experiments, and software development. A part of our research work has dealt with developing and testing new proposals. We address here the implementaton of a time-shifted TV service in CCN. We present two representative contributions:

- Our first contribution is the design of a protocol for a better use of the cache storage inside the routers. Our main contribution is an admission

control policy, a strategy that is rarely explored in distributed caching systems.

- Our second contribution is a performance evaluation campaign based on real prototype development, protocol evaluation, and laboratory test.

Chapter 3

Measurements

In their vast majority, scientists in the area of networks and systems should deal with measurement papers, either because they conduct by themselves some measurement campaigns in order to better understand the characteristics of systems, or because they have to decipher details in existing measurement papers in order to justify some choices they do in their model and in their simulations. Since networks and systems are applied sciences, researchers also frequently validate proposals through measurements based on real implementations.

During the past five years, we have contributed to the scientific community with several studies directly related to measurements. In the following, we present two of them, which reflect our belief that services and delivery infrastructure should be considered together and not separately. Indeed, we present one measurement campaign on a delivery architecture (see Section 3.1) and another on a massive interactive multimedia service (see Section 3.2).

3.1 Latency in Hybrid Delivery Architectures

Latency is a key requirement for interactive services. In the case of multimedia services, two types of latencies have to be considered: *encoding (respectively decoding) latency*, that is, the time to compress (resp. decompress) the video output at the server (resp. end-user) side, and *network latency*, which is the delay in sending the user input and video output back and forth between the end-user and the server.

In the following, we take cloud gaming as a representative interactive multimedia service because the latency requirement is critical for this service. Indeed, past studies [24, 25, 42] have found that players begin to notice a delay of 100 ms [43]. Despite progresses in video encoding latency, at least 20 ms of the overall latency should be attributed to playout and processing delay at respectively client and server sides [12]. It means that 80 ms is the threshold above which the network latency begins to appreciably affect user experience, among which a portion of latency is unavoidable as it is bounded by the speed of light in fibre.

We focus on the network latency since the other latencies, especially the generation of game videos, have been studied in previous works [18, 42]. Our goal is to study the network latency of the three main delivery architectures that are usually considered

for latency-sensitive applications: a small multi-DC architecture (see Section 3.1.2), a large multi-DC architecture (see Section 3.1.3), and a hybrid DC-CDN architecture (see Section 3.1.4). We present here only a selection of this study. Full details can be found in [21, 22].

3.1.1 Measurement Settings

As emphasized in previous network measurement papers [29, 85], it is challenging to determine a representative population of real clients in large scale measurement experiments. For our measurements, we utilize a set of 2,504 IP addresses, which were collected from twelve different BitTorrent[1] swarms. These BitTorrent clients were participating in popular movie downloads. We use the *GeoIP* service to restrict our clients to the United States, which is the focus of this measurement study. Although 2,504 IP addresses represent a fraction of the total population in the United States, these IP addresses likely represent home users who are using their machines for entertainment purposes. Therefore, we believe that these users are a reasonable cross-section of those who use their computers for entertainment purposes, which includes gaming. We refer to this selected users as the *population*.

We consider three delivery infrastructure for cloud gaming services:

- **A Small Multi-DC Architecture**. We performed a measurement campaign on the Amazon EC2 infrastructure during May 2012. Although EC2 is one of today's largest commercial clouds, it only has three DCs in US.
- **A Large Multi-DC Architecture**. We use PlanetLab [14] nodes to serve as DCs. Since there are 44 PlanetLab nodes in US, we can evaluate the behavior of a larger, more geographically diverse multi-DC cloud infrastructure.
- **A DC-CDN Architecture**. Out of the 2,504 IP addresses collected in our measurement study, we select 300 of them to represent edge-servers in a CDN.

We use TCP measurement probe messages to determine latency between "servers" and clients. Note that we measure the round-trip time from the initial TCP handshake, which is more reliable than a traditional ICM *ping* message and less sensitive to network conditions. We use the median value from ten measurements to represent the latency between an end-host to a server (PlanetLab node or EC2) and we filter out the IP addresses that experienced too variable latency results.

3.1.2 Latency in a Small Multi-DC Architecture

The Amazon EC2 cloud offers three DCs in the US to its customers. We obtain a virtual machine instance in each of the three DCs. Every half hour over a single day, we measure the latency between each DC to all of the 2,504 clients. We say that an end-user is *covered* for a given latency when this end-user is connected to at least one of the three DC with a median latency below this latency target.

The ratio of covered end-users for various latency targets is depicted in Figure 3.1. Two observations can be made from this graph:

[1] http://www.bittorrent.com/

- *More than one quarter of the population experience a degraded quality of play from an EC2-powered cloud gaming platform.* The 80 ms threshold network latency yields a 70% coverage.
- *Almost 10% of the potential clients are essentially unreachable.* In our study, unreachable clients are clients that have a network latency over 160 ms, which renders them incapable of using an on-demand gaming service. This result confirms the measurements made by previous work [85].

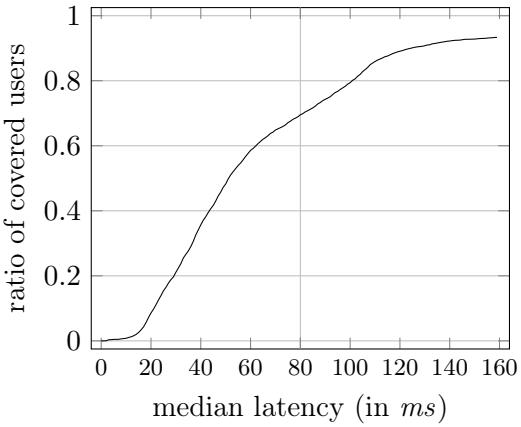


Figure 3.1 – Population covered by EC2 cloud infrastructure as a function of the median latency. The thin vertical line indicates the network latency threshold after which the experience is noticeably degraded for the end-user.

3.1.3 Latency in a Large Multi-DC Architecture

We then investigate the gain in population coverage when new DCs are added into the existing EC2 infrastructure. We utilize 44 geographically diverse PlanetLab [14] nodes in the United States as possible locations for installing DCs. We consider a cloud provider that can choose from the 44 locations to deploy an infrastructure containing k DCs. We will note it k-DC cloud infrastructure in the following. We determine latencies between clients and PlanetLab nodes with the same measurement process as for the previously described EC2 campaign. Afterwards, we determine the end-user coverage when using PlanetLab nodes as the component of a large multi-DC architecture.

We consider the most favorable deployment strategy for a cloud provider that wants to build a dedicated cloud infrastructure for interactive multimedia services. The network latency is the *only* driving criteria for the choice of the DC locations. For a given number k, we select the k-DC that yields the highest ratio of covered end-users among all k-subsets of DCs. It is thus an "optimal" multi-DC infrastructure with respect to average latency.

For cloud providers, the main concern is to determine the minimum number of DCs required to cover a significant portion of the target population. Figure 3.2 depicts

the ratio of covered users as a function of the number of DCs for two targets network latencies: 80 ms and 40 ms. In addition to our 80 ms initial threshold, we select 40 ms as a stricter requirement for games that require either a significant amount of processing or multiplayer coordination.

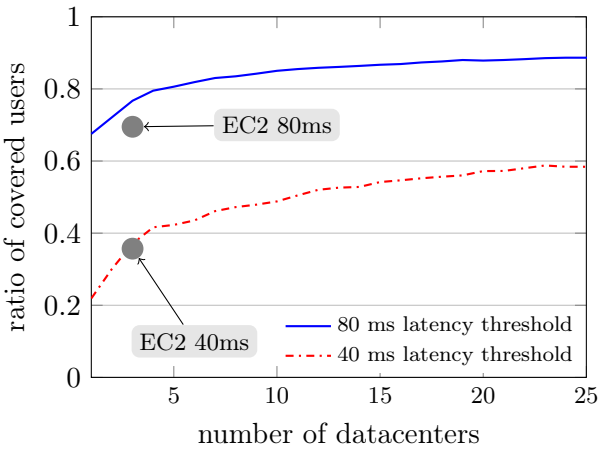


Figure 3.2 – Coverage vs. the number of deployed DCs

We observe that a large number of DCs is required if one wants to cover a significant proportion of the population. Typically, a cloud provider, which gives priority to latency, reaches a coverage ratio of 0.85 with ten DCs for a target latency of 80 ms. A 0.9 coverage ratio with a 80 ms response time is not achievable without a significant increase in the number of DCs (around 20 DCs). For more demanding games that have a lower latency requirement (e.g. 40 ms), we find that cloud provides exceedingly low coverage. Even if 20 DCs are deployed, less than half of the population would have a response time of 40 ms. Overall, the gains in coverage are not significant with regard to the extra-cost due to the increase of the number of DCs. Please note also that EC2 performs extremely well for the 40 ms threshold with a quasi-optimal coverage.

3.1.4 Latency in a Hybrid DC-CDN Architecture

Since the multi-DC hybrid solution has some significant shortcomings, we then explore the potential of a hybrid DC-CDN infrastructure to meet the latency requirements of on-demand gaming end-users. A BitTorrent client may be used to represent either an CDN edge-server or an end-user. Since we do not have control of our collected BitTorrent clients, we estimate client-to-client latency as follows. Suppose we wish to determine the latency between two end-users C_1 and C_2. We consider the end-user, say C_1 without loss of generality, that has the smallest latency threshold due to a fast connection to a PlanetLab P. The latency between C_1 and C_2 is the sum of P's latency to C_2 and a fuzzing factor that is between 0 ms to 15 ms. In other words, we assume that C_1 is somewhere around P, which is represented by the additional 0 ms to 15 ms latency.

We evaluate the effectiveness of a deployment or configuration by the number of

end-users that it is able to serve. In all of our experiments, we only model active users, and they are statically matched to either a datacenter or an edge-server. An end-user is served (or satisfied) if either its latency to a datacenter is less than its required latency, or it is matched to an edge-server that is within its latency requirement.

In Figure 3.3, we present the ratio of covered users by a small multi-DC architecture (Amazon EC2) and the hybrid DC-CDN architecture. We consider here an idealized system without constraints on client-CDN matching, so there is only one game, and each server at the CDN can serve an unlimited number of clients.

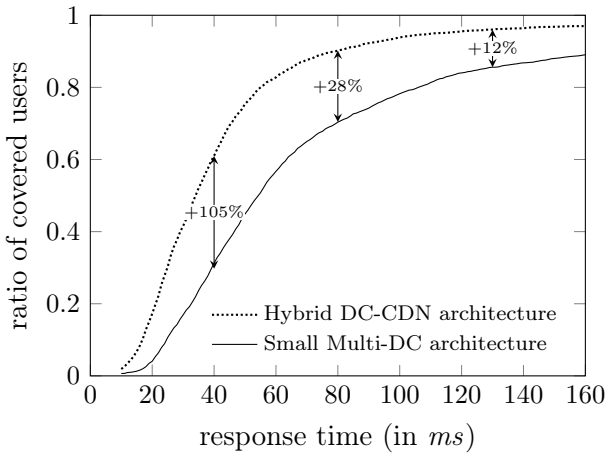


Figure 3.3 – Maximum achievable performances for an augmented infrastructure

These results demonstrate that significant gains can be achieved by adding some CDN edge-servers to a traditional multi-DC cloud architecture. The CDN allows to double the ratio of covered users whose target response time is below 40 ms and to achieve a 28% increase in the ratio of users for a 80 ms target response time. Our results in Figure 3.2 show that more than 20 datacenters are required to achieve a similar improvement.

3.1.5 General Observations from the Measurements

The results of our measurement study point to a hybrid DC-CDN infrastructure that combines existing DCs with CDN servers. Because CDN servers are in closer proximity to end-users, they are able to provide lower latency for end-users than distantly located cloud DCs. In addition, a hybrid DC-CDN infrastructure is more attractive than a multi-CDN because DC, which are less costly than CDN, can serve a significant fraction of users. Therefore DC-CDN is attractive for such demanding interactive, multimedia service.

Yet, there are still many challenges that need to be addressed. One challenge is to determine the selection of edge-servers that maximizes user-coverage. Unfortunately, this is an instance of the facility location problem which is NP-hard. Furthermore, since edge-servers cannot host an infinite number of games, due to physical limitations and cost considerations, another challenge is to strategically place games on edge-servers in order to achieve a maximal matching between end-users and edge-servers.

Solutions to these challenges are especially required in case of a growth of the number of concurrent gamers. We develop some solutions for these problems in [21, 22].

3.2 Popularity in Massive Live Streaming Services

UGC live streaming services have become popular in the beginning of the 2010's, driven by the rise of online game *screencasting* platforms [26]. Every month in 2013, around one million gamers have broadcasted themselves playing games live, and more than 40 millions people have watched these gameplay video channels [3]. This popularity has made the leading live game streaming platform, namely Twitch,[2] become the fourth largest source of US peak Internet traffic in February 2014 [30].

In Twitch we distinguish *broadcasters* and *viewers*. The broadcasters are registered gamers, who are in charge of one *channel*. We will interchangeably use the terms channel and broadcaster hereafter. A channel can be either *online* at a given time, which means that the broadcaster is broadcasting a gameplay live, or *offline* when the user in disconnected. A channel can alternatively switch from offline to online and vice versa. When a channel is online, we say that it corresponds to a *session*. The number of *viewers* watching a session can change over the time of the session. We illustrate in Figure 3.4 the evolution of the popularity of a given channel over time, this channel containing two sessions.

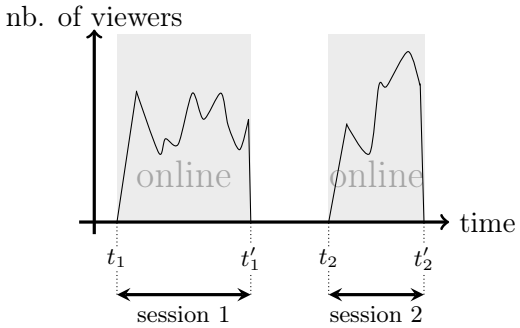
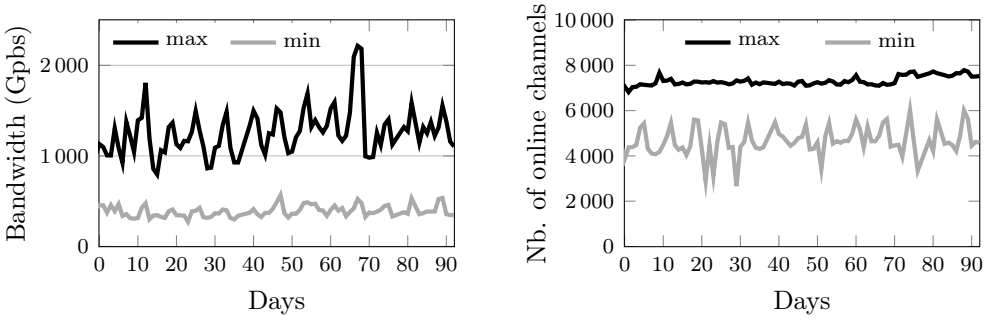


Figure 3.4 – A life in a channel

3.2.1 Twitch Dataset

Twitch provides an Application Programming Interface (API) that allows anybody to fetch information about the state of Twitch. We used a set of synchronized computers to obtain a global state every five minutes (in compliance to API restrictions) between January, 6th and April, 6th 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. We then filtered the broadcasters having abnormal behavior (*e.g.* no viewer over the three months or only one five-minute session). The dataset, containing more than five millions sessions, is available on a

[2] `http://twitch.tv`



(a) Bandwidth usage for live video delivery (b) Number of concurrent online channels

Figure 3.5 – Three months in Twitch

public website.[3] We have used this dataset in several published papers as a basis for the evaluation of live streaming systems [11, 68, 75]. We also shared the public release about this dataset in a conference [76].

3.2.2 Characterizing Twitch

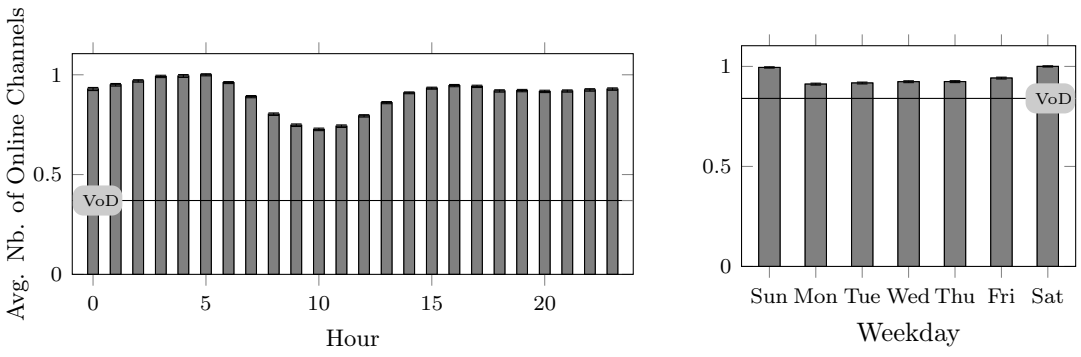
We give in the following a short analysis of the main characteristics of Twitch.

Delivery Needs We evaluate the overall bandwidth needed to deliver video channels to the viewers (we do not take into account the bandwidth required 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 bitrates of each session multiplied by the number of viewers for this session. We see in Figure 3.5a 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. For each incoming raw video, the DC should make sanity check, process all metadata, sometimes transcode the raw video into several video representations, and more generally prepare the stream to be delivered. We present in Figure 3.5b the average number of concurrent online channels, which is a metric for estimating the data-center dimensions. Between 4,000 and 8,000 concurrent sessions always require data-center processing. Such sustained incoming traffic requires a computing infrastructure that, to our knowledge, is unique in the area of live streaming.

We compute the average numbers of online channels per hour of a day (respectively per day of the week) over the three months in order to measure diurnal (respectively weekly) patterns. Broadcasters come from all over the world, we do not restrict this analysis to a particular country or continent. We show results in Figure 3.6, where we normalize the results so that the peak of the number of online channels is equal to 1. For the weekdays, the difference between the lowest number of online channels (0.9) and the peak (1) is not significant (we note it $0.90 : 1$). The main point to notice in

[3] `http://dash.ipv6.enstb.fr/dataset/videonext-2014/`



(a) Per hours (b) Per weekdays

Figure 3.6 – Average number and confidence interval of simultaneous online channels

Figure 3.6 is that the diurnal pattern is weaker than what has been observed on other UGC platforms. We indicate with a horizontal line the same lowest ratio as it was measured for the number of uploaded videos per minute in the YouTube VoD service (discussed in [16] and [17]). The diurnal difference on twitch is 0.65 : 1 although it is as low as 0.37 : 1 on YouTube Video on Demand (VoD). The main consequence is that Twitch does not necessarily concern about elasticity in its data-centers since a sustained incoming traffic from loyal broadcasters has to be processed.

We have to recall that live video streaming platforms essentially differ from other UGC services like VoD in the sense that the service depends on the activity of broadcasters *at every time*. According to our measures, Twitch can rely on loyal broadcasters to guarantee a 24/7 service.

Channel Popularity The distribution of popularity in UGC platforms typically follows the Zipf law [33]. We first need to check whether the popularity of Twitch broadcasters follows a Zipf law as well. We produce an approximation of the Zipf parameters using a fitting curve process on the R software. We validate the results of the approximation by calculating the Normalized Root-Mean-Square Deviation (NRMSD) between the real data and the fitting curve. The mean NRMSD value is 0.0095 with confidence intervals lesser than 1%. In other words, broadcaster popularity in Twitch follows a Zipf law. We then analyze the value of the α parameter, which says how much heterogeneous is the popularity of broadcasters. The larger is α, the more heterogeneous is the platform. Figure 3.7 shows the results obtained for the Zipf α coefficient. The horizontal line indicates the value found on YouTube [33]. What is relatively surprising here is the high value of α for Twitch. 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 encoded in various resolutions and bitrates. Our dataset confirms the intuitive idea that the better is

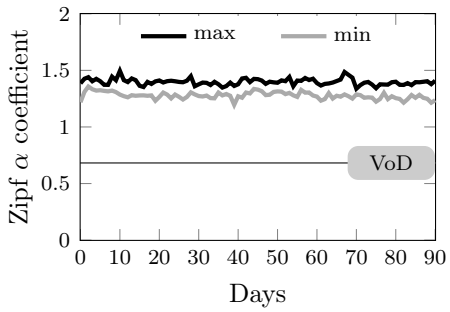


Figure 3.7 – Variation of the Zipf α coefficient over time

the quality of the video, the more popular it is. In the left part of Figure 3.8, we show the ratio of sessions with raw video at a given resolution, as well as the ratio of viewers on a channel with a video at this resolution. The 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. The right part of Figure 3.8 shows the Cumulative Density Function (CDF) of the bitrates of sessions for the three most popular resolutions. The bitrates of 720p and 1080p channels are significantly higher than for 480p channels. To emphasize the gap, we draw a thin vertical line at 2 Mbps. Only half of the video sessions at both 720p and 1080p have a bitrate lower than 2Mbps although such a bitrate is larger than 90% of the bitrates of 480p channels.

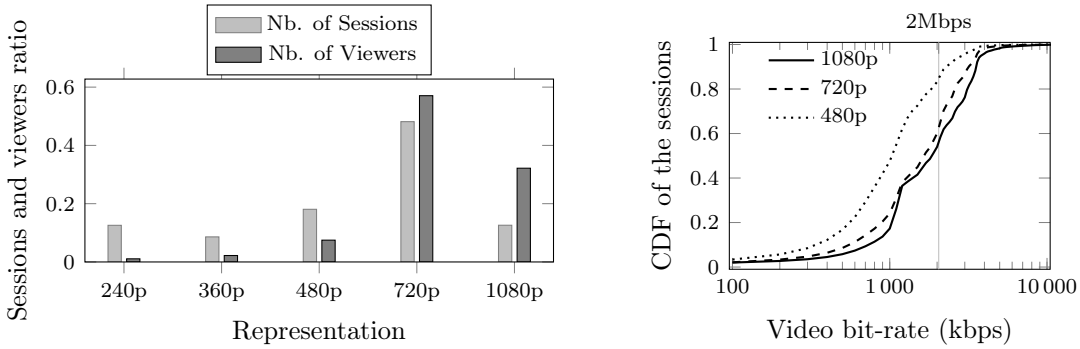


Figure 3.8 – On the left, number of sessions and viewers by video representation; On the right, CDF of the session bitrates

3.2.3 General Observations from the Measurements

To the best of our knowledge this is the first attempt to understand the behavior of game live streaming videos systems. Our findings bring a light on the specific characteristics of live streaming video services. Various open problems are still to be addressed. Among others, we highlight the implementation of ABR streaming technologies for such massive systems. With more than 7,000 concurrent channels, the computing needs for transcoding all these channels into multiple bit-rate streams are significant [11]. As shown in our study [75], a solution is to focus on only a small

fraction of popular, high-quality channels. But another open problem is then to detect as early as possible the channels that have good chances of becoming popular. The classification of channels according to multiple characteristics is a critical research challenge for these service providers.

3.3 Other measurement studies

We have highlighted in this chapter these two measurement studies since they reflect our work on both infrastructure and interactive multimedia services. We have made other studies related to measurement during the past years. Interested readers can refer to publications related to selection strategies in "peer list" [81], to time-shifted streaming services [65, 66], and to cross-domain traffic for CDN based on set-top-boxes [19].

Chapter 4

Optimization

The main theoretical approach we have used to improve interactive multimedia services is related to combinatorial optimization. As said in the preamble of this thesis, operations research is a topic that we have progressively learned to use since 2006. In this chapter, we present two representative studies, which show the diversity of approaches to solve optimization problems. In the first example (see Section 4.1), we present a polynomial-time algorithm that computes an optimal solution. Unfortunately, for most of the problems we have been working on, no polynomial-time algorithm can find the optimal solution (unless P=NP). In these cases, and typically in Section 4.2, our approach is to formally define an Integer Linear Program (ILP), which allows a generic solver to compute the optimal solution for relatively small problem instances. These two studies are thus representative of the theoretical works we have explored by the mean of optimization. Again, more details, especially how we make use of these theoretical developments for practical purposes, can be found in the related publications.

4.1 Minimize Resource Waste in Multi-P2P Architecture

Our measurement from the Twitch live streaming service reveals that the majority of the 7,000 simultaneous channels has very few viewers [76], so a CDN is not a cost-effective delivery infrastructure for these "unpopular" channels. An alternative is to use a P2P delivery architecture for each channel. Each P2P overlay contains the broadcaster (which we will call *source* in the following to stick with the usual terminology in use in the P2P community) and the viewers of the given channel (which we will call *peers* for the same reason). Thus, some users of the overall system are sources, which emit user-generated live videos, while the others are peers, which receive one or several videos and participate in the delivery of these videos. The delivery architecture as a whole is a *multi-P2P* architecture, which consists of multiple concurrent P2P live video streaming networks. Various proposals have studied such architecture [89, 90].

It is common for an end-user of e-sport live streaming systems to watch the actions of several players simultaneously, for instance several gamers from the same team. In that case, the end-user should *subscribe* to several P2P overlays. Since the peer

participates to several P2P overlays, it must share its upload bandwidth among several P2P overlays as well. For example, in Fig. 4.1, where the system contains three overlays, peer p_1 participates in overlay s_1 and s_2 while peer p_2 participates in all the three overlays. As a result, they must determine how to allocate their uplink bandwidth among these overlays. This threatens the availability of resources as it is now widely accepted that the resource bottleneck of P2P systems is the upload bandwidth of peers [60].

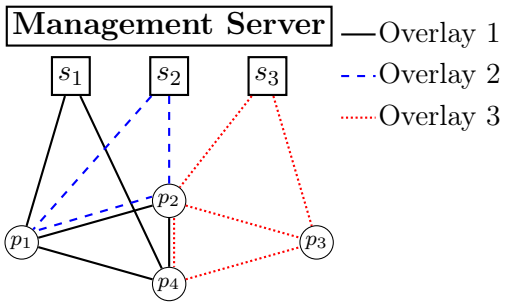


Figure 4.1 – Three sources in a multioverlay live video system.

To address this problem, we study bandwidth sharing strategies in multi-P2P architectures. We assume that a management server orchestrates the multiple overlays (Fig. 4.1). We focus on the provisioning of the overlays. The provisioning of an overlay is defined as the difference between the overlay demand (the amount of bandwidth required to serve all peers in this overlay) and the overlay capacity (the amount of bandwidth actually reserved). In streaming applications, the provisioning of an overlay has a direct impact on the video quality perceived by the peers. As resource deficit can lead to packet loss, peers in *underprovisioned* P2P networks experience video quality degradation [67]. Consequently, in the multioverlay context, resources can be wasted if they are allocated to overprovisioned overlays, although they could be allocated to underprovisioned ones.

Our goal is to minimize the waste of resources. We show that an optimal solution can be found in a time that is polynomial in the number of users. Thus, it is possible to compute an optimal bandwidth allocation for a large system in a reasonable delay. The main previous work [89, 90] proposes only an ILP, which requires a generic solver to compute the optimal solution, so it is restricted to small instances and it requires long computation time. In this chapter, we present our polynomial-time optimal algorithm. It is only a part of our contributions in the area ; the contributions related to this topic can be found in [61, 62].

4.1.1 Optimization Problem Definition

We first give the notations in Table 4.1.

Sources. The set of sources is denoted by S. A source s is associated with an overlay G_s, which contains the set P_s of all peers that have subscribed to this overlay. To avoid confusion, $s \notin P_s$.

Peer-to-Peer Streaming. The intra-overlay P2P streaming system is out of our scope in this study. Our system is independent of intra-overlay structures. Therefore,

P, S	set of peers, set of sources
G_s, P_s	overlay of source s and set of peers in G_s
B_p	upload capacity of a peer p
b_p^s	upload capacity reserved by p for G_s
$G(p)$	set of sources to which peer p subscribed
d_s	video bit rate of the video emitted by s
o_s	average overhead of P2P streaming protocol in overlay G_s
D_s, C_s	demand and capacity of G_s
Δ_s, Δ_s^r	provisioning and relative provisioning of G_s

Table 4.1 – Notations.

any state-of-the-art P2P live video streaming system can be used.

Peer Uplink Management. The set of all peers is denoted by P. We denote by $G(p)$ the set of sources from which peer p receives a video. The upload capacity of p is denoted by B_p while the upload capacity that p has reserved to serve video chunks in the overlay G_s is denoted by b_p^s. Clearly, $\sum_{s \in G(p)} b_p^s \leq B_p$.

Overlay Capacity and Demand. The capacity of an overlay G_s is denoted by C_s. It is equal to $\sum_{p \in P_s} b_p^s + B_s$, which is the aggregated upload bandwidth allocated from peers to G_s, plus the capacity of the source. The demand of an overlay corresponds to the smallest overlay capacity required to satisfy all peers in P_s. In a real system, the overhead resulting from the control traffic of P2P streaming protocols cannot be neglected. Therefore, the demand D_s of an overlay G_s contains two parts. The first part is the bandwidth required to stream the video to all peers, the other part is the bandwidth used by the streaming protocol. As a result, $D_s = |P_s| \cdot (d_s + o_s)$, where d_s denotes the bit rate of the video emitted by s and o_s is the average overhead in the overlay G_s.

Overlay Provisioning. The provisioning Δ_s of a given overlay G_s is the difference between its capacity C_s and its demand D_s, i.e., $\Delta_s = C_s - D_s$. An overlay is said to be underprovisioned when Δ_s is negative. The smaller the provisioning, the worse the video quality experienced by the peers. On the other hand, the overlay is overprovisioned when Δ_s is positive.

Overlay Relative Provisioning. The relative provisioning Δ_s^r of a given overlay G_s is defined as the overlay provisioning divided by the number of peers in the overlay, that is, $\Delta_s^r = \frac{\Delta_s}{|P_s|}$.

Our goal is to minimize the total underprovisioning. The resource allocation should ensure that no resource is allocated to overprovisioned overlays when it could have been allocated to underprovisioned ones. Let S^+ (respectively S^-) be the set of sources (overlays) with a positive (respectively negative) provisioning. We look for an uplink sharing among the overlays such that the total underprovisioning is minimum. Hence, our goal is to minimize $\sum_{s \in S^-} |\Delta_s|$, which is the resource waste.

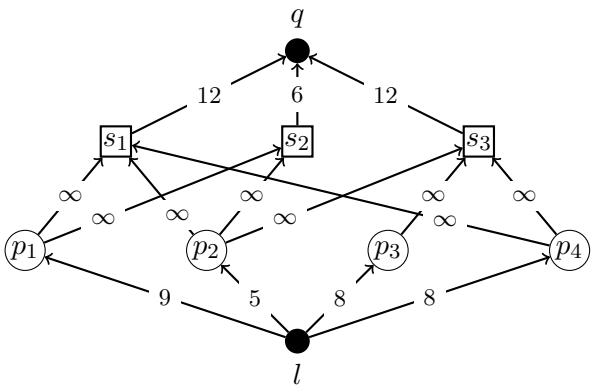


Figure 4.2 – Example of the bipartite flow network model. Numbers in the arrows are capacities

4.1.2 Our Solution: a Polynomial-Time Algorithm

In order to solve the bandwidth allocation problem, we build an abstract structure, which is a bipartite flow network $N = (V, E)$. This bipartite flow network model is the basis of our proposals. For example, Fig. 4.2 shows the bipartite graph related to the scenario of Fig. 4.1.

The set V contains a virtual fountain l, a virtual sink q, the set P of all peers in the system, and the set S of all sources. Thus $V = P \cup S \cup \{l, q\}$. The set of directed edges E gives the source-peer relationship. It includes three subsets. The first one, $E_1 = \{(l \to p) : p \in P\}$, contains edges from the fountain to each peer p with a maximum capacity of B_p. The second one, $E_2 = \{(p \to s) : p \in P, s \in G(p)\}$, contains edges from p to s if p subscribes to s with infinite maximum capacity. The third set, $E_3 = \{(s \to q) : s \in S\}$, contains edges from each source s to the sink with a maximum capacity equal to $D_s - B_s$, the overlay demand minus the source capacity.

The flows on edges E_2 represent bandwidth allocation from peers to sources. The capacity of E_1 indicates the limitation a peer can reserve, whereas the capacity of E_3 shows the demand of overlay.

Proposition 1 *The total underprovisioning $\sum_{s \in S^-} |\Delta_s|$ is minimum if and only if the maximum flow is achieved in the flow network.*

Proof. We denote by $f_{s,q}$ the flow on the arc $(s \to q)$. The cut-set between $V \setminus \{q\}$ and $\{q\}$ bounds a flow $|f| = \sum_{s \in S} f_{s,q}$. For each source s, $|\Delta_s| = D_s - B_s - f_{s,q}$. For a source s in S^+, $D_s - B_s - f_{s,q}$ is equal to zero because the flow $f_{s,q}$ cannot be greater than $D_s - B_s$. Thus,

$$\sum_{s \in S^-} |\Delta_s| = \sum_{s \in S} (D_s - B_s - f_{s,q}) = A - \sum_{s \in S} f_{s,q}$$

where A is a constant. Then, minimizing $\sum_{s \in S^-} |\Delta_s|$ is equivalent to maximizing $\sum_{s \in S} f_{s,q}$. Moreover, (i) edges from l to P provide the system with all capacities $C = \sum_p B_p$, (ii) edges from P to S follow the rule of the bandwidth allocation.

Hence, the overall underprovisioning $\sum_{s \in S^-} |\Delta_s|$ is minimized if and only if the flow is maximized. □

The max-flow problem has been extensively studied in the literature. The Goldberg-Tarjan *preflow-push* algorithm [32] is one of the most famous algorithms for this problem. This algorithm in a bipartite graph is further discussed in [8].

4.1.3 Remarks about the Minimization of Resource Waste

We have presented in this Section only the main algorithm. When the system is globally overprovisioned, our algorithm makes sure that all peers receive high quality videos. When the system is underprovisioned, we need another algorithm to manage the resource deficit. Various policies (or strategies) can be defined: for example minimize the number of underprovisioned channels, prioritize popular channels, and prioritize fee-paying users. In our publications, we have presented an algorithm based on a minimum-cost maximum flow, for which various cost functions can be defined. This algorithm finds allocations that minimize the waste of resources and correspond to the best allocations with respect to the policy of the service provider.

Overall, this study has strong ties with practical applications. We have comprehensively evaluated the (theoretical and practical) performances of our solution, and then this algorithm was partly implemented into a real system that has been tested by dozens of users in a collaborative project. This is conform to our approach, which is to mix applied and theoretical research objectives. All details are in [61, 62].

4.2 Minimize Live Stream Delivery Cost in CDN

The techniques used by CDN providers to deliver terabits of data per second are mainly exposed by reverse-engineering studies [6, 88]. The previous theoretical works related to live streaming in CDNs [7, 9, 10, 72] have highlighted the main characteristics of these networks, in particular the 3-tier topology (origin servers, reflectors and edge-servers), and the restriction on the upload capacity of the equipments. A major concern is the sharp growth in the volume of video traffic [23], and the capacity problem that this growth produces [40] with regard to the limited outbound rate of inner CDN equipments. In addition, the development of DASH accelerates the stress on the CDN since adaptive streaming techniques put much stress on the infrastructure, since for a single *video channel* the whole set of representations (with an aggregated bit-rate over 30 Mbps for a regular HD raw stream) should be delivered to the edge-servers.

The streaming capacity of networks has been addressed in a series of works [51, 80, 95]. These works aim to determine the maximum bit-rate that can be delivered to all nodes in a network. There algorithms, mostly based on network coding, obtain near-optimal performances in terms of bandwidth utilization [71]. Unfortunately, these solutions are unrealizable in a CDN due to three main reasons. First, they rely on heavy computations which are intractable in the CDN hardware. Second, the model used in these works is idealized since it assumes one infinitely divisible data stream, which has to be either delivered in its entirety, or not delivered at all. However, the

data that has to be delivered is a large set of distinct non-divisible streams (either video representations, or *bundles* of video representations); some of them can be delivered independently from the others. Third, since video channels may have only local popularity, not all edge servers are interested in receiving all representations of all channels. The strength of the CDN is to deliver the right content to the right edge-server.

We have proposed *discretized streaming*, which is a more suitable model for delivering multiple live video channels in modern CDNs. The main challenge is to determine the delivery scheme that maximizes the number of delivered streams in a 3-tier network, which is constrained by the capacity of its inner equipment.

In the following, we give a formal formulation of the general problem for this model and prove that it is NP-complete. We have derived some practical algorithms, which match real implementation issues, and we have done large-scale simulation campaigns. All details about these results can be found in related publications [63, 64].

4.2.1 Optimization Problem Definition

A CDN is composed of a set of communication devices and a set of directed communication links. There are three types of communication devices, also referred to as nodes, in a CDN: a relatively small number of origin servers (in short origin), a medium size network of reflectors, and a large number of edge servers. Each of these types is responsible for one of the phases of live stream delivery as outlined below.

- **Phase I: Transcoding** – The raw live stream data is delivered by the content provider to the origin nodes. These nodes are responsible for transcoding it into a set of live representations and then forwarding the representations to the reflector nodes.
- **Phase II: Multiplication** – The reflectors multiply the received stream data and forward the stream data to the edge servers. Note that the origins themselves cannot achieve the required output capacity to deliver the stream to all edge servers, and thus reflectors are used to increase the fan-out of the origin nodes.
- **Phase III: Delivery** – The edge servers receive the representations of the live stream and offer them to the clients inside their respective ISPs.

The topology of a CDN (see Figure 4.3) is modeled by a directed graph $G = (V, E)$, where V represents the communication devices, and E represents the communication links. Let $V_S, V_R, V_E \subset V$ be the set of origins, reflectors and edge servers, respectively. There are only three types of possible connections in E: E_{SR}, E_{RR}, and E_{RE}, defined as

$$E_{SR}=\{(u,v) : u \in V_S, v \in V_R\}$$
$$E_{RR}=\{(u,v) : u, v \in V_R\}$$
$$E_{RE}=\{(u,v) : u \in V_R, v \in V_E\}.$$

These three subsets essentially correspond to the three phases described above. That is, E_{SR} connects origins to reflectors, E_{RR} allows communication between reflectors, and E_{RE} delivers the various representations of the live stream to the edge servers.

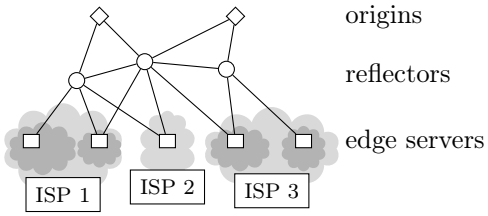


Figure 4.3 – Our model of a CDN network

The live streams consist of l different channels. The raw video of each channel is transcoded into k representations, where the bit-rate of the i-th representation, $1 \leq i \leq k$, is λ_i. For simplicity of notation hereafter we denote by $[m]$ the integer interval ranging from 1 to m. Also, let d_{ij} be the i-th representation of the j-th channel, $i \in [k]$, $j \in [l]$.

When the CDN builds a stream delivery scheme, the delivery of a representation d_{ij}, $i \in [k]$, $j \in [l]$, from the origin nodes to the edge servers is carried out at a given time through a set, $\mathcal{T}_{ij}$, of node-disjoint subtrees of G. Each tree in $\mathcal{T}_{ij}$ has one of the origin nodes as its root and edge servers as its leafs. We denote by T_s^{ij} the tree of d_{ij} that is rooted at $s \in V_S$. We also refer to this tree as the *delivery tree* rooted at s. For convenience, let $V(T)$ and $E(T)$ denote the node and edge sets of tree T, respectively.

Note that every *forwarding node* v, either origin or reflector, can participate in the delivery of representations. However, for any representation d_{ij}, $i \in [k]$, $j \in [l]$, v can be a part of only a single delivery tree in $\mathcal{T}_{ij}$ (otherwise the delivery would be sub-optimal because the node v does not need to receive twice the same stream). In addition, every forwarding node $v \in V_S \cup V_R$ is also limited by the total outbound bit-rate (capacity) it can support, $c(v)$. Let $D(v)$ be the set of representations forwarded by v, $v \in V_S \cup V_R$. Then,

$$\sum_{i \in [k]} \lambda_i \cdot |\{j : d_{ij} \in D(v)\}| \leq c(v).$$

Like all previous works (*e.g.),* [7, 96]), we consider that the outbound capacity of equipment is the only resource constraint in the system.

Ultimately we would like every edge server to receive all the representations it requires. This however might not be possible due to the outbound capacity constraints at the forwarding nodes, and thus the CDN may support the delivery of only a subset of representations for each edge server. In order to identify the preferences of edge servers in respect to the available representations and evaluate the performance of the proposed solutions we define a *utility score function* $\alpha_u(\mathbf{X_u})$ at each edge server $u \in V_E$ as follows:

$$\alpha_u(\mathbf{X_u}) = \sum_{i \in [k]} \sum_{j \in [l]} \alpha_u^{ij} x_u^{ij},$$

where α_u^{ij} is the utility score that edge server u assigns to representation d_{ij} and $\mathbf{X_u}$ is an indicator matrix of size $k \times l$ such that the element in the i-th row and j-th column, x_u^{ij}, has a value of 1 if d_{ij} is received by u and 0 otherwise.

Our objective is to study the *Maximum Average Utility Score* (MAUS) problem, which essentially is the maximization of the *average* utility score function of the edge servers, as summarized below.

Problem 1 (MAUS) *Given the topology and capacity constraints of a CDN, find delivery tree sets,* $\{\mathcal{T}_{ij}\}_{i\in[k],j\in[l]}$*, such that* $\sum_{u\in V_E}\alpha_u(X_u)$ *is maximized.*

Let DMAUS be the decision version of the MAUS problem.

Problem 2 (DMAUS) *Given topology and capacity constraints, and a real number* B*, do there exist delivery tree sets,* $\{\mathcal{T}_{ij}\}_{i\in[k],j\in[l]}$*, such that* $\sum_{u\in V_E}\alpha_u(X_u)\geq B$*?*

The proof that DMAUS is NP-hard can be found in [63].

4.2.2 Formulation as an ILP

We define two new variables, y and h. Let $T_s^{ij}\in\mathcal{T}_{ij}$, $i\in[k]$, $j\in[l]$, be a delivery tree. Then, for every edge $(u,v)\in E$, y_{uv}^{ijs} is an indicator variable such that:

$$y_{uv}^{ijs}=\begin{cases}1 & \text{if } (u,v)\in E(T_s^{ij}),\\ 0 & \text{otherwise.}\end{cases}$$

For nodes $u,v\in V$ such that $(u,v)\notin E$ we define $y_{uv}^{ijs}=0$. For every node $v\in V$, h_v^{ijs} is an upper bound on the depth of v in T_s^{ij}, i.e.

$$h_{uv}^{ijs}=\begin{cases}\geq \text{depth of } v \text{ in } T_s^{ij}, & \text{if } (u,v)\in E(T_s^{ij}),\\ =\infty, & \text{otherwise.}\end{cases}$$

To ease the notation, let us define $I_v^{ijs}(U)$ to be the sum of y variables that correspond to incoming edges into $v\in V$ from the nodes in $U\subseteq V$, i.e.

$$I_v^{ijs}(U)=\sum_{u\in U}y_{uv}^{ijs}.$$

Similarly, let $O_v^{ijs}(U)$ be the sum of y variables that correspond to outgoing edges from v to nodes in U, i.e.

$$O_v^{ijs}(U)=\sum_{u\in U}y_{vu}^{ijs}.$$

In the ILP formulation (which is shown in the manuscript in a the next page), we omit the use of set membership indication $\in$ for the main notations. Whenever we write $\forall i$, $\forall j$, $\forall s$, $\forall r$, $\forall u$, and $\forall v$, we imply $\forall i\in[k]$, $\forall j\in[j]$, $\forall s\in V_S$, $\forall r\in V_R$, $\forall u\in V_E$, and $\forall v\in V$, respectively.[1]

The constraints in (4.2) ensure that the indicator variables x have non-zero values only if there are incoming edges in the respective trees. Constraints in (4.3)-(4.4) guarantee that every node has only one parent in every delivery tree. Cycles are avoided in (4.7)-(4.8). The capacity restrictions are enforced in (4.5)-(4.6). Finally,

[1] We use i, j, s, r, u, and v to refer to representations, channels, origins, reflectors, edge servers, and general nodes, respectively.

ILP formulation: MAUS

$$\textbf{max.} \quad \sum_{u \in V_E} \sum_{i=1}^{k} \sum_{j=1}^{l} \alpha_u^{ij} x_u^{ij} \tag{4.1}$$

$$\textbf{s.t.} \quad x_u^{ij} \leq \sum_{s \in V_S} I_u^{ijs}(V_R) \qquad \forall i,j,u \tag{4.2}$$

$$I_r^{ijs}(V_S \cup V_R) \leq 1 \qquad \forall i,j,s,r \tag{4.3}$$

$$I_u^{ijs}(V_R) \leq 1 \qquad \forall i,j,s,u \tag{4.4}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{l} O_s^{ijs}(V_R)\lambda_i \leq c(s) \qquad \forall s \tag{4.5}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{l} \sum_{s \in V_S} O_r^{ijs}(V_R \cup V_E)\lambda_i \leq c(r) \qquad \forall r \tag{4.6}$$

$$h_s^{ijs} = 0 \qquad \forall i,j,s \tag{4.7}$$

$$h_r^{ijs} + 1 - h_v^{ijs} \leq |V|(1 - y_{rv}^{ijs}) \qquad \forall i,j,r,v \tag{4.8}$$

$$O_r^{ijs}(V_R \cup V_E) \leq |V|(I_r^{ijs}(\{s\} \cup V_R)) \quad \forall i,j,s,r \tag{4.9}$$

$$I_r^{ijs}(\{s\} \cup V_R) \leq O_r^{ijs}(V_R \cup V_E) \qquad \forall i,j,s,r \tag{4.10}$$

in (4.9)-(4.10) we require that reflector nodes have outgoing edges in delivery trees iff there is an incoming edge.

This formulation as an ILP can be implemented in a generic solver like CPLEX. That is, when a CDN wants to know what is the best possible delivery scheme for a given population and a given set of channels, it is possible to determine the exact optimal delivery scheme. This formulation thus allows the CDN to evaluate how far from the optimal are the current delivery schemes it implements.

4.3 Other Optimization Studies

What we presented in this Chapter is representative of our activities related to optimization: formal definition of problems that industrial partners can face, and design of solutions based on either fast optimal algorithms or the definition of ILP with proof of NP-completeness. In parallel of these two studies, we have studied some other topics, including another tree packing problem for telco-CDN [97], scheduling and optimization for heterogeneous wireless networks [31], multi-sink data gathering in sensor networks [34–36], virtual DC embeddings into real DCs [78, 94], and optimization of encoding parameters in adaptive streaming [11, 86, 87].

Chapter 5

System Design

Until 2006, we have exclusively focused on the conception and the realization of distributed, networked systems, such as the Solipsis system [47, 48]. Since 2008, our activities related to the design of new systems have been less intense, to the profit of optimization and measurements as shown in previous chapters. We have however been active in the emerging field of ICN. Our contributions related to caching in CCN have been especially visible. We present in this Chapter two scientific contributions that are related to what we call system design: a new protocol for a network of caches, and a large-scale simulation campaign conducted in our laboratory.

5.1 Cooperative Cache Admission Policy in CCN

One of the motivations behind CCN [41] and other ICN proposals is to exploit the storage resources of the equipments in the network. Manufacturers have released new generations of Internet routers, which are able to store and cache content in a reasonably large storage area. We refer to such a router as a *Caching Router* or abusively *cache* in short. The term *in-network caching* encompasses all technologies that are related to the implementation of a caching strategy on a potentially large set of inter-connected caches. The vast majority of studies related to caching focuses on *replacement policies*, which aim at deciding which cached data to eject from the cache in order to get room for caching a new data. The other policy to implement is an *admission policy*, which is to decide whether a non-cached data should be cached. In any case, the caching policy should be as simple and lightweight as possible. Both replacement and admission policies should respect two requirements. First, they should be completely distributed. A cache takes a caching decision by using only local information with minimum control traffic. Second, they should be able to run on machines with restricted hardware capabilities. Modern routers have to handle traffic at Gbps, and no extra-resource is available to sustain the implementation of complex cooperative caching policies.

We have proposed in [56, 58] a new cooperative caching policy, which is especially designed for videos, more generally for objects that contain many segments accessed in a consecutive manner. We assume each segment is identified by an integer. Our caching policy is merely an admission policy: each cache does not cache all the seg-

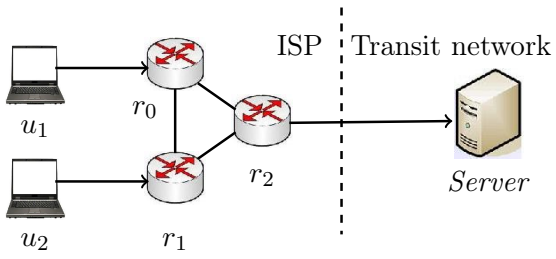


Figure 5.1 – Two end-users u_1 and u_2 requesting the same movie

ments that it routes, but only a part of them. A cache is associated with a *label*, which is a positive integer smaller than a fixed integer k. Only the segments having an identifier, modulo k equal to the said label are considered for being stored in the cache. For the replacement policy, caches use the traditional Least Recently Used (LRU) policy.

Let us illustrate by an example in Figure 5.1 the limitations of implementing LRU with no admission policy (*i.e.*, all data can be stored in all caches). Assume that both end-users u_1 and u_2 watch the same video with a delay corresponding to eight *segments* (a video segment is usually in the order of dozens of seconds), and that each router has a cache capacity of five segments for this video. We show in Table 5.1 the status of the storage areas of the three caches at a given time (when user u_1 plays segment 18 while u_2 plays segment 10). When no admission policy is implemented, the next request from u_2 has to be forwarded to the server because no cache stores segment 11. The lack of a smart admission policy results in an inefficient caching strategy with redundant data stored on adjacent routers. For our policy, the next request from u_2 can be fulfilled by one of the three caches within the Internet Service Provider (ISP) network. The consequence is a reduction of the number of requests that have to pass through the transit network. The main idea is that we eliminate redundancy among adjacent caches, and therefore, we increase the amount of distinct segments that are stored in the in-network cache.

Caching Policy		**Segments stored in caches**		
admission	replacement	r_0	r_1	r_2
none	LRU	$14, 15, 16, 17, 18$	$6, 7, 8, 9, 10$	$16, 9, 17, 10, 18$
our proposal	LRU	$11, 5, 14, 8, 17$	$12, 6, 15, 9, 18$	$4, 13, 7, 16, 10$

Table 5.1 – State of the three caches represented in Figure 5.1, for two users, u_1 and u_2 playing segment 18 and 10, respectively. The replacement policy is LRU

All the details of our caching policies can be found in [56, 58]. In short, the attribution of labels to caches is a variant of a domatic partitioning problem. We proposed a practical distributed algorithm to make each cache decide its label according to the labels of its closest caches, the distance being here set by the network operators (for example the distance can be set according to the latency). In this Chapter, we present two contributions, which are representative of our scientific efforts for the design of new systems. First, we present the augmented CCN protocol that implements our

policy. Second, we show the analysis of the performances of this new protocol.

5.2 Conception of an Augmented CCN Protocol

We present hereafter the implementation of our proposal into CCN. Changes include two additional tables integrated in a cache router, a slight modification in the message forwarding scheme and two novel message types named *cooperative interest* and *piggyback interest*.

5.2.1 New Tables in CCN

Our cooperative caching strategy requires two new tables.

- **Cooperative Router Table (CRT)**: every cache stores some data related to the $k-1$ cache routers that are the closest to itself. A CRT include three fields: the label, the identifier of the cache and the interface. Thus, every cache knows where to redirect an interest or forward a video segment.
- **Cooperative Content Store (CCS)**: a cache stores the names and the sequence numbers of all the segments that can be found in the storage area of its closest caches. When an interest arrives, the preference of the four prefixes matches is a Content Store (CS) match to a CCS match to a Pending Interest Table (PIT) match to a Forward Interest Table (FIB) match.

At the very beginning of the deployment of our protocol, both CRT and CCS tables are empty. Once a cache router i determines its label, it advertises the label to all caches in its 2-hop network neighbourhood. Thus, these latter can register the label, the identifier of i and the interface toward i. In the future, at the reception of a chunk matching the label of i, these cache routers will forward the chunk to i according to their CRT. In the meantime, the sequence number of the received chunk is stored in their CCS.

5.2.2 Reception of a Data Message

When a cache i with label c_i receives a segment s, it first forwards s according to the corresponding PIT entry. Then it has to make a decision (whether to cache it or not) based on c_i and the identifier of segment s. We give the pseudocode in Algorithm 1. In case of a match, our admission policy lets the cache manage the segment as usual (here a traditional LRU replacement policy). What is more interesting is when the admission policy rejects the segment for caching.

In case of match failure, the cache i first finds in its CRT the router j whose label c_j matches segment s ($s \mod k = c_j$). See line 8. Then, i checks its PIT to see whether the interface to j is a matching entry. In the case that no PIT match is found for the interface to j, the cache i sends a cooperative interest (*c-interest*) with the segment s out of the interface pointing to j according to CRT (line 10). Thus, the cache j receiving this c-interest gets contents that pass nearby. See Algorithm 2 for the treatment of c-interest messages.

Once the segment s is transfered to j, the cache i adds s in the CCS Table (line 11), so that future interests requiring the same segment will be forwarded to j, but no longer according to the FIB. Finally, i cleans up the PIT entry. Although two extra rounds of message exchanges are yielded by the cooperative caching protocol, there is no side-effect on the responding delay to the users. To prevent a broadcast storm, each data packet carries a random *nonce*. A duplicated packet with the same nonce is immediately discarded.

Algorithm 1: At reception of a *Data* message

Require: {i **receives s:**}

```
1:  forward s according to PIT
2:  if (s mod k = c_i) then
3:     if cache is not full then
4:        cache s
5:     else
6:        replace least recently used segment with s
7:  else
8:     determine a nearby cache j such that s mod k = c_j
9:     if interface to j is not in PIT then
10:       send c-interest with s
11:    add the identifier of s in CCS
12: delete PIT entry for s
```

5.2.3 Table Consistency and Reception of an Interest Message

The CS of a given cache i should ideally be *always* consistent with the CCS tables of all caches that cooperate with i. In particular, when an CS entry of i is discarded by the replacement policy, the corresponding entry in the CCS of a cache j should also be deleted, otherwise interests for the eliminated content may be lost in the forwarding process. For example, if j receives an interest requiring segment s, it finds a match to i in its CCS. Let us assume that segment s in i has been discarded, so cache i forwards the interest following its FIB entry. If j is an intermediate cache between i and the data source, the interest will be regarded as a duplicated one, and discarded by j.

To both maintain consistency and minimize the number of control messages, we use a piggybacking approach to carry the control information. The idea is to augment interest messages with *p-interest*, which allows a cache to claim that it does no longer store a given segment. A cache i with label c_i acts as shown in Algorithm 2 when an interest for segment s is received. There are two cases:

- i is in charge of the requested segment s ($s \mod k = c_i$). Then the message can be either a *c-interest*, carrying a segment s to be cached, or a standard *interest* message, which trigger the regular CCN process. When no match is found, the cache i changes the message into a *p-interest*, which indicates to all

Algorithm 2: Action upon Receiving *Interest*

Require: {i **receives** ***interest*** **for** s**:**}

```
1: if (s mod k = c_i) then
2:    if message is c-interest and s is not in CS then
3:       store s in CS
4:    else if CS match then
5:       send back data
6:    else if PIT match then
7:       add receiving interface in PIT
8:    else
9:       generate p-interest with s identifier
10:      multi-cast p-interest according to FIB
11: else
12:   if message is p-interest and j ∈ CRT_i then
13:      eliminate s in CCS
14:      insert request for s in PIT
15:      multicast p-interest according to FIB
16:   else
17:      use the regular CCN process (check CS, then PIT, then multicast interest)
```

other nearby cache that i does not store the segment s, so the CCS table of nearby caches should not keep a trace of s (lines 8 to 10).

- i is not in charge of the requested segment s ($s \mod k \neq c_i$). For regular *interest* messages, the cache i runs the regular CCN process (line 17). As for the case of a *p-interest*, the cache is aware that s is no longer in the CS of the message source. The entry should eliminated and the message forwarded.

Our cooperative caching introduces an extra RTT and some control messages between cooperating caches during the management of interests. Note however that the caches that cooperate are expected to be directly linked, so the extra-latency should stay low. We analyse the overhead by simulations in Section 5.3.

5.3 Performance Analysis

We now show the results we obtained from the implementation of our cooperative caching policy into the open-source CCNx project. Other theoretical analysis of our policy can be found in [59].

5.3.1 Platform Setup and Simulation Settings

Our modified CCNx prototype was deployed on 40 machines with a dual 2.70 GHz Pentium processor and 4 GB RAM. Each machine used a Ubuntu 10.04 system and was connected to a switch via a 100 Mbps ethernet card. Since we focus on the performances of the cooperative caching policy, the assignment of labels for each

cache is pre-computed separately. Both the determined label and k are thus the input parameters of the *ccnd* function.

The topology of the tested ISP network is the European Backbone *Ebone* [83]. Every machine in our platform is a cache. In particular, 20 out of the 40 routers act as edge routers, with the responsibility to emit requests from the equivalent of 10,000 end users, and three routers act as PoPs (virtual Internet exchange points). The cache capacity of every router is limited to 1,000 segments.

We evaluate five caching policies: the standard policies without any admission policy and LRU and Least Frequently Used (LFU) replacement policies, and our admission policy with LRU where k is equal to 2, 4, and 6. We measure (*i*) the **total caching diversity** by counting the number of distinct segments that are stored in the network. The more distinct segments are stored, the better is the cooperative caching system. The maximal caching diversity is 40,000 segments. (*ii*) the **per-video caching diversity** is the percentage of cached segments (including replicas) belonging to each video (or channel), and (*iii*) the **ISP-friendliness** of the policy by measuring the number of requests that are treated by servers outside the network. The lesser is the number of requests, which means the less inter-domain traffic is produced, the friendlier is the caching policy for the ISP. This latter measure is the main one since it reveals the hit-ratio performance of the overall network of caches.

We use the synthetic model of [65] for modeling the behavior of users of catch-up TV. We assume that 20 channels should be delivered in the network. The channel popularity follows the Zipf distribution [77]. The basic data unit of video streaming is a segment, which contains an one minute video playback. A TV stream of each channel is then divided into *programs*. The popularity of programs decreases with time. Every client is assigned a role: half of the clients are *surfers* (watch a same program during 1 or 2 segments before switching to another program), 40% of them are *viewers* (switch after a duration uniformly chosen between 2 and 60 minutes), and the rest (10%) are *leavers* (stay on a program during more than 60 minutes). The duration of our simulation is also one week, and 200,000 segments are produced during this period.

5.3.2 Performance Analysis

The caching diversity of our caching policy with an admission policy is larger than that of the traditional caching policy without admission policy. In Figure 5.2, the gain reaches 30% for $k = 6$ although the use case is not favorable due to the large number of segments that are generated over the simulation week.

We then analyze the caching policy channel per channel. We focus on the 20 most popular channels. For each channel, we compute the ratio of the number of cached segments with a policy to the number of cache segments with the traditional LRU policy. When this ratio is more (respectively less) than 1, this policy caches more (respectively less) segments than the LRU. As seen in Figure 5.3, LFU caching polices do not differ much from the LRU ones. On the contrary, our policy with admission control produces a different pattern. The number of stored segments from the first and second most popular channels decreases, while it significantly increases from the 7th to the 17th most popular videos. That is, the aforementioned higher

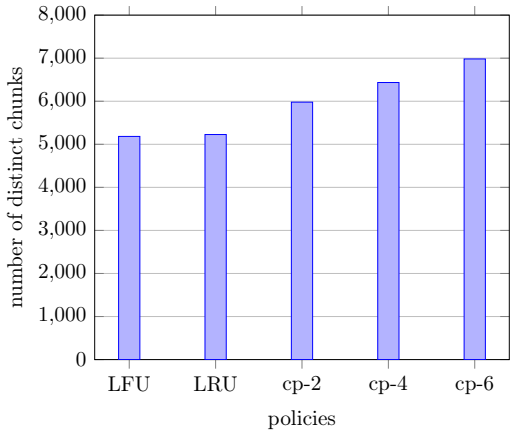


Figure 5.2 – Caching diversity for the time-shifted TV service

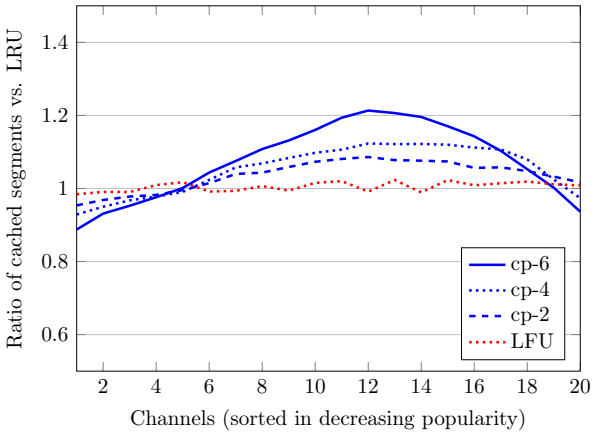


Figure 5.3 – Stored segments for the channels of the time-shifted TV service.

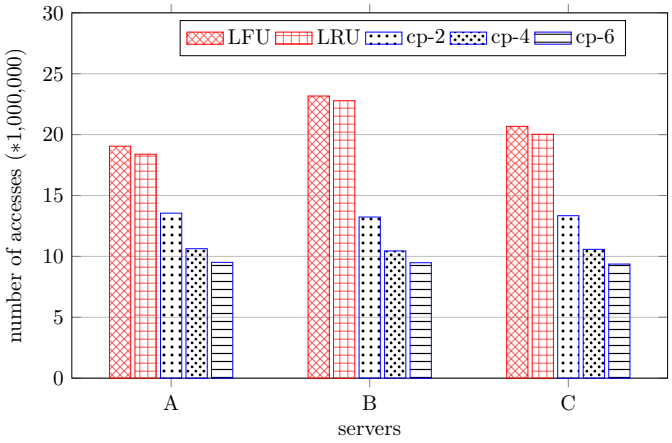


Figure 5.4 – Number of times each server is accessed for the time-shifted TV

		LRU	*CP-4*	*overhead*
Time-shifted TV	*Average response time (ms)*	351.5	383.8	+9.1%
	Total number of messages ($\times 10^9$)	4.217	4.508	+6.9%

Table 5.2 – Latency and Message Overhead

diversity applies to the segments from the "middle-popular" videos. In other words, the segments of the most popular videos monopolize the storage space in the basic LRU caching system. In comparison, our cooperative caching is smarter as it puts less emphasis on the content of these top popular videos. The highest benefit of these middle-popular films reaches more than 20% for $k = 6$.

The most important result is the ISP-friendliness of these policies, which we measure with the outgoing request traffic at the three PoP routers. We demonstrate in Figure 5.4 the ISP-friendliness of our policy with impressive gains. The gain reaches 50% when we implement our cooperative caching with $k = 6$. This result indicates an overall hit-ratio greater than the hit-ratio of the traditional caching policies.

Finally, we evaluate the extra traffic generated by the implementation of our policy within the ISP network, as well as the overall latency increase. We measure the total number of messages, and we compare the average response times for each request. We combine results in Table 5.2. Our policy causes neither a significant degradation of the latency, nor a network flood. Regarding the computation overhead, our protocol can be implemented currently at the edge of an ISP network according to the measurement of [74]. Thus, we claim that our policy is a practical protocol to improve the ISP's in-network caching performances.

5.4 Other Studies Related to System Design

What we present here is one of the multiple studies related to system design. We chose this one in particular since it is representative of the diversity of our work in this area, including sophisticated simulations based on a large number of machines, real traces, and program implementations. Our other works in the same vein include other contributions to time-shifted video streaming systems [65, 66], video delivery in telco-CDN [55, 57], channel switching in P2P-based IPTV [20, 50], and CCN-related developments about the management of routers [92, 93].

Chapter 6

Perspectives

As said in the preamble of this thesis, the art of designing multi-year research programs is a challenge for researchers in applied research areas. Multiple companies, public actors, and billions of users interplay to build a multi-form ever-changing object that is the Internet. This object of research is expected to expand by orders of magnitude in the next years, especially through the emerging field of autonomous connected objects. The so-coined Internet of Things will generate many new problems, which we are still unable to imagine. In the meantime, multimedia and interactive services are also expected to explode in a world where digital environment will be pervasive (also known as ubiquitous) and multimedia. Again, nobody can be sure of the problems that these new services will generate on the infrastructures.

In the following, despite the above doubts, we provide some research perspectives, which, in our opinion, are worth investigating for the next years and appear to be in the continuity of the research we have conducted over the past years.

6.1 Interactive Multimedia Services in the Fog

The traffic related to multimedia content, and in particular video, has exploded over the past years. This growth is expected to continue with the advent of new video formats (e.g. 4k videos) and the integration of multimedia into our daily lives (e.g. video in education). As shown in this thesis the world is switching from TV with a handful of broadcasters to OTT video services with thousands of broadcasters. The big challenge appears with the new features of multimedia services: always more *interactive* service, the *personalization* of content, and *adaptive* multimedia streaming.

As shown in Chapter 3, the *latency* of today's cloud architecture is not low enough to guarantee QoE for users of interactive services. The need for servers close to the end-users is already strong. This trend is expected to be even stronger in the next years because the devices that we will use to consume multimedia content are more integrated into our body, for example watches, glasses, and eye lens. Therefore, as explained in Section 3.1, an appealing architecture is a CDN managing servers that are very close to the end-users, in other words at the edges of the network. It is thus natural that network operators (more precisely the entities that will be in charge of operating the networked infrastructure for delivering content on behalf of service

providers) develop their ability to leverage servers as close as possible to the end-users.

In the meantime, the *personalization* of multimedia services is also a major, sustainable trend. The wide adoption of ABR streaming technologies to deliver the video content makes that CDNs have to take into account the characteristics of every end-user to prepare the content, distribute it to the edge servers, and deliver it to the end-users. In addition to this delivery trend, the business models of service providers are more often based on personalized offers. For example the advertisement model of Google is widely admitted as a brilliant technique to push advertisement while staying friendly to the users, but it requires adapting the content to every end-user. At the end, we envision that every end-user will be served by a somehow "personalized" server, which will perform computing tasks for every end-user separately. In this trend, cloud gaming is again an example of such personalized, computing-hungry, interactive service. Personalized advertisement is another customized feature that can be worth the price of reserving specific resources for every end-user.

Overall, the picture of the context for massive, interactive, multimedia delivery includes:

- A large, pervasive set of servers, with relatively poor hardware capabilities (mainly memory and computing units). This forms the *resource pool*.
- A massive population of end-users, consuming personalized multimedia services requiring heavy resource reservations.
- A number of service providers, which manage elastic, massive software through the mean of thousands of Virtual Machine (VM).
- A network operator or a CDN, which will be the main orchestrator of the resource pool.

The research works related to a so complex area is vast. We would like here to highlight some of the topics that are, in our opinion, the most promising. In short, there are two major areas: to develop new technologies, and to improve content delivery architecture.

6.1.1 To develop new technologies that enhance multimedia delivery

To improve multimedia delivery in some specific use cases, the scientific community has started the conception of new dedicated technologies. We highlight hereafter two of these technologies:

- **Multimedia Broadcast Multicast Services (eMBMS)**. Video broadcast technologies are essentially not designed to address a heterogeneous population of end-users. The same content is delivered, without regard to the characteristics of the reception devices (e.g. display size and hardware capabilities) and to the quality of the support network. Yet, the heterogeneity of the devices that are used to consume video has increased, from HDTV to smartphones. The specification of video broadcast technologies for the Long-Term Evolution (LTE) networks and the next generation (5G) of cellular networks opens new opportunities

to address the limitations of video broadcasting. Promising opportunities hold in combining broadcast delivery from the server to the end-users, and unicast feedback from each end-user to the server. Our proposal is to find a balance between video broadcasting and heterogeneous population by implementing ABR video streaming technologies into cellular broadcasting. More specifically, we merge two technologies: DASH and eMBMS. The eMBMS specification aims at broadcasting a video stream into some selected cells in a cellular network. Though eMBMS includes all the modes that are possible with standard broadcast technologies (including nation-wide TV channel and regional program), it typically targets the use case of the manager of an event who would like to offer a dedicated TV channel to any attendee of the said event (*e.g.* a tennis channel for attendees of a tennis tournament). From a technical perspective, the overall problem is twofold: (i) No feedback loop has been defined in the eMBMS specification though it is easily available in a cellular infrastructure; and (ii) No mechanism has been defined to broadcast several representations of the same video in an eMBMS channel. Our goal is to study and develop proposals for enhancing eMBMS in the 5G cellular networks. The objective is to keep the benefits of broadcast for massive distribution of the same content (one transmission for several recipients) and to combine them with dynamic adaptation thanks to a feedback channel. The support for adaptive streaming is a major new feature, which requires significant protocol revisions, as well as optimization studies.

- **Multi-Path Transmission Control Protocol (MPTCP)**. The availability of multiple network accesses becomes a commonplaces. Devices equipped with multiple network interfaces like smartphones and laptops are often shipped with built-in network adapters of different wireless technologies, which enables the connection to multiple wireless local area networks and cellular data networks. At the same time, wireless network coverage has become so widespread that mobile devices are often located in overlapping coverage areas of independent access networks. To profit from the availability of these networks accesses, network scientists have started the specification of a new protocol, based on the widely adopted Transmission Control Protocol (TCP). We will not enter into the details of MPTCP here, but we would like to emphasize that MPTCP has not considered the specific case of the transmission of video streams. In particular, we see a lot of opportunities in the development of algorithms that exploits the fact that streams consist of a series of frames with various significance. In particular, we would like to analyze the potential delivery improvements that could be achieved by implementing a new scheduler, which does not request all data according to the video timestamp but rather to the importance of the frames in the overall decoding process.

As the above highlights, the research on applied sciences like the one we are talking about with Internet multimedia is definitely technology-oriented. What is said above corresponds to the reality of today, but undoubtedly this text will be outdated very fast. Nonetheless, not only they represent attractive research areas for the next few years, but also they show that a scientist should keep strong ties with leading

innovative companies and standards groups to keep a good enough understanding of the main technological trends.

6.1.2 To better organize the architecture of content delivery

Many network management problems related to CDNs are still open. We plan to contribute to the research activities that aim to better understand and organize the content delivery in general, and the integration of CDN in the networks in particular. Here are some topics:

- **Content Placement**. The placement of content into the thousands of edge servers so that the QoE of the end-users is maximized is a significant topic, which spans multiple areas, including software management (*e.g.*, easy-to-deploy and easy-to-migrate virtualized software and hypervisor integration in connected objects) and network management (*e.g.* fast network re-configuration and mobility management with interactive, continuously demanding services). The placement of VM into data-centers is done today by libraries like OpenStack. Though, OpenStack does not currently implement sophisticated VM placement algorithms. However, the topic becomes both critical and challenging when the resource pool is not a data-centers, but what is sometimes coined as *the fog*, which consists in thousands of loosely controlled servers at the edge of the networks. Multiple constraints interfere, in particular, the fact that fog resources are shared with other applications having impredictible resource needs. Furthermore, due to the real-time nature of multimedia services, the migration of VM from one host to another is especially hard, to not say impossible in practice. The placement of VMs in the cloud has been intensively studied in the recent years. These studies however do not take into account two aspects: the tunability of the resource consumption per VM, and the impact of VM placement on the QoE. Furthermore, the described problem considers a multi-dimensional problem where multiple resources have to be shared among the VMs that run in the same host. Finally, the virtualisation of multimedia encoders has rarely been studied, in particular never in conjonction with cloud middleware such as OpenStack. A short-term objective is the design of a software library, whose main function is to optimally place VMs in the right resource pools and tweak the multimedia parameters in real time to fit the resource consumption to the actual, variable amount of available resources.

- **Economics of CDN and net neutrality**. 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; *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. 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. 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, but the role of CDNs is barely mentioned. More generally, the performance analysis community has barely considered the economics of CDN actors so far (at the notable exception of our own works [68, 69]). We plan to address 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 aim at developing 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.

6.2 Massive Stream Optimization

We have shown throughout this thesis that optimization techniques can be powerful to improve network management. They are however still not very used by operators in practical network management. The problem is often that computation time is far too large for big (even not massive) problems, while networks are dynamic environments, which require continuous adaptation. Therefore, optimization is mostly used on static cases to evaluate performances and to identify trends that can be derived in guidelines. A typical current optimization study is the one described in Section 4.2.

The term Big Data refers to the technologies that have been developed to process very large amount of data (in the order of terabytes at least) in parallel by a large amount of machines in DC (in the order of hundreds machines for a task). The Hadoop suite of software has typically been designed to provide the middleware that can be directly used in large-scale services. For example the Map-Reduce framework has become extremely popular. More recently, a new family of tools has been studied: a middleware to process streams of data in an online fashion. So far, the tool that has get the most attention is Storm, initially proposed by Twitter. The idea behind Storm is to keep a small "system state", which is constantly updated by processes running on multiple streams of data, typically tweets. Some recent papers have studied how to use Big Data technologies to make large-scale computation of optimization problems [27, 52] and also one of our papers [57]. In parallel, there is a huge literature about online optimization but, as fas as we know, no recent papers has dealt with utilizing Storm to make online parallel computation of optimization problems.

Applying such massive, parallel, online processing tasks on network management problem is an appealing research program. Again, it spans several fields, including parallel computation, optimization techniques, forecasting algorithms, software and network management.

On our side, we will pay a special attention to cellular networks. For years, cellular network operators have followed standards, which usually imply costly infrastructure deployment, long-term return on investment, and safe network behavior.

More recently, such model has been challenged, partly because short-term business optimizations are preferred by stakeholders, but also because the abrupt (and quite unexpected at this extent) increase of data traffic carried by cellular network has become a major concern. In this context, network operators see virtualization techniques as potential panacea. The popularity of the Network Function Virtualization (NFV) standardization body, as well as the actual deployment of some of the first results of this body, are a good indicator of how fast are network operators willing to embrace virtualization technologies for cellular networks.

In the meantime, technologies related to Software-Defined Networks (SDN) have received an overwhelming attention not only due to their appealing features but especially because vendors have actually started implementing and deploying switches and routers that support these technologies, especially OpenFlow. SDN technologies are excellent candidates for a revamping of the core network of cellular network. A series of recent papers [45, 49] have studied a new implementation of the Evolved Packet Core (EPC) of the fourth generation (4G) network, with extension to the next fifth generation (5G), including one of our papers [15].

The combination of all virtualized network management techniques and of massive data extraction calls for the development of smarter online optimization middleware, which can be implemented in DC. Thus, the management of cellular networks will enter in a new era, which, at least for the regular Internet traffic as we know it today, will offer better performances and interactivity.

Acronyms

ABR Adaptive Bit-Rate 6

API Application Programming Interface 16

CCN Content-Centric Network 8

CCS Cooperative Content Store 33

CDF Cumulative Density Function 19

CDN Content Delivery Network 7

CRT Cooperative Router Table 33

CS Content Store 33

DASH Dynamic Adaptive Streaming over HTTP 6

DC Data-Center 7

DNS Domain Name System 8

DVR Digital Video Recorder 6

eMBMS Multimedia Broadcast Multicast Services 40

FIB Forward Interest Table 33

HD High-Definition 6

ICN Information-Centric Network 8

ILP Integer Linear Program 21

ISP Internet Service Provider 32

LFU Least Frequently Used 36

LRU Least Recently Used 32

LTE Long-Term Evolution 40

MPTCP Multi-Path Transmission Control Protocol 41

NAT Network Address Translator 7

NFV Network Function Virtualization 44

OTT Over-The-Top 17

P2P Peer-to-Peer 7

PIT Pending Interest Table 33

QoE Quality of Experience 7

SDN Software-Defined Networks 44

TCP Transmission Control Protocol 41

UGC User-Generated Content 6

VM Virtual Machine 40

VoD Video on Demand 18

Bibliography

[1] Webrtc: Web browser with real-time communications. 6

[2] Cisco visual networking index: Forecast and methodology, 2012-2017. May, 2013. `http://goo.gl/5QHEFO`. 5

[3] Twitch.tv: 2013 retrospective, Jan. 2014. `http://twitch.tv/year/2013`. 16

[4] 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 *Proc. of IEEE INFOCOM*, 2012. 6, 8

[5] Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen, and Zhi-Li Zhang. Vivisecting youtube: An active measurement study. In *Proc. of IEEE INFOCOM*, 2012. 8

[6] Vijay Kumar Adhikari, Sourabh Jain, and Zhi-Li Zhang. Where Do You "Tube"? Uncovering YouTube Server Selection Strategy. In *Proc. of IEEE ICCCN*, 2011. 25

[7] Micah Adler, Ramesh K. Sitaraman, and Harish Venkataramani. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55(18):4007–4020, 2011. 25, 27

[8] Ravindra K. Ahuja, James B. Orlin, Clifford Stein, and Robert E. Tarjan. Improved algorithms for bipartite network flow. *SIAM J. Comput.*, 23(5):906–933, 1994. 25

[9] Jussara M. Almeida, Derek L. Eager, Mary K. Vernon, and Stephen J. Wright. Minimizing delivery cost in scalable streaming content distribution systems. *IEEE Trans. on Multimedia*, 6(2):356–365, 2004. 25

[10] Konstantin Andreev, Bruce M. Maggs, Adam Meyerson, and Ramesh K. Sitaraman. Designing overlay multicast networks for streaming. In *Proc. of ACM SPAA*, 2003. 25

[11] Ramon Aparicio-Pardo, Karine Pires, Alberto Blanc, and Gwendal Simon. Transcoding live adaptive video streams at a massive scale in the cloud. In *Proc of ACM MMSys*, 2015. 17, 19, 29

[12] Sean K. Barker and Prashant Shenoy. Empirical Evaluation of Latency-sensitive Application Performance in the Cloud. In *Proc. of ACM MMSys*, 2010. 11

[13] 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. 7

[14] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Proc. of Usenix NSDI*, 2004. 12, 13

[15] Siwar Ben Hadj Said, Malla Reddy Sama, Karine Guillouard, Gwendal Simon, Jean-Marie Bonnin, Xavier Lagrange, and Lucian Suciu. New Control Plane in 3GPP LTE/EPC Architecture for On-Demand Connectivity Service. In *Proc. of IEEE CLOUDNET*, 2013. 44

[16] 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. 18

[17] Gloria Chatzopoulou, Cheng Sheng, and Michalis Faloutsos. A first step towards understanding popularity in youtube. In *Proc. of IEEE INFOCOM Workshops*, 2010. 18

[18] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, , and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *Proc. of ACM Multimedia*, 2011. 11

[19] Yiping Chen, Jimmy Leblet, and Gwendal Simon. On reducing the cross-domain traffic of box-powered cdn. In *Proc. of IEEE ICCCN*, 2009. 20

[20] Yiping Chen, Erwan Le Merrer, Zhe Li, Yaning Liu, and Gwendal Simon. OAZE: A network-friendly distributed zapping system for peer-to-peer IPTV. *Computer Networks*, 56(1):365–377, 2012. 38

[21] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *Proc. of ACM NetGames*, 2012. 12, 16

[22] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Systems Journal*, 20(2), March 2014. 12, 16

[23] Cisco. Visual Networking Index: 2011-2016. Technical report, Cisco Inc., May 2012. 8, 25

[24] Mark Claypool and Kajal Claypool. Latency can kill: precision and deadline in online games. In *Proc. of ACM MMSys*, 2010. 11

[25] Mark Claypool and Kajal T. Claypool. Latency and player actions in online games. *Communications of The ACM*, 49:40–45, 2006. 11

[26] Dan Cryan. eSports video: A cross platform growth story. Technical report, IHS Tech., June 2014. http://is.gd/NHVdfi. 16

[27] Thibault Debatty, Pietro Michiardi, Wim Mees, and Olivier Thonnard. Determining the k in k-means with mapreduce. In *Proc. of EDBT/ICDT*, 2014. 43

[28] Mikael Desertot, Clement Escoffier, and Didier Donsez. Towards an autonomic approach for edge computing: Research articles. *Concurr. Comput. : Pract. Exper.*, 19(14):1901–1916, September 2007. 8

[29] Marcel Dischinger, Andreas Haeberlen, P. Krishna Gummadi, and Stefan Saroiu. Characterizing residential broadband networks. In *Proc. of ACM IMC*, 2007. 12

[30] Drew Fitzgerald and Daisuke Wakabayashi. Apple Quietly Builds New Networks. Wall Street Journal, Feb. 2014. http://is.gd/MXc2b7. 16

[31] Juan Pedro Munoz Gea, Ramon Aparicio, Houssein Wehbe, Loutfi Nuaymi, and Gwendal Simon. Optimization framework for uplink video transmission in hetnets. In *Proc. of ACM MoVid*, 2014. 29

[32] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988. 25

[33] Fabrice Guillemin, Bruno Kauffmann, Stephanie Moteau, and Alain Simonian. Experimental analysis of caching efficiency for youtube traffic in an isp network. In *Proc. of IEEE ITC*, 2013. 18

[34] Bing Han, Jimmy Leblet, and Gwendal Simon. Hard multidimensional multiple choice knapsack problems, an empirical study. *Computers & operations research*, 37(1):172 – 181, january 2009. 29

[35] Bing Han, Jimmy Leblet, and Gwendal Simon. Query range problem in wireless sensor networks. *IEEE communications letters*, 13(1):55 – 57, january 2009.

[36] Bing Han and Gwendal Simon. Fair capacity sharing among multiple sinks in wireless sensor networks. In *Proc. of IEEE MASS*, 2007. 29

[37] Todd Hoff. Gone fishin': Justin.tv's live video broadcasting architecture. High Scalability blog, Nov. 2012. http://is.gd/5ocNz2. 8

[38] Cisco Inc. Cisco visual networking index: Forecast and methodology, 2010 âĂŞ 2015. June, 2011. http://goo.gl/G0jcGS. 5

[39] Sandvine Inc. Global internet phenomena report, fall 2011. 2011. http://www.sandvine.com/news/global_broadband_trends.asp. 5

[40] Mathew Ingram. You think the internet is big now? akamai needs to grow 100-fold. Om Malik, Jun 2012. http://is.gd/3vTZPC. 25

[41] Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs, and Rebecca Braynard. Networking named content. In *Proc. of ACM CoNEXT*, 2009. 8, 31

[42] Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hoßfeld. An evaluation of qoe in cloud gaming based on subjective tests. In *Proc. of IMIS*, 2011. 11

[43] Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hoßfeld. Gaming in the clouds: Qoe and the users' perspective. *Mathematical and Computer Modelling*, Dec. 2011. 11

[44] 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. 7

[45] Xin Jin, Li Erran Li, Laurent Vanbever, and Jennifer Rexford. Softcell: scalable and flexible cellular core network architecture. In *ACM Conext*, 2013. 44

[46] 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 *Proc. of ACM WWW*, 2012. 6

[47] Joaquín Keller and Gwendal Simon. Toward a peer-to-peer shared virtual reality. In *Proc. of RESH*, 2002. 31

[48] Joaquín Keller and Gwendal Simon. Solipsis: A massively multi-participant virtual world. In *Proc. of PDPTA*, 2003. 31

[49] James Kempf, Bengt Johansson, Sten Pettersson, Harald Luning, and Tord Nilsson. Moving the mobile evolved packet core to the cloud. In *Proc. of IEEE WiMob*, 2012. 44

[50] Anne-Marie Kermarrec, Erwan Le Merrer, Yaning Liu, and Gwendal Simon. Surfing peer-to-peer iptv system: Distributed channel switching. In *Proc. of Euro-Par*, 2009. 8, 38

[51] Joohwan Kim and R. Srikant. Achieving the maximum p2p streaming rate using a small number of trees. In *Proc. of IEEE ICCCN*, 2011. 25

[52] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *Proc. of ACM SPAA*, 2011. 43

[53] Jimmy Leblet, Zhe Li, Gwendal Simon, and Di Yuan. Optimal network locality in distributed virtualized data-centers. *Computer Communications*, 34(16):1968–1979, 2011. 8

[54] Avraham Leff and James T. Rayfield. Alternative edge-server architectures for enterprise javabeans applications. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, Proc. of ACM Middleware '04, 2004. 8

[55] Zhe Li, Mohamed Karim Sbai, Yassine Hadadj-Aoul, Anine Gravey, Damien Alliez, Jeremie Garnier, Gwendal Simon, and Kamal Singh. Network friendly video distribution. In *Proc. of IEEE NoF Conf.*, 2012. 38

[56] Zhe Li and Gwendal Simon. Time-Shited TV in Content Centric Networks: the Case for Cooperative In-Network Caching. In *Proc. of IEEE ICC*, 2011. 31, 32

[57] Zhe Li and Gwendal Simon. In a telco-CDN, pushing content makes sense. *IEEE Transactions on Network and Service Management*, 10(3):300–311, 2013. 38, 43

[58] Zhe Li and Gwendal Simon. Cooperative caching in a content centric network for video stream delivery. *Journal of Network and Systems Management*, pages 1–29, 2014. 31, 32

[59] Zhe Li, Gwendal Simon, and Annie Gravey. Caching policies for in-network caching. In *Proc. of IEEE ICCCN*, 2012. 35

[60] Chao Liang, Miao Zhao, and Yong Liu. Optimal bandwidth sharing in multi-swarm multiparty p2p video-conferencing systems. *IEEE/ACM Trans. Netw.*, 19(6):1704–1716, 2011. 22

[61] Jiayi Liu, Shakeel Ahmad, Eliya Buyukkaya, Raouf Hamzaoui, and Gwendal Simon. Resource allocation in underprovisioned multioverlay live video sharing services. In *Proc. of ACM CoNEXT Workshop*, 2012. 8, 22, 25

[62] Jiayi Liu, Eliya Buyukkaya, Shakeel Ahmad, Raouf Hamzaoui, and Gwendal Simon. Resource allocation in underprovisioned peer-to-peer live video sharing services. *Peer-to-Peer Networks and Applications*, 2014. 22, 25

[63] Jiayi Liu and Gwendal Simon. Fast Near-Optimal Algorithm for Delivering Multiple Live Video Channels in CDNs. In *Proc. of IEEE ICCCN*, 2013. 26, 28

[64] 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. 26

[65] Yaning Liu and Gwendal Simon. Distributed Delivery System for Time-Shifted Streaming System. In *Proc. of IEEE LCN*, 2010. 8, 20, 36, 38

[66] Yaning Liu and Gwendal Simon. Peer-assisted time-shifted streaming systems: Design and promises. In *Proc. of IEEE ICC*, 2011. 8, 20, 38

[67] Zhengye Liu, Yanming Shen, Keith W. Ross, Shivendra S. Panwar, and Yao Wang. Layerp2p: Using layered video chunks in p2p live streaming. *IEEE Transactions on Multimedia*, 11(7):1340–1352, 2009. 22

[68] Patrick Maillé, Karine Pires, Gwendal Simon, and Bruno Tuffin. How Neutral is a CDN: An Economic Approach. In *Proc. of IEEE/IFIP CNSM*, 2014. 17, 43

[69] Patrick Maillé, Gwendal Simon, and Bruno Tuffin. Toward a Net Neutrality Debate that Conforms to the 2010s. March 2015. 43

[70] Usama Mir, Houssein Wehbe, Loutfi Nuaymi, Aurelie Moriceau, and Bruno Stevant. The zewall project: Real-time delivering of events via portable devices. In *Proc. of IEEE VTC-Spring*, 2013. 6

[71] Di Niu and Baochun Li. Asymptotic optimality of randomized peer-to-peer broadcast with network coding. In *Proc. of IEEE INFOCOM*, 2011. 25

[72] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The Akamai network: a platform for high-performance internet applications. *Op. Sys. Rev.*, 44(3):2–19, 2010. 25

[73] Andrea Passarella. A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Computer Communications*, 35(1):1–32, 2012. 7

[74] Diego Perino and Matteo Varvello. A reality check for content centric networking. In *Proc. of ACM ICN*, 2011. 38

[75] Karine Pires and Gwendal Simon. DASH in Twitch: Adaptive Bitrate Streaming in Live Game Streaming Platforms. In *Proc. of ACM CoNEXT VideoNext Workshop*, 2014. 17, 19

[76] Karine Pires and Gwendal Simon. Youtube live and twitch: A tour of user-generated live streaming systems. In *Proc. of ACM MMSys (dataset track)*, 2015. 17, 21

[77] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Qi Zhao, and Jun Xu. Modeling channel popularity dynamics in a large iptv system. In *Proc. of ACM SIGMETRICS/Performance*, 2009. 36

[78] Md. Golam Rabbani, Rafael Pereira Esteves, Maxim Podlesny, Gwendal Simon, Lisandro Zambenedetti Granville, and Raouf Boutaba. On tackling virtual data center embedding problem. In *Proc. of IFIP/IEEE IM*, 2013. 29

[79] Jan Seedorf, Sebastian Kiesel, and Martin Stiemerling. Traffic localization for p2p-applications: The alto approach. In *Proc. of IEEE P2P*, 2009. 7

[80] Sudipta Sengupta, Shao Liu, Minghua Chen, Mung Chiang, Jin Li, and Philip A. Chou. Peer-to-peer streaming capacity. *IEEE Trans. on Information Theory*, 57(8):5072–5087, 2011. 25

[81] Gwendal Simon, Yiping Chen, Ali Boudani, and Gilles Straub. Internet-scale simulations of a peer selection algorithm . In *Proc. of MSOP2P Workshop*, 2008. 20

[82] Iraj Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. *MultiMedia, IEEE*, 18(4):62–67, 2011. 6

[83] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. In *Proc. of ACM SIGCOMM*, 2002. 36

[84] Thomas Stockhammer. Dynamic adaptive streaming over HTTP: standards and design principles. In *Proc. of ACM MMSys*, 2011. 6

[85] Srikanth Sundaresan, Walter de Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Broadband internet performance: a view from the gateway. In *Proc. of ACM Sigcomm*, 2011. 12, 13

[86] Laura Toni, Ramon Aparicio-Pardo, Karine Pires, Gwendal Simon, Alberto Blanc, and Pascal Frossard. Optimal selection of adaptive streaming representations. *ACM Trans. Multimedia Comput. Commun. Appl.*, 11(2s):43:1–43:26, February 2015. 29

[87] Laura Toni, Ramon Aparicio-Pardo, Gwendal Simon, Alberto Blanc, and Pascal Frossard. Optimal set of video representations in adaptive streaming. In *Proc. of ACM MMSys*, 2014. 29

[88] Ruben Torres, Alessandro Finamore, Jin Ryong Kim, Marco Mellia, Maurizio M. Munafo, and Sanjay Rao. Dissecting Video Server Selection Strategies in the YouTube CDN. In *Proc. of IEEE ICDCS*, 2011. 25

[89] Miao Wang, Lisong Xu, and Byrav Ramamurthy. A flexible divide-and-conquer protocol for multi-view peer-to-peer live streaming. In *IEEE P2P*, 2009. 21, 22

[90] Miao Wang, Lisong Xu, and Byrav Ramamurthy. Improving multi-view peer-to-peer live streaming systems with the divide-and-conquer strategy. *Computer Networks*, 55(18):4069–4085, 2011. 21, 22

[91] 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 *Proc. of ACM Multimedia*, 2009. 8

[92] Wei You, Bertrand Mathieu, and Gwendal Simon. Exploiting end-users caching capacities to improve content-centric networking delivery. In *Proc. of 3PGCIC*, 2013. 38

[93] Wei You, Bertrand Mathieu, Patrick Truong, Jean-François Peltier, and Gwendal Simon. Dipit: A distributed bloom-filter based PIT table for CCN nodes. In *Proc. of IEEE ICCCN*, 2012. 38

[94] Mohamed Faten Zhani, Qi Zhang, Gwendal Simona, and Raouf Boutaba. Vdc planner: Dynamic migration-aware virtual data center embedding for clouds. In *Proc. of IEEE/IFIP IM*, 2013. 29

[95] Can Zhao, Xiaojun Lin, and Chuan Wu. The streaming capacity of sparsely-connected P2P systems with distributed control. In *IEEE INFOCOM*, 2011. 25

[96] Fen Zhou, Shakeel Ahmad, Eliya Buyukkaya, Gwendal Simon, and Raouf Hamzaoui. Minimizing Server Throughput for Low-Delay Live Streaming in Content Delivery Networks. In *Proc. of ACM NOSSDAV*, 2012. 27

[97] Fen Zhou, Jiayi Liu, Gwendal Simon, and Raouf Boutaba. Joint optimization for the delivery of multiple video channels in telco-cdn. In *Proc. of CNSM*, 2013. 29