Designing Adaptive Replication Schemes for Efficient Content Delivery in Edge Networks

Guthemberg Silvestre

Université Pierre et Marie Curie
Laboratoire d'Informatique de Paris 6

THÈSE

en vue d'obtenir le grade de

Docteur de l'Université Pierre et Marie Curie, Sorbonne Universités

Spécialité: Informatique

au titre de l'École Doctorale Informatique, Télécommunication et Électronique (ED 130)

présentée et soutenue publiquement le 18 octobre 2013 par

Guthemberg da Silva Silvestre

Sujet de la thèse :

Designing Adaptive Replication Schemes for Efficient Content Delivery in Edge Networks

Directeur de thèse:	M.	Pierre Sens	
Encadrants de thèse:	M.	Sébastien Monnet	
	Mme.	Ruby Krishnaswamy	
Après l'avis de:	M.	Gaétan Hains	
	M.	Gabriel Antoniu	
Devant la commission d'examen formée de:			
	M.	Gaétan Hains	Membre et Rapporteur
	M.	Gabriel Antoniu	Membre et Rapporteur
	Mme.	Sara Bouchenak	Membre
	Mme.	Maria Potop-Butucaru	Membre
	Mme.	Ruby Krishnaswamy	Membre
	M.	Sébastien Monnet	Membre
	M.	Pierre Sens	Membre

Designing Adaptive Replication Schemes for Efficient Content Delivery in Edge Networks

Abstract

Content availability has become increasingly important for the Internet delivery chain. In order to enhance availability, content distribution network (CDN) providers have invested in hybrid designs that combines resources from datacenter and edge networks. Although, to deliver videos with an outstanding availability and meet the increasing user expectations, hybrid CDNs must enforce strict QoS metrics, like bitrate and latency, through SLA contracts. Adaptive content replication has been seen as a promising way to achieve this goal. However, it remains unclear how to avoid waste of resources when strict SLA contracts must be enforced.

In this dissertation, we focus on studying and evaluating adaptive replication schemes for a new generation of hybrid networks, whose resources come from consumers' devices, such as set-top boxes. To this end, we propose (i) Caju, a general-purpose content distribution system, which handles resource allocation in edge networks, and (ii) three novel adaptive replication schemes, AREN, Hermes, and WiseReplica. Extensive simulations with Caju show that our adaptive replication schemes are very efficient and can easily be extended to other CDN architectures. Finally, we provide some guidelines about our ongoing development of Caju in a world-wide testbed deployment.

Designing Adaptive Replication Schemes for Efficient Content Delivery in Edge Networks

Résumé

La disponibilité des contenus partagés en ligne devient un élément essentiel pour toute la chaîne de distribution de vidéos. Pour fournir des contenus aux utilisateurs avec une excellente disponibilité et répondre à leurs exigences toujours croissantes, les opérateurs de *content delivery networks* (CDNs) doivent assurer une haute qualité de services, définie par des métriques comme le taux de transfert ou la latence inclus dans les contrats de *Service Level Agreement* (SLA). La réplication adaptative se présente comme un mécanisme de stockage très prometteur pour atteindre cet objectif. Par contre, une question importante reste encore ouverte: comment assurer la mise en place de ces SLAs, tout en évitant le gaspillage de ressources?

Le sujet de la thèse porte précisément sur l'étude et l'évaluation de systèmes de réplication de données pour la nouvelle génération de CDNs hybrides, dont une partie des ressources de réseaux et de stockage proviennent de l'équipement des utilisateurs. Pour cela, nous proposons (i) une architecture de gestion de ressources des utilisateurs nommée Caju, et (ii) trois nouveaux systèmes de réplication adaptatifs, AREN, Hermes, et WiseReplica. Des simulations précises avec Caju montrent que nos systèmes de réplication adaptatifs sont très performants et peuvent être facilement étendus à d'autres types d'architecture. Comme perspectives, nous comptons réaliser le développement et l'évaluation d'un prototype *proof-of-concept* sur PlanetLab.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I am deeply indebted to my advisers, Pierre Sens and Sébastien Monnet, for their reliable and helpful guidance in my thesis. For the past three years they have supported and encouraged me in seeking both questions and answers. They have strengthened my confidence and enthusiasm for carrying out research. I will always remember their generous and quite patient support that helped me to overcome the hardships and pitfalls during my studies. I would like to thank Ruby Krishnaswamy for having proposed the fascinating subject of this thesis, and Orange Labs for having provided properly support to conduct my research.

I am thankful and profoundly delighted that Gaétan Hains and Gabriel Antoniu have accepted to examine and report on the content of this dissertation. Their voluntary commitment to not only read and understand this thesis, but also to make part of thesis committee, is remarkable, particularly as their invaluable efforts must be done in parallel to their own research. Similarly, I am grateful to Sara Bouchenak and Maria Potop-Butucaru for having helpfully agreed to make part of my thesis committee.

I have been fortunate to have met a smart set of colleagues at UPMC. I would like to thank Véronique Simon and Sergey Legtchenko for their helpful support and advices at the early stage of my research. I am so grateful to David Buffoni for his precious advices, tireless motivation, and patient collaboration on our joint work of the Chapter 6 of this dissertation. I would like to thank Karine Pires for our fruitful discussions and her important contributions to Chapter 5 and 6. I would also like to recognize the work done by Gaétan Hains and Marcelo Amorim during my midterm thesis evaluation. Their comments and advices were particularly useful to the accomplishment of my thesis.

I leave UPMC with a precious set of friends. I will not forget the amazing and bright people that I am quite pleased to have met in the NPA team, Marguerite Sos, Thomas Bourgeau, Marco Bicudo, Sophia MacKeith, Frédéric Vaissade, Timur Friedman, Jordan Augé, Mohamed Diallo, Serge Fdida, Panayotis Antoniadis, Fábio Leão, and Jonggun Lee, to name a few. I cannot imagine what my postgraduate studies would have been like without Swan Dubois, Anissa Lamani, Luciana Arantes, Thomas Preud'homme, Ruijing Hu, and Jonathan Lejeune.

I am especially grateful to my parents, Antônio and Irene Silvestre, and to my brother, Diêgo Silvestre, for their continual encouragement and their real-life examples of willpower. I am very appreciative of all warm and affectionate support my dear in-laws have given me for pursuing my goals. Finally, I owe an immeasurable debt to my family, my dear wife Laureen and our beloved son Túlio, whose unwavering love and support have made me a better person.

Chapter 1

Introduction

Contents

1.1 Pushing Content Delivery to the Edge

Multimedia content delivery has changed dramatically in the recent years. Content delivery networks (CDNs) have allowed operators to provide content to the masses. Nowadays, content providers are able to reach worldwide audiences with reduced cost thanks to web platforms deployed on top of CDNs.

In order to deliver content efficiently, CDNs must provide mechanisms, and schemes, such as data replication and caching, that are able to track content popularity growth properly, cover different geographical locations, and provide system scalability.

The vast majority of existing CDN architectures though rely on big, remote and centralized sites, close to the core networks. Despite being definitively scalable architectures for content delivery, datacenters remain huge distributed systems that are very expensive to build and operate. Moreover, the resource allocation efficiency of such architectures depends mainly on over-provisioning.

An alternative to over-provisioning is to include edge network to CDN architectures. Resource allocation at the edge of the networks presents several advantages over traditional CDN deployments, such as the lowest ever latency, and fined grained bandwidth allocation. It might

also be seen as eco-friendly, because it allows us to reduce the energy cost of data transmission, and it might dramatically decrease the path length between the content source and destination.

Bandwidth and storage capacities available on edge networks have consistently increased in the recent years. Since the start of 2011, Free, a French Internet Service Provider, has offered to their subscribers Internet connection speed up to 100Mbps, and storage capacities at home in the order of 250GB. The ready availability of these resources have contributed to create and popularize Internet service offers, such as video on demand, high quality live streaming, backup and high speed storage synchronization.

Therefore, we refer to *hybrid CDNs* as content delivery infrastructures that combine datacenters and edge network resources, e.g. Akamai new offers [1] and Velocix [2].

1.2 Using Replication Schemes and Quality of Service Guarantees for Enhancing Availability of Internet Content in Edge Networks

There exists an increasing need for more research in easy-to-deploy, self-adapting techniques for ensuring tough Quality of Service (QoS) guarantees brought by the cloud paradigm. One of the most important new opportunities for ISPs is to offer highly available content at the edge of the network through hybrid CDNs. But to achieve this goal, they must provide mechanisms for enforcing bitrate as QoS metric of Service Level Agreement (SLA) contracts.

However, efficient resource allocation on hybrid CDNs to meet user expectations imposes new challenges. We identify *adaptive content replication* as one of such challenges. Adaptive replication plays an important role on the content availability of distributed systems, contributing directly to both storage and bandwidth provision. As the demand for content varies, the number of replicas, or nodes serving that content, must be adapted accordingly. Generally speaking, the faster and more precise the replication scheme reacts to changes on content demand, the better is the resource allocation.

1.3 Dealing with Resource Allocation for Delivering Internet Videos Efficiently

The increasing consumption of Internet videos has made fundamental changes in the Internet traffic and consumers' behaviour. Cisco System, Inc[3] forecasts that the video traffic will reach 86% of the global consumer traffic by 2016, including video on-demand (VoD), live streaming, and

[1] Akamai acquires Red Swoosh. http://www.akamai.com/html/about/ press/releases/ 2007/press_041207.html, April 2007.

[2] Enabling digital media content delivery: Emerging opportunities for network service providers. www.velocix.com, 2010.

[3] Cisco Visual Networking Index: Forecast and Methodology, 2011-2016. www.cisco.com, 2012.

Peer-to-Peer (P2P) file sharing. In fact, as the Internet access has become ubiquitous, continuously faster, and cheaper, streaming video has become mainstream. Users are progressively moving from the old-fashioned scheduled television to VoD services. This contributes to increase the expectations of consumers on Internet video delivery.

Since broadcasting future seems to be online, customers have become more sensitive to VoD quality, expecting ever-higher bitrates and lower rebuffering. Contrary to many traditional workloads, e.g. social network messaging or search engines, specifying just latency as QoS metric does not suffice. Instead, streaming traffic requires proper average bitrate to avoid rebuffering and to improve user experience. For example, Dobrain *et al.*[44] found that a 1% increase in buffering ratio can reduce the consumer's expected viewing time by more than three minutes. This suggests that SLA contracts must include bitrate as a key QoS metric.

Yet current CDN architectures are not ready to fulfil the requirements of the increasing demand for streaming and meet consumers' expectations. Through fine-grained client-side measurements from over 200 million client viewing sessions, Liu *et al.*[81] showed that 20% of these sessions experience a re-buffering ratio of at least 10%, 14% of users have to wait more than 10 seconds for video to start up, more than 28% of sessions have an average bitrate less than 500Kbps, and 10% of users fail to see any video at all.

To deal with these issues, CDN providers have started using hybrid CDNs to delivery Internet videos. This includes peer-assisted VoD systems. The aim is to take advantage of both infrastructure-based resources and P2P communication facilities. Huang *et al.* [66] suggest peer-assisted VoD systems improve resource allocation for Internet video delivery. They argue that devices on edge networks, e.g. set-top-boxes, contribute with storage and bandwidth to video delivery, reducing considerably the burden on infrastructure-based servers, and cutting operations costs. Many recent studies [97, 69, 24] confirm that exploring peer-assisted VoD system permits enhancing resource allocation for streaming videos, but none has properly evaluated the performance of video delivery regarding SLA enforcement.

In this thesis, we propose to enhance Internet video availability in edge networks using easy-to-deploy, adaptive replication schemes that enforce SLA contracts properly.

1.4 Highlighting Problems and Setting Goals

Most of today's CDNs rely on over-provisioning to offer a scalable infrastructure to Internet services. This approach leads to two main problems: (i) a waste of computational resources and (ii) an extra infrastructure deployment and maintenance cost. Yet, these architectures are not ready to fulfil the requirements of the increasing demand for emerging could services and meet consumers' expectations. As high-speed Internet connection is widespread, content availability in CDN becomes one of the main performance goals. Customers have become more sensitive to content delivery quality, expecting ever-higher bitrates and continuously lower latencies. In this context, average bitrate is a key QoS metric. CDNs and content providers must be committed to enforcing average bitrate through SLA contracts.

We claim that adaptive replication scheme in edge network resources can improve content availability, enhance resource utilization in ISPs, and meet customers' expectations properly. However, this still offers a major challenge for CDNs, particularly as they aim to avoid waste of resources. In addition, the existing CDN's mechanisms are agnostic to edge networks load, and their infrastructures are not able to provide highly available content.

Therefore, we set the following goals:

- **Designing and evaluating a hybrid content delivery system for edge networks**. The design must allow ISPs to (i) manage their infrastructure to offer outstanding content delivery for end customers, and (ii) provide interoperability with main CDN infrastructures. The novel hybrid CDN must provide techniques for content delivery and enforcing strict SLA contracts, including caching and bandwidth reservation.
- **Proposing an adaptive replication scheme for edge networks that enhances content availability**. This is at the core of this thesis's problem. Assuming that content availability is defined through bitrate as the QoS metric of SLA contracts, we are interested in adapting the number of contents' replicas according to the demand.
- **Reducing edge resources utilization while preventing SLA violations**. The replication scheme must self-adapt to changes in content demand with two specific performance goals. First, as the primary service of edge devices is Internet connection, not content delivery, we must reduce as much as possible network and storage utilization. Second, it must improve customers' satisfaction by preventing as many SLA violations as possible.
- **Providing easy-to-deploy, adaptive replication schemes**. The replication scheme must be easy-to-deploy and rely on legacy techniques available on commodity servers. Despite focusing on edge network resources, the adaptive replication scheme must be seen an additional building block for content delivery systems, flexible enough for permitting interoperability with other existing mechanisms, including caching, and other delivery protocols.
- **Coping with Internet videos delivery efficiently**. Since Internet video has become the most popular online service, our contributions must cover efficient content delivery for video services. For a matter of simplicity, we focus on the efficient delivery of popular video-on-demand services, like YouTube.

1.5 Contributions

Our contributions build an understanding of the effect of replication schemes upon the availability of Internet content, and develop techniques to adapt resource allocation in CDNs, taking advantage of edge networks. We list our main contributions as follows:

1. **Handling edge network resources for efficient content delivery**. We describe the design, model, and implementation of Caju, a hybrid content delivery system for edge networks, that allow us to handle network and storage resources from edge devices, e.g. set-top

boxes, and offer interoperability with existing CDN infrastructures based on datacenters. Our design provides mechanisms to cope with strict SLA enforcement through adaptive replication schemes. Using a collaborative caching, content replicas are either pre-fetched or removed randomly. Caju operates adaptive replication over sets of devices located close by in edge networks, namely *storage domains*. We published Caju in [121] last year.

2. **Designing and Evaluating AREN, an Adaptive Replication scheme for Edge Networks**. We present the design and evaluation of AREN, a novel replication scheme, that provides high-quality content delivery for Internet content. Based on network resources' reservation, AREN prevents most of SLA violations. It also improves the content delivery system performance, by increasing the aggregate bandwidth and reducing storage usage for replication. The functioning of adaptive replication per storage domain is straightforward. Gradually, it verifies the bandwidth reservation of a content whenever a new local request arrives, and adapts the replication degree accordingly. Using simulations on top of PeerSim [88], we evaluated the number of strict SLA violations, storage, and bandwidth usage for AREN, and we compared our results to common replication schemes. We show that AREN prevents the vast majority of SLA violations under heavily load situations. It also reduces by nearly seven-fold the required storage usage for replication through caching, and it increases by roughly 20% the aggregate bandwidth. AREN was published in [120] last year. For simulating bandwidth reservation for AREN, we designed and published a PeerSim component for bandwidth scheduler, which was published in [119] this year. A detailed description of this module is available in Appendix A.

3. **Predicting the demand for Internet contents to adapt their replication degree properly**. AREN provided promising results in adapting replication according to SLA contracts. An important technical drawback of this approach is that its functioning depends on network resources' reservation, which may discourage Internet providers from adopting it. We coped with this issue by making predictions using statistical learning methods. Therefore, our approach becomes more flexible, and it can learn system behaviour from different information sources and big amounts of data, providing a robust framework for controlling resource allocation in edge devices. We propose two learning models, from which we design two replication schemes:

    - Hermes: This is a general-purpose, adaptive replication scheme for edge networks based on accurate predictions of Internet content popularity. It (i) classifies Internet content into popularity groups, and then (ii) replicates them according. Hermes enhances content availability by enforcing strict SLA contracts properly quite similar as AREN does. But unlike most of the recent deadline-aware approaches as that used in AREN, it does not require any modification of network stack to enforce strict QoS metrics.
    - WiseReplica: We specialized our predictive approach for adapting replication of Internet videos. For that, we designed and evaluated WiseReplica, an easy-to-deploy,

SLA-based replication scheme that meets users' expectations for VoD services. Unlike Hermes, WiseReplica uses a novel machine-learned ranking that predicts the *hotness* of a video accurately, where *hotness* represents both content popularity and system resources usage. Thus, the higher the rank position, the higher the demand for fresh replicas. This intuitive model allows us to decouple streaming demand from replication policy. Hence one can evaluate different replication policies regardless of the workload. WiseReplica is fully compliant with peer-assisted VoD systems, and is flexible enough to offer interoperability with de facto approaches, including HTTP adaptive streaming technique and BitTorrent protocol[128]. We show through simulations using YouTube traces that WiseReplica outperforms a non-collaborative caching by preventing violations, reducing storage usage, and enhancing network provision.

1.6 Dissertation Plan

This dissertation proceeds as follows:

Chapter 2 We start the state-of-the-art part by studying in Chapter 2 the replication schemes. We describe their main roles in content maintenance and the most common techniques, including caching. We classify replication schemes according to the dynamics of their resource allocation. Then we show the most relevant aspects of replication schemes in P2P networks.

Chapter 3 In Chapter 3, we describe Content Delivery Networks (CDNs), focusing on the description of the most common techniques, components and building blocks. Based on the location of servers, we classify CDNs into core and edge architectures. We briefly describe the emerging hybrid designs, which relation they have with edge devices and P2P networks, and we present some upcoming challenges faced by CDNs today.

Chapter 4 We conclude the state-of-the-art part by studying the Internet content availability in Chapter 4. We selected works where content availability is defined and enforced through Service Level Agreements (SLA). As Internet video has become the most popular type of the content on the Internet, we describe how to measure the availability of Internet videos, particularly video-on-demand services like YouTube. We study which role P2P networks may play in improving content availability.

Chapter 5 We introduce the second part of our dissertation by describing in Chapter 5 our work on designing and evaluating adaptive replication schemes in edge networks. First, we revise our problem statement in delivering Internet content in edge devices properly. Then, we describe the design of Caju, that provides an architecture for organizing and using edge resources. Finally, we present one of the main contribution of this work by detailing AREN, that stands for Adaptive Replication scheme for Edge Networks. Our evaluation shows the efficiency of AREN in allocating resources according to SLA contracts.

Chapter 6 We describe in Chapter 6, the final chapter of the second part, how we adapt content replication in edge resources using predictions of resource allocation and customer demand. This provides a step forward in terms of system dependability. In this chapter, we show that accurate predictions allowed us to overcome important limitations of AREN, notably by providing a flexible, self-adaptive mechanism of resource allocation that does not rely on bandwidth reservation.

Chapter 7 We conclude this thesis in Chapter 7, highlighting the most relevant parts of this work and showing perspectives on research on remaining open problems.

Part I

State of the Art

Chapter 2

Replication Schemes

Contents

2.1 Introduction

The main purpose of replication schemes is to create and maintain copies of objects in order to enhance performance and fault tolerance of computer systems. In this chapter, we focus on replication schemes techniques for distributing Internet data in content delivery networks (CDNs), such as videos, photos, and web pages.

For CDNs, replication schemes provide two very important content properties: durability and availability. In this work, content durability represents the capacity of preventing data loss, including system failures and data corruption. For instance, storage system must replicate and place a content in at least N different devices in order to prevent simultaneous crashes and recovery

copies as quick as possible. We assume that data durability is a primary goal of any replication scheme. Fortunately, there exists an extensive coverage in the state-of-the-art works. In our thesis, we consider a couple of simple mechanisms and best-practice policies that provides data durability for storage systems in datacenter. Content availability though measures how easy a content can be fetched by a customer in terms of geographical locations and transfer rate or bitrate.

Towards these two content properties, web caching is a widespread mechanism. It is simple and efficient for delivering content in distributed storage systems. Replication schemes also adopt different strategies to improve content delivery with regard to specific performance goals. In addition, P2P systems provide attracting features for content delivery in the Internet. These, and other related issues will be also addressed in this chapter.

We consider that data consistency is outside the scope of our work, although it remains one of the most relevant issues in distributed systems. Our research focuses rather on content availability, therefore, for matter of simplicity, we assume that our target content delivery system deal with immutable, read-only content.

We organized this chapter as follows. We describe the two content properties provided by replication schemes in Section 2.2. In Section 2.3, we highlight how caching mechanisms can be used in replication schemes. We provide in Section 2.4 an intuitive classification of replication schemes according to their resource allocation strategies. We briefly discuss the role of P2P networks in replication schemes in Section 2.5, and we summarize the most important points of this chapter in Section 2.6.

2.2 Providing Storage Properties in Content Delivery Networks

In this section, we explain durability and availability as content properties provided by replication schemes in content delivery networks (CDNs). Similar to Chun *et al.* [29] definition, we assume that content durability is mostly related to the capacity of preventing data loss, while availability measures the easiness of fetching a content by a customer. Unlike general-purpose distributed systems, we consider that CDNs put great emphasis on content availability in order to address performance goals properly.

2.2.1 Content Durability

One of the most important aims of replication schemes is to provide data durability against failures [12, 114, 14, 60, 107]. To achieve this goal, replication schemes maintain a target number of data copies across different CDN nodes. They also deal with data placement, where sources might be grouped in different classes and allocated properly in order to improve data reliability. Reliability is an essential feature for storage systems, contributing to the design of fault-tolerant services. Thus, content durability provides system reliability by increasing resiliency to faults or failures. Essentially, there are two common approaches: proactive and reactive replication.

2.2.1.1 Proactive Replication

As proposed by Sit *et al.* [125], replication mechanisms must maintain a pre-defined, constant set of replicas across the system to cope with failures. The procedure is straightforward. Whenever a failure is detected, e.g. in response to a crashed node, the replication scheme proactively or reactively creates new copies of lost content. The goal is to have at least N identical replicas for any content, where N is the guarantee of failure resilience. An intrinsic problem of this approach is to distinguish between permanent and transient failures in order to avoid unnecessary network and storage usage.

Ford *et al.* [53] have extensively studied this issue through a statistical model in Goggle datacenters. They highlight that simple placement policies contributes significantly to improve content durability. These policies include to place content from the same replication set in different hacks inside a datacenter or even in datacenters remotely located to each other. For example, assuming N equals to four, they found that we can improve durability by two orders of magnitude by placing half of replicas in different datacenters, as depicted in Figure 2.1. The trade-off is higher cost in network usage to recover replicas between different datacenters.

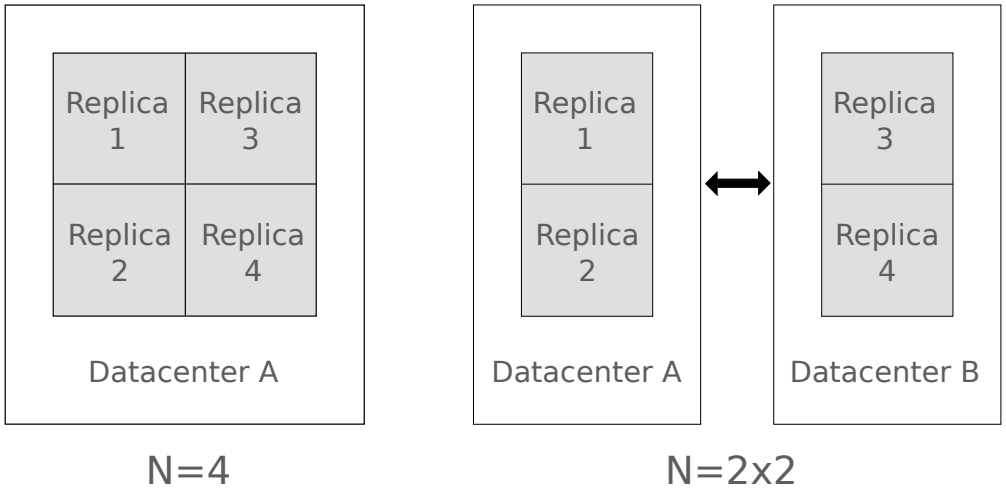

Figure 2.1: Increasing durability by simply placing copies in different datacenters.

2.2.1.2 Reactive Replication

Replication schemes based on erasure coding [108, 139] allow systems to ensure content reliability with better storage efficiency. Using erasure coding, a content of size $\mathcal{M}$ is divided in m pieces of size $\mathcal{M}/m$. Then, these m pieces are encoded into n coded pieces, with $n > m$,using a (n, m) maximum distance separable (MDS) code, to be finally stored in n different nodes. The original content can be recovered from any set of m coded pieces. We show in Figure 2.2 a scheme comparing replication and erasure coding. In the figure, content is replicated in N equals to three replicas, while erasure coding uses a $(9, 6)$ setting.

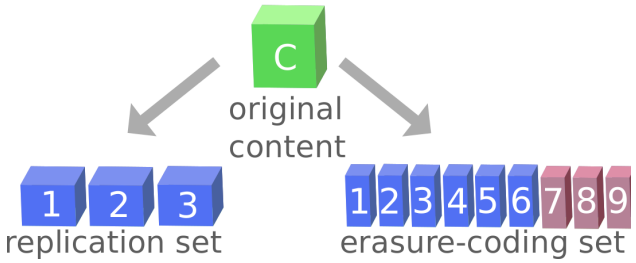

Figure 2.2: Comparing proactive replication to erasure coding.

To enhance durability with optimal trade-off between redundancy and reliability, many storage system designs like Total Recall [14] and Pond [106] utilise a replication scheme based on erasure coding instead of proactive replication. However, the benefits of erasure coding in CDN might be outweighed by two important factors. First, it adds architectural complexity to system design. Second, it does not contribute to improve availability for content delivery.

2.2.2 Content Availability

A step forward of replication scheme is to enhance storage system performance through data availability. We assume that data availability measures the easiness of data being fetched by customers. We are particularly interested in two aspects of that measure: the geographical location and the bandwidth availability.

Coping with geographical location requirement permits us to place data close to end customers, reducing communication latency. This requirement is met by placing data according to the geographic locations of the data requests. This has been addressed through a number of state-of-the-art works in datacenters in the recent years. Loukopoulos and Ahmad [83] show that locality-aware replication improves response time and saves network resources. Shvachko *et al.* [118] and Terrace *et al.* [131] highlight that system throughput can be increased when data placement takes into account locality. Considering CDNs, recent works [68, 130] show that latency-based placement of replicas and nearest-source selection is the common way to improve network resources provision for customers. This includes the usage of caching mechanisms. Figure 2.3 depicts how geographical location problem is commonly addressed by the nearest replication selection in CDNs.

Yet coping with bandwidth availability for data is more challenging, particularly if a specific performance goal is drawn [46, 50, 133, 79]. In this thesis, we are particularly interested in studying replication schemes that provide data or content availability in a self-adaptive manner. We provide further information about network provision and content availability in content distribution networks and datacenters in Chapter 4. In that chapter, we introduce definition and relevant approaches to cope with content availability, as well as we highlight the importance of self-adaptive replication schemes for enforcing cloud services. We also show some insights into our research contributions.

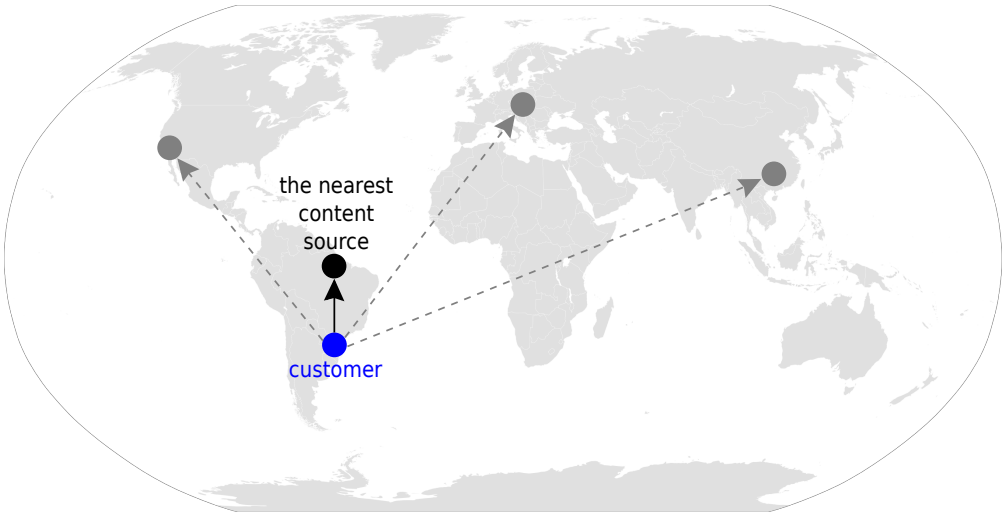

Figure 2.3: Selecting the nearest content source in content distribution networks.

2.3 Caching and Adaptive Replication

Caching is the most common mechanism to enhance performance in content distribution networks. It is the simplest way to meet Internet content demand, and offer popularity-aware content replication. Caching is also highly scalable, since resource allocation is spread across nodes in a rather distributed way. In this section, we list the well-known local caching policies. Then, we categorize caching strategies into collaborative and non-collaborative.

2.3.1 Caching Policies

Fortunately, there exists a very large literature on caching policies, particularly for web caching. They essentially establish how web content is replaced in distributed caching.

One of the most well-known state-of-the-art example is the Least-Recently-Used (LRU) policy. It puts emphasis on temporal locality, assuming that recently accessed content is more likely to be accessed again in the future. Likewise, Least-Frequently-Used (LFU) policy performs cache replacement, but instead of swapping contents based on temporal locality, it considers the number of access, providing a basic popularity-based replacement policy somehow. The bare assumption is that customers are more likely to access in the future contents which are frequently accessed. Neither LRU nor LFU takes directly into account the content size. For that, there exists Largest-File-First (LFF) policy that assumes that small contents have higher likelihood of being accessed again in the future.

Many studies have investigated the performance of basic cache replacement policies in distributing web content. In their remarkable work on the subject, Breslau *et al.* [17] show that the *independent reference model* [49], initially proposed in operating systems paging studies [42], fairly explains the distribution properties of web content access, including a rather Zipf-like dis-

tribution behaviour. They highlight that under this model, frequency-based policies are particularly efficient. Recent studies [51, 129] confirm that new web media popularity still follows a Zipf-like distribution.

However, Barford *et al.* [13] results suggest that Internet content access patterns have become less sensitive to temporal locality [6, 10, 59] and small-sized contents [37], therefore contributing to diminish the performance of LRU and LFF respectively. For LFU, Arlitt and Williamson [10] suggested that web content popularity pattern is bursty, which undermines LFU performance due to cache pollution.

In order to address well-known web content properties, such as highly variable content sizes and burst of content access pattern, and to overcome performance issues of basic policies, many enhanced variants [142, 94, 75] of both LRU and LFU have been proposed ever since. A good example is the GreedyDual-Size(GDS) policy, initially proposed by Irani [71]. GDS allows caching to be sensitive to both variability in the content size and retrieval cost, a penalty for content misses. Yet, it remains unclear how well all these policies perform under specific workload of newborn cloud application and SLA-based service provision.

2.3.2 Caching Strategies

We focus on two kinds of caching strategies: non-collaborative and collaborative. We highlight that our main goal is to distributing Internet content, hopefully according to content features as popularity and size. Considering this goal, a good caching strategy provides to the replication scheme to adapt the number and content location regarding its demand.

Non-collaborative caching remains the simplest approach to provide replication of Internet content[73, 74]. It adapts the replication degree to the content popularity using cache replacement policies, and assuming fair-sharing as scheduling policy. Replacement policies, such as basic ones described on Subsection 2.3.1, are enforced locally. Therefore, non-collaborative caching is highly scalable. This strategy is simple to adopt and it performs efficiently with generic Web workloads with infrastructure-based architectures, i.e. content distribution networks built on top of big datacenters. However, its distributed intrinsic functioning undermines resource allocation efficiency, contributing to the waste of resources.

In order to cope with this issue, collaborative caching adopts a different strategy. Caching levels or content sources communicate to each other to enhance overall performance or meet the system's goal. A very popular technique to improve global delivery system performance is to set up a caching hierarchy [11]. In this approach, different levels of caching cooperate in order to delivery content. Rodriguez *et al.* [109] found that despite resulting unwanted redundancy across the levels, hierarchical caching enhances connection times.

Distributed caching [104, 132] emerged as a promising alternative to hierarchical caching. Basically, it differs in caching collaboration. In fully distributed caching, only caches in nodes at the edge of the delivery system collaborate. This provides shorter transmission times and higher aggregate bandwidth usage than hierarchical caching. Rodriguez *et al.* [109] also show that a well-configured hybrid design can combine advantages of both, permitting delivery systems to meet specific performance goals. For instance, content sources can directly exchange messages

in order to enforce a minimum number of replicas, or to create more copies for contents that require higher bandwidth or lower latency. Communication can take place in different levels: datacenters and surrogate/edge nodes. The protocol to exchange these messages depends on the infrastructure architecture. For example, Figure 2.4 depicts a hybrid, simplified caching infrastructure for content delivery networks. In this infrastructure, there are two caching layers, one in the core network and a second one in the edge network. Nodes of each level collaborate to deliver content from content provider to end customers properly.

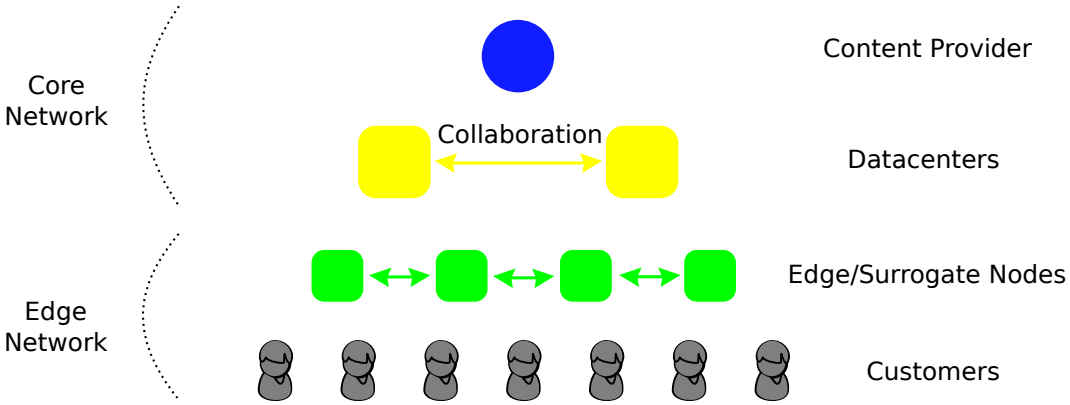

Figure 2.4: Collaboration in different caching layers of content delivery networks.

2.4 Replication Strategies to Improve Resource Allocation

In this thesis, we are interested in studying replication schemes whose common goal is to create and maintain copies in content delivery networks and storage systems. According to the resource allocation strategy, we categorize replication schemes as uniform, proportional, and adaptive.

2.4.1 Static Replication Strategy

Most of the biggest Internet companies, including Google, rely on content delivery networks based on datacenters, the so-called *infrastructure-based CDNs*. Broadly speaking, these datacenters differ in size, rely on commodity servers, and are deployed close to end customer, reducing latency of content retrieval, and improving content delivery performance. Inside the datacenter, as in any other distributed storage system, the primary goal is to ensure content durability.

To this end, the Google File System (GFS) [55] and Ceph [140] adopt a pragmatic strategy where the number of replicas per object is fixed and static. This approach has had a considerable success in the industry, particularly for datacenters deployment, because it is easy to adopt, ensuring content durability as a function of the number of replicas. It normally relies on overprovisioning to meet the average customer demand.

Yet static replication offers poor content availability, being inefficient to delivery popular content. Since infrastructure-based deployments are based on over-provisioning, they are normally quite expensive to deploy and maintain.

2.4.2 Proportional Replication Strategy

Proportional replication strategy was introduced to permit systems to perform replication according to content demand. They are simple, easy to adopt, and enhance resource allocation when demand varies.

Most replication schemes that rely on caching mechanism fall into this category. Using caching, content can be replicated proportionally to the number of requests, as detailed in Section 2.3. Indeed, caching outperforms static replication in terms of performance. In order to ensure data durability, a replication scheme might enforce a minimum number of replicas through a uniform strategy, and rely on caching for performance. To enhance resource allocation, proportional replication strategies can include resource utilization. Cohen *et al.* [33] initially suggested that storage capacity and bandwidth must be taken into account to enhance proportional replication algorithms, but their aim was specialized to minimize the expected search size in unstructured P2P networks. Adya *et al.* [3] and On *et al.* [92] propose an interesting proportional replication schemes based on availability of untrusted storage nodes.

Overall, the proportional strategy provides a big improvement on performance for content delivery network compared to uniform replication strategy, particularly for popular content. They are recommended for general-purpose content distribution systems, where specific performance goals, like latency of bandwidth, are not required.

2.4.3 Adaptive Replication Strategy

We consider adaptive replication strategy the best way to meet specific performance goal. In this thesis, we focus on content availability, and we expect that a *good* adaptive replication scheme could enhance bandwidth allocation in content delivery networks.

Carbonite [28] extends proportional replication strategy's concepts and introduces an adaptive replication scheme that takes into account both availability (for retrievals) and durability (for additions) of stored objects. On the one hand it shows how much bandwidth usage is important for replication schemes, but on the other hand their assumptions are based on a very idealized mathematical model, that ignores object popularity and node overhead. We assume that tracking popularity growth and load over nodes are essential for enforcing strict performance goals in terms of content availability. EAD [116] and Skute [15] tackle these issues by using a cost-benefit approach over decentralized and structured P2P systems. EAD creates and deletes replicas throughout the query path with regard to object hit rate using an exponential moving average technique. Skute provides a replication management scheme that evaluates replicas price and revenue across different geographic locations. Skute's evaluation technique relies on equilibrium analysis of data placement. Despite being highly scalable and providing an efficient framework

for replication in distributed systems, these approaches result in inaccurate bandwidth allocation, hence inappropriate for high-quality content delivery.

More recently, Zhou *et al.* proposed DAR [144], an adaptive replication algorithm for P2P-assisted Video-on-Demand (VoD). DAR permits distributing content in scalable way by balancing the expect bandwidth load per node. But, it remains unclear if DAR approach can enhance content availability when bandwidth allocation is the main performance target.

2.5 Replication Schemes in Peer-to-Peer Networks

A large number of replication schemes have been proposed in peer-to-peer networks. In this section, we provide a brief overview of replication schemes in peer-to-peer systems, highlighting the main insights for current content delivery designs.

One very important family of replication scheme emerged in 2001 with distributed peer-to-peer storage systems based on distributed hash table's key-value operations. This includes the Cooperative File System (CFS), designed by Dabek *et al.* [38] and based on Chord [127], and PAST, a large-scale, persistent peer-to-peer storage utility, that was designed by Rowstron and Druschel [111] and was implemented over Pastry [110]. Their replication schemes are quite similar. CFS replicates objects in a static, predefined number of DHT successors of a node, constituting the replication set. PAST performs similarly, using the term of leafset for denominating neighbours peers of a replication set. Both systems allow the system to cache objects throughout the DHT routing path. Actually, Shen [116] wisely explored and extended this caching mechanism in order to design the adaptive replication scheme EAD. These approaches are resilient to failures, scalable, and reliable in terms of data durability. However, the lesser control of replica allocation and maintenance undermines the enforcement of performance guarantees.

In order to enhance performance guarantees, On *et al.* [93] highlight that the number of replicas and their locations are two very important features for improving QoS guarantees for end customers. But their approach requires global knowledge of resource consumption in the system. Landers *et al.* [78] propose to improve DHT-based backup systems performance through replication reciprocity among node. Indeed, this is intended to reduce the impact of problems caused by churns and the amount of transmitted data amongst nodes. But replication reciprocity implies symmetric of node resources, what can result in long lookup time for finding peers, harming replication scalability. Meanwhile, BitTorrent [31], a well-known file-sharing, swarm-based peer-to-peer protocol, has gained an increasing attention from researchers. Using a straightforward replication scheme based on current peers interested in a content, namely content swarms, BitTorrent has coped with many performance issues [103, 113] of previous peer-to-peer designs. It adapts the number of sources dynamically and provides better network resource allocation, improving content availability. For instance, Peterson *et al.* [102] propose a mechanism to improve bandwidth provision from sources and increase aggregate bandwidth for content delivery networks. However, there still is a poor literature in content delivery using BitTorrent to meet specific QoS goals in the emerging cloud paradigm.

2.6 Summary

In this chapter, we introduce the most relevant replication scheme's issues for this thesis. Since our study focuses on content delivery networks (CDNs), we are interested in using replication schemes as a technique for enhancing content availability for customers.

We assume that replication schemes are mostly intended to provide content durability and availability. While durability involves content resiliency, availability measure the easiness of accessing contents, particularly in terms of network resource provision. We showed that caching is a key mechanism for content delivery, and has been widely used in CDNs. We described the most relevant caching policies, including LRU and LFU. We explained how they are combined with different strategies, and how it is linked to content delivery.

We described and classified replication schemes according to resource allocation strategy in three groups: static, proportional, and adaptive. We highlighted the main advantages and weakness of each one of them, showing that adaptive replication schemes are better in improving specific performance target. That was followed by a section that briefly described the most important issues involving P2P networks, replication schemes, and content delivery networks.

In the next chapter, we will present with more details how content delivery networks are structured, and provide further information about how replication schemes in CDNs.

Chapter 3

Content Delivery Networks

Contents

3.1 Introduction

The increasing popularity of web content in the late 90's had made content and network providers pay a particular attention to reliability and scalability of globally distributed services. The deployment and maintenance cost of infrastructures to permit this growth has become one of the major concerns of Internet players. To cope with these issues, emerging, pioneer Internet companies [87, 47, 80, 4, 43] proposed Content Delivery Network (CDN) infrastructures.

CDNs are distributed systems that maintain content servers in many different locations in order to cope with load management in scalable way, and also to enhance latency and bandwidth

available for end customers. Since CDN infrastructures are shared amongst different content providers, the overall cost of content delivery might be considerably reduced.

In general, CDNs deploy or outsource infrastructure across the Internet to allow origin content providers' servers to reach end customers. Their infrastructures might differ in many different aspects, e.g. the geographical coverage, dedicated content type or Internet service, according to Service Level Agreements, or deployment policies. One of the main infrastructure components is the delivery server, also known as surrogate or edge server. Actually, customers connect to these servers to retrieve content. Therefore, there exists a particular interest in monitoring and selection of these servers for delivering content and enforce performance metrics. Based on the delivery servers deployment, we propose a classification that will be useful for providing a better understanding of the context of our research.

Nowadays, one of the most appealing aspects of CDNs is the video delivery, including streaming and live videos. To meet the increasing demand for high quality content broadcast, CDNs must cope with low starting and rebuffering. In addition to provide a reliable and scalable infrastructure, CDNs must now enforce strict Quality of Service metrics, such as bitrate. This chapter does not cover directly performance issues. We address CDN's performance in Chapter 4. Security is also important in CDNs [20], that ranges from preventing Distributed Denial of Service (DDoS) attacks to cybercrimes on the whole. Moreover, mobility has become a hot topic for CDN researchers. For instance, how to handle customer mobility of devices, like tablets and smartphones, in order to deliver content properly. However, since security and mobility in CDNs are outside the scope of this thesis, they will not be discussed in this chapter.

We present this chapter with the following organization. We explain briefly the CDN's fundamental techniques in Section3.2. Then we discuss the CDN's main building blocks, components, and their interactions in Section 3.3. We classify CDN infrastructures according to delivery server location in Section 3.4. In Section 3.5, we provide an overview of hybrid CDNs, and peer-assisted content delivery systems. We list some of the most relevant challenges faced by content and CDN providers in Section 3.6. Finally, we conclude in Section 3.7.

3.2 The Fundamental Techniques for Delivering Content

There are two fundamental techniques for delivering content in the Internet: hierarchical caching systems and datacenter deployment.

Hierarchical caching permits to improve content delivery performance by proportionally adapting the number of replicas according to the customer demand. By placing popular content in surrogate servers, it reduces latency and saves bandwidth.

The deployment of datacenters has been extensively used in the Internet for many years now. Mostly based on commodity servers, they allow content providers to outsourcing infrastructure and enhance scalability.

3.2.1 Hierarchical Caching Systems

Hierarchical caching systems are the most common way to improve the performance of content delivery [35]. It also allows us to reduce bandwidth consumption and latency between the content source and destination. For that, a hierarchical caching system instruments cache nodes or proxies across the Internet.

Customers connect to the their preferred caching proxy to in order to download the content with reduced network cost, e.g. according to latency or jitter measurements. Hopefully selected caching proxy maintains the most wanted contents in its cache memory for responding to customer's requests accordingly. Whenever a caching proxy handles a request on behalf of customers, it must interact with parent caching proxies in order to fetch content and ensure content consistency.

Parent caching proxies might cooperate in order to ensure data durability, as described in Section 2.3. Similarly, parent caching proxies might perform content delivery partitioning according to the content providers in order to enhance the use of caching proxies. Figure 3.1 depicts the functioning of hierarchical caching systems.

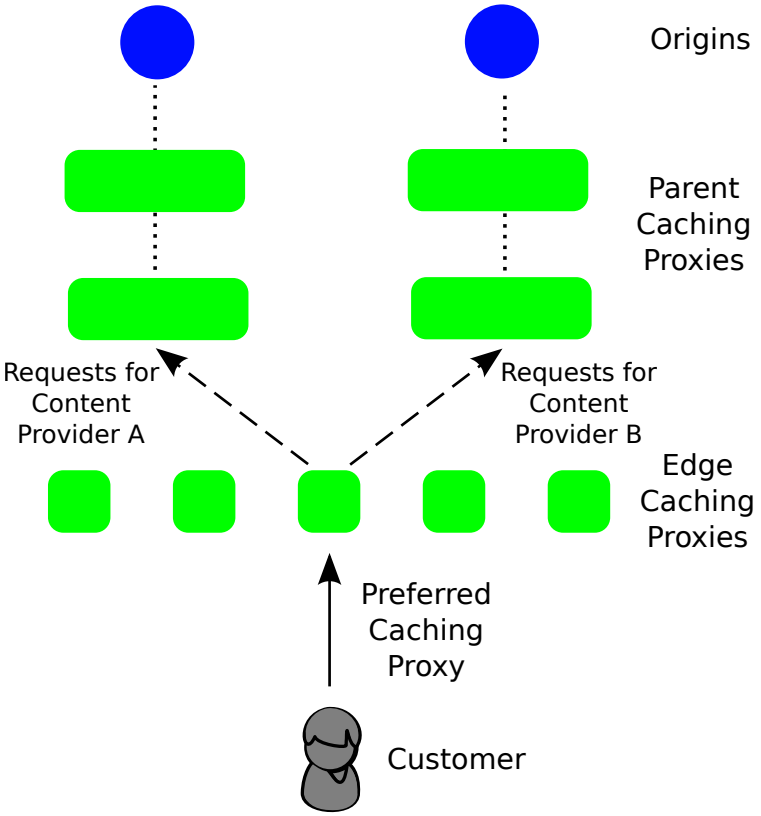

Figure 3.1: Hierarchical caching systems for content delivery.

3.2.2 Datacenters

Datacenters have been widely used to deliver content to the masses. Nowadays they may contain tens of thousands of servers with huge amounts of storage, processing, and aggregate bandwidth. Internally, it permits to compute customers' requests and dispatch them throughout large group of servers [40, 18]. Datacenters providers a number of interesting features for content delivery, such as emulating a cluster of servers acting as an single origin server, performing load-balancing across different group of servers, enforcing content durability and resiliency against failure, and enhancing resource allocation for customers in an *elastic* way, where resources are allocated according to the demand.

Datacenters can improve significantly content delivery scalability, particularly in terms of storage. However, they are commonly deployed in core network, relatively close to origin. Thus, their performance might be undermined by poor network availability or congestion across the Internet before reaching end customers.

3.3 Components and Building Blocks of Content Delivery Networks

In order to provide a better understanding of the basic functioning of CDNs, we provide an overview of the node roles, components, and customer interaction in this section.

3.3.1 The Main Roles of Servers in CDNs

There exist two main node roles for delivering content in CDNs: content origin and delivery servers, so-called surrogate or edge servers [98].

3.3.1.1 Origin servers

Origin servers are the start point of content delivery and may not be part of CDN provider infrastructure. They store contents that come directly from the content or service provider, the major customer of CDN providers. They rely on CDN infrastructure to delivery a wide range of contents and services. Contents basically differ in nature, for instance:

- **Static Content**: Data that is stored once in the origin server in a read-only manner, and then is distributed. This category includes photos, e-books, as some file sharing services.
- **Interactive Services**: This category is mostly related to web services that allow users to maintain their own data or handle online transactions over the Internet. For example, this includes backup and synchronization services, directory services, and e-commerce.
- **Streaming Content**: Streaming services have attracted an increasing attention of both content and CDN providers. There exists a growing demand for this type of content with a

constantly increasing level of QoS expectations, particularly in terms of response time and bitrate. This is mainly the case of video delivery, including audio and video on demand, and live streaming.

Considering the storage properties of replication schemes described in Chapter 2, origin servers eventually provide content durability, while delivery servers inside the content delivery system provide rather content availability, focusing on delivery performance.

3.3.1.2 Delivery Servers

Delivery, surrogate, or edge servers play the main role in CDN infrastructure. They are deployed across the Internet in order to distribute content to end customers. Deployed in datacenters or in edge network clusters, they constitute a distributed storage system to disseminate content. In this thesis, we assume that delivery servers contribute to the performance of content delivery by replicating data in different locations. We assume that caching is the most common mechanism for performing replication in delivery servers. We provide further information about delivery servers in Section 3.4.

3.3.2 Infrastructure Components

Based on the the simplified scheme for CDNs proposed by Rahul *et al.* [105], we consider that there are three main components in CDNs infrastructures: monitoring, distribution system, and request-routing system [23]. We provide a simplified scheme of these components in Figure 3.2, and describe their functioning as follows:

- **Monitoring** component plays a key role in CDN's infrastructures. It continuously collects measurements of CDN servers and customers. This data is essential for performance analysis and for tuning resource allocation in surrogate servers. For example, this module can be combined to hierarchical caching systems in order to provide collaborative caching, as discussed in Section 2.3.

- **Distribution system** controls the replication and placement of content that comes from the origin servers. Its main role is to provide content availability for customers. In cloud-like applications, a distribution system may work by enforcing Service Level Agreements, where Quality of Service metrics are defined either by content providers or customers. This component gets information from monitoring component in order to enhance resource allocation and content replication. For instance, SOLA Sphere content distribution from Akamai [126, 4] offers a dynamic resource allocation for content delivery based on the platform utilization.

- **Request-routing system** is the most important component in content delivery infrastructures. Roughly speaking, it receives content requests from customers and sends replies by

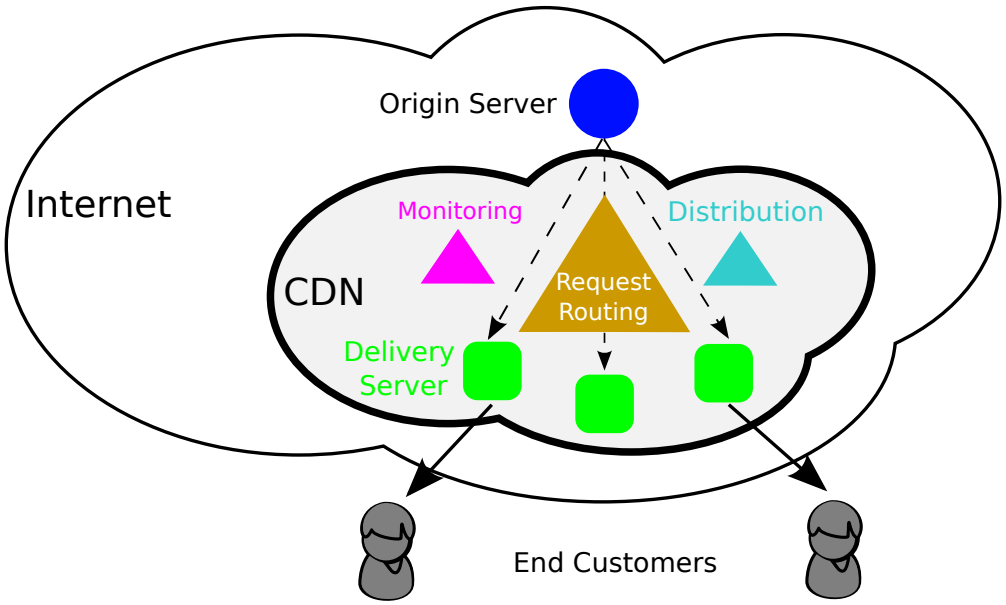

Figure 3.2: Main Components of Content Delivery Networks.

forwarding them to the *best* surrogate nodes. Surrogate server selection is based on several criteria ranging from performance issues, such as load balancing, to contractual terms or high-level policies. For that, it maintains an updated inventory of content from different origins. Good examples of routing mechanisms for CDNs include Resilient Overlay Networks project at MIT [8] and the Detour project at University of Washington [34].

3.3.3 Interactions Between Content Provider and Customers

We provide in Figure 3.3 a highly simplified scheme of interactions between customers and content provider through CDNs. We describe this scheme step-by-step as follows:

1. Disseminate Content. Content provider sends content from its origin server to the CDN's infrastructure provider.

2. Get Content's URI. CDN provider replies the dissemination requests with the Uniform Resource Identifier (URI) for the just-added content.

3. Content Publishing. Once the content provider has the content's URI, it publishes the content in the Internet, e.g. using its web page.

4. Content Lookup. Therefore, customers are able to look up the content in the publishing platform, such us the web site.

5. Retrieve Content URI. The customer retrieves the information about the content location from the content provider.

6. Search the Content in the CDN. Customer sends a request with the content's URI to the CDN provider in order to know where the content is actually stored.

7. Forward Customer's Request to the Proper Surrogate Server. The customer's request is processed by the CDN's request-routing system, which selects that *best* surrogate server on behalf of the CDN provider and sends the right Uniform Resource Locator (URL) to the customer.

8. Send Content Fetching Request. Since the customer has the URL to the content, she sends a fetching request to the node that stores the content in the CDN.

9. Fetch Content. Finally, the customer fetches the content directly from the right surrogate server.

While customers fetch content from the system, monitoring and distribution module continuously interoperate to improve customers' experience, meet performance targets, and enhance resource allocation.

3.4 Classifying Content Distribution Networks According to Delivery Servers' Location

We can therefore differentiate CDNs on the basis of their surrogate servers placement, and classify them into core and edge architectures.

3.4.1 Core CDN Architecture

Core CDN architectures rely on private datacenters deployment close to ISPs' points of presence (PoP). This has been a successful approach used by pioneers as Akamai [4], as well as by major content and service providers. The Akamai platform [91] has been built on top of large number of small server clusters highly distributed in many different countries. Hence, such architectures require complex algorithms for locating and delivering content properly, e.g. very precise infrastructure mapping and monitoring.

Some content providers, including Amazon [7] and Google [57], and service providers, such as Limelight [80], have opted to deploy very expensive and large datacenters in very few strategic locations. As core architectures are connected to PoPs, they do not have control of traffic throughout ISP until the end-customer, that undermines QoS guarantees enforcement.

3.4.2 Edge CDN Architecture

Interoperable CDNs in edge networks have emerged to tackle directly these issues. Internet Service Providers (ISPs) look forward to (i) taking advantage of their infrastructure, (ii) deploying

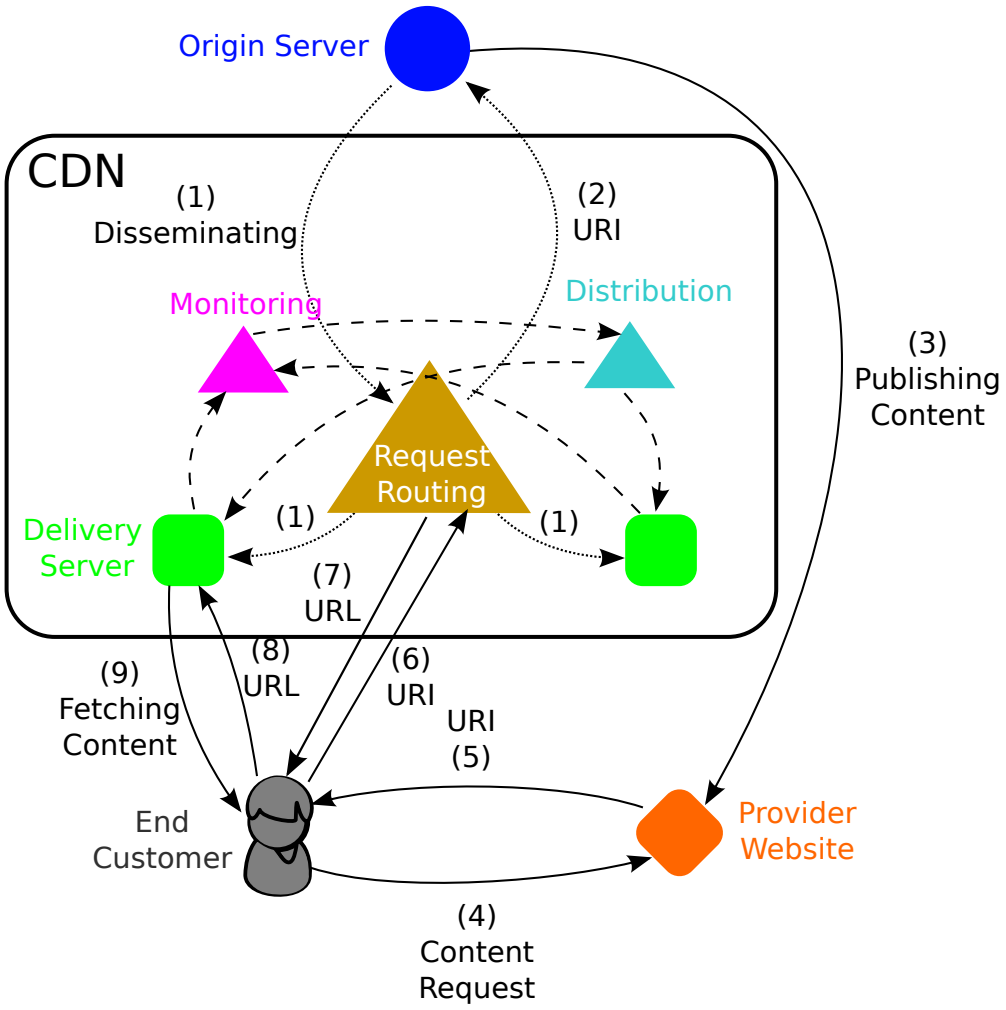

Figure 3.3: Simplified scheme for publishing and fetching contents in CDNs.

their own datacenters, and (iii) delivering content as close as possible to end-customer. The aim is to be able to offer differentiated QoS guarantees to regular customers [95].

Another highly distributed approach of edge CDN architectures is P2P network content distribution. This consists of content servers deployed on consumer-edge devices, where peers cooperate to share and distribute the content. P2P network distribution comprises video stream handlers, such as PPLivehuang2007experiences and Zattoo, and content swarming, e.g. BitTorrent [32], eMule [76], and NaDa [134], a distributed content distribution platform based on nanodatacenter in home gateways. NaDa relies on BitTorrent protocol to manage unused edge resources. In this work, we are mostly interested in challenges risen by edge CDN architectures.

3.5 Hybrid Designs in Content Delivery Networks

In order to improve the performance, scalability and energy efficiency of content delivery, content must be placed as close as possible of end customers. This has motivated CDN providers such as Google [55] and Akamai [4] to invested huge sums of money in the deployment and maintenance of datacenters in different locations. Even though, the enforcement of some QoS metrics, like target bitrate, remains quite hard to meet due to the unpredictable availability of networks resources.

Edge networks can contribute to enhance the placement and network resources availability by placing content directly in customers' devices. CDN providers could include devices deployed by Internet Service Providers (ISP), as set-top-boxes or home gateways, to extend CDN infrastructure in so-called hybrid CDN designs. As described in Subsection 3.4.2, ISP's devices might be seen as first-class nodes in terms of content availability since they are deployed close to the end user. They allow to provide content delivery with very low latency, contributing significantly to improve performance, and save storage and network resources. However, an efficient utilization of these resource for network providers is not a straightforward task, as described by Hei *et al.* [64].

To cope with this issues, a new class of peer-assisted content delivery systems [67, 115, 25] has emerged. In these hybrid designs, edge or surrogate nodes collaborate to the content delivery using the peer-to-peer paradigm. For instance, BitTorrent [32] protocol can easily be deployed in edge devices in order to provide robust content delivery. Furthermore, Valancius *et al.* [134] show that a coalition ISPs and CDN providers can improve energy efficiency while improving content availability. Considering network resources usage in edge nodes, Peterson and Sirer [101] proposed Antfarm, a content delivery system based on BitTorrent protocol that enhances the aggregate bandwidth in peer-assisted architectures by collecting and computing active measurements of swarms activity. In their latest work [102], they were able to improve still further the aggregate bandwidth in edge nodes by introducing a Content Propagation Metric (CPM) that captures how quickly a content is disseminated through swarms in hybrid CDNs.

3.6 Challenges

The widespread availability of high-speed Internet connections has reshaped the way we consume online content. While content has become increasingly costly in therms of computational resources, customers expectations on the performance of content delivery networks has evolved. In order to meet increasing consumers' expectations, a *good* CDN system must overcome the following challenges:

1. It must cope with dramatic, unexpected variations in content popularity.
2. It must avoid waste of resources, and reduce as much as possible storage and network usage, specially in edge networks' nodes.
3. It must provide small response time delay and high bandwidth availability through a design improvements and self-adaptive techniques.

We describe in Chapter 5 our contributions for coping with these problems. We present how CDN providers can take advantage of hybrid designs in order to cope with these issues successfully. Our simulations suggest that our approach meets consumers' expectations on nowadays online content properly, specially under heavy load. Finally, we show that our approach produces a good balance between resource allocation and users' satisfaction.

3.7 Summary

Content Delivery Networks (CDNs) are scalable and widely distributed systems for disseminating content in the Internet. They provide a globally distributed infrastructure for content providers with reduced cost and enhanced content delivery performance. CDNs have been continuously improved since their creation in the late 90's, becoming increasingly complex and heterogeneous distributed systems. Hopefully there exists a rich literature about CDNs, including a number of surveys [98, 96, 27], books [22, 65], and technical reports written by Internet companies [4, 80, 55]. Therefore, we do not provide a fully comprehensive guide about CDNs in this chapter. Instead, we highlight the most relevant matters for easing the understanding of our work.

We briefly showed the two most important techniques for building content delivery systems, namely hierarchical caching and datacenters' deployment. Then we discussed about the main components and building blocks that were heavily used in this work, notably in Chapter 5. This included the essentials of CDN's components role and interactions. This chapter also covers the two most common designs for CDN architectures, either core or edge architecture. In addition, we presented hybrid CDN architectures as a step forward to improve content delivery performance. We provided an overview of peer-assisted content delivery systems, particularly those based on BitTorrent protocol.

Chapter 4

Content Availability in Content Delivery Networks

Contents

4.1 Introduction

Internet content providers rely mostly on contend delivery networks' (CDNs) infrastructure to reduce costs and improve the performance of content delivery. On behalf of the content provider, CDNs replicate content in different geographical locations, reducing latency to end customers' retrievals and providing content resilience or durability against system faults. Despite that, customers have become increasingly sensitive to multimedia content distribution, as Internet connection is widespread and continuously faster. Today, customers expect that Internet content

will be promptly available, otherwise their interesting in a particular content or service naturally declines.

To meet the increasing customers' expectations for content delivery, CDN providers must consider availability as a key metric of their infrastructure's performance. Not only CDNs must deliver content with low latency, but they must ensure enough bandwidth for fetching a content according to its demand. Therefore, we assume that content availability measures how easy a content can be fetched by customers from a CDN, as described in the Section 2.2 of the previous chapter. To cope with content availability issues, we consider content replication scheme as a key mechanism to provide content availability. So that, we are particularly interested in studying how data placement and the adaptive replica creation could enhance content availability.

This chapter gathers the most relevant points in content availability for developing our research work. We will discuss the main technical aspects and issues related to content availability on the Internet. We will pay a particular attention to the availability of Internet videos, since it has become one of the most popular services around the Web. The chapter is therefore organized as follows. We describe in Section 4.2 how the enforcement of Quality of Service metrics may contribute to improve the availability of online content in CDNs. Then in Section 4.3, we report on the availability of Internet videos, focusing on video-on-demand services, as YouTube. We show which role Peer-to-Peer (P2P) networks can play in order to improve content availability in Section 4.4. Finally, Section 4.5 summarizes the main points of this chapter.

4.2 Improving the Availability of Online Contents through Quality of Service Guarantees

The increasing competition between network service providers, along with ever-growing demand for multimedia content, push through tighter delivery guarantees. Consumers and providers engage in Service Level Agreement (SLA), that formally establishes which system performance is expected for a particular service. Generally speaking, system performance is defined through Quality of Service metrics to be enforced. A SLA violation happens whenever the system fails to enforce SLA's metrics to a customer.

A new range of cloud content delivery services have emerged in recent years, but to the best of our knowledge, none of them are really able to offer strict QoS metrics for clients, like bitrate. The most common way to provide better QoS guarantees is by over-provision, either by increasing backbone throughput rates or deploying big datacenters around the world. However these approaches are expensive and counter-productive for improve QoS for final users, because their content's sources remaining far enough from end-users that turns network delivery metrics such latency and mean throughput unpredictable. We are able to overcome these problems in edge networks and finally define precise QoS metrics on a new era of SLA contracts. That takes advantage of nearby sources and contributes significantly to offer services with better quality, which tackles more efficiently popularity growth issues, and increases network service providers revenues. As described in Chapter 3, hybrid CDN designs are able to manage resource alloca-

tion on edge nodes. In this chapter, we consider that these infrastructures may improve content availability through a SLA-driven services. Moreover, we consider that the QoS metric of a SLA contract determines the level of availability of a content for customers. According to the QoS metric and target content availability, we classify SLA contracts in common and strict.

4.2.1 Common SLA contracts

Request rate and response time are commonly included in current negotiated contracts [41]. Request rate defines the client's expected request rate for a particular service, while response time measures the time delay for treating a request and starting fetching a content. In the industry, both metrics are normally expressed in terms of means, median, and expected variance. To improve performance for customers, response time might be expressed and measured at the 99.9^{th} percentile of the distribution, as in the case of Amazon [1]. However, they provide low content availability since they not provide any guarantees of content retrieval speed or overall download time.

4.2.2 Strict SLA contracts for Highly Available Contents

We assume that CDN providers must offer highly available contents by enforcing SLA contracts that include QoS guarantees of the content download time, so-called high-quality QoS metrics. However, high-quality QoS metrics, such as end-to-end latency and strict bitrate, are avoided by providers because they are very tough to enforce. The most common way to achieve this goal would be the use of well-known network resource reservation protocols. Integrated Services architecture (InServ) with Resource reSerVation Protocol (RSVP) guarantees QoS metrics by reserving end to end resources before sending any data, but suffers from poor scalability. Differentiated Service (DiffServ) groups distinct traffic flows in classes and configures routers to correctly follow a Per Hop Behaviour (PHB) without resource reservation. However, it lacks proper end to end QoS enforcement due to PHB mismatches across different ASes.

To overcome these issues, Evans *et al.* [48] argue that appropriate engineering of edge networks is essential for strict SLA enforcement. Recent works have extensively studied this problem in datacenters networks [141, 5, 138]. D3 [141], provides a deadline-aware transport protocol that uses explicit rate control to apportion bandwidth according to flow deadline. Deadline-aware protocols provide mechanism for preventing SLA violations by ensuring the minimum bitrate for any transfer. They allow to increase the aggregate throughput in datacenter environments compared to TCP. However, deadline-aware protocols are particularly designed for transporting tiny objects from homogeneous nodes across datacenter Ethernets with very low delay and high throughput. Furthermore they are agnostic to popularity peaks, and they are not customized for wide area network environments with unpredictable transfer rate demands and high variances in network latency. Most of deadline-aware protocols, including D3, require changes in the network stack, which might undermine their deployment in globally distributed systems.

We assume deadline-aware protocols are a potentially important mechanism for providing highly available content in CDN networks. In Chapter 5, we study the feasibility of integrating

this mechanism in CDNs that use edge resources. More specifically, we show how we can improve content availability by combining bandwidth reservation and collaborative caching mechanisms.

4.3 The Availability of Internet Videos

Video distribution over the Internet has increased dramatically in the recent years. A study published by Cisco System, Inc [30] revealed that the global Internet video traffic has surpassed peer-to-peer traffic since 2010, becoming the largest type of Internet traffic. Cisco also forecasts that video traffic will reach 86% of the global consumer traffic by 2016, including TV, video-on-demand (VoD), live streaming, and peer-to-peer (P2P) file sharing.

One of the most remarkable examples of Internet video providers is Netflix [89]. Founded in 1997, it has now reached 36 million subscribers [90] around the world, mostly providing video-on-demand service. Since 2013, Netflix has also became streaming television network service, planing to compete with cable and network television. Another good example is Hulu [70], a website and on-demand streaming video platform, which delivers billions of videos per month, including television shows, movies, web series episodes, trailers, clips, as well as exclusive content from television channels like NBC, Fox, ABC, and TBS.

This increasing customers' interest in Internet videos has not only reshaped the Internet traffic, but also the way we watch television. Customers expect to watch Internet videos in increasingly bigger screens, and with higher video definition. Thus, content availability will be a essential feature of content delivery systems in order to meet customers expectations.

In this section, we highlight the main performance issues related to Internet video streams. We firstly describe CDN infrastructure issues and generic Internet video characteristics. Then, we present how to enhance video availability for the increasingly high customers' expectations. Finally, we focus on VoD services study, where we show key challenges and provide a deeper explanation about YouTube.

4.3.1 Evaluating Availability of Internet videos in Content Delivery Networks

In order to improve resource allocation for Internet video delivery, content and CDN providers has studied that characteristics of this kind of service, including traffic and system behaviour.

Some studies [51, 129] have drawn attention to reach a better understanding of Internet videos properties, such as popularity growth. They point out that well-known popularity characteristics on web content are also applicable to video delivery services. For instance, Internet videos popularity distribution follows power law, and popularity bursts have a short duration and are quite likely to happen just after the content publication.

More recently, Dobrian *et al.* [44] shed some light on the performance of Internet videos provision on CDNs. They show that average bitrate plays an important role in videos availability. This has motivated network designers and architects to improve content delivery infrastructure. For instance, a hybrid solution between CDNs and P2P is presented by Mansy *et al.* [85]. Their

aim is to model and analyse a live video system and one of their main concerns is to adapt bitrate for guarantee user satisfaction.

Similarly, Adhikari *et al.* [2] work described the YouTube video delivery system through measurements of DNS resolutions and video playback traces. One of their findings is that over a globally distributed network (PlanetLab) most part of the nodes have a nearby Youtube video cache server to delivery the video data. Moreover, Brodersen *et al.* [19] presented a detailed study over the strong connection between popularity and geographic locality of Youtube videos. These facts endure our decision of a locality aware solution for infrastructure.

Liu *et al.* [81] propose a different approach. They make a case for a video control plane that can use a global view of client and network conditions to dynamically optimize the video delivery in order to provide a high quality viewing experience despite an unreliable delivery infrastructure. However, the granularity of their server selection mechanism is at a CDN, ignoring edge network resources. As we show in Chapter 5, we are particularly interested in providing enhanced resource allocation in edge networks, and we have decided to cope with this issue by adapting replication degree close to the viewers. In any case, we suppose that our approach can play an important role in collaborating with an Internet control plane.

4.3.2 Delivering Internet Videos

The performance of Internet videos delivery has become an important issue for CDN and content providers. To ensure *good* performance of video delivery for meeting customers' expectations, many efforts have been spent in studying delivery infrastructure for video services and video traffic characterization.

4.3.2.1 Emerging Infrastructures and Mechanisms

Content delivery networks and content providers have made important investments in infrastructure in order to improve video dissemination. It appears that caching remains a key mechanism for video delivery. In an extensive network YouTube traffic analysis, Gill *et al.* [56] show that caching can improve significantly the user experience, and reduce the burden of core servers. Unlike common Web object, caching policies must rely though on rich metadata information gathered from users' interactions through Web interfaces. Moreover, content providers may rely on peer-to-peer protocols to enhance scalability, improve content locality, and reduce infrastructure costs. For instance, PPLive [63], one of the most popular mesh-pull P2P system, is able to gather thousands of simultaneous viewers watching video channels at rates between 300kbps and 1Mbps. Despite of providing relatively low rates, these video delivery systems are very scalable, require minimal infrastructure, and adapt well to highly dynamic, high-churn network environments.

4.3.2.2 Characterizing and Handling Video Traffic

As described in Section 4.3.1, studies [51, 129] have showed that well-known popularity characteristics are applicable to video content. Gill *et al.* [56] confirm these finds for YouTube workloads, highlighting that access pattern is strongly related with human behaviour, as traffic volumes vary significantly by time-of-day, day-of-week, as well as longer term activities, such as academic calendars. Despite that, these studies fails to define a trustful and definitive multimedia growth pattern due to the the inherent lack of details about publication, search and promotion engines used by content providers. In Chapter 6, we show that predictions on content replication may be an interesting alternative approach to overcome resource allocation problems caused by highly variable multimedia growth patterns.

4.3.3 Improving the Availability of Internet Videos to Better User Experience

Many studies have shown that quality of user experience while watching online videos is related to the good quality of content transmission. Still being an open issue, the community presented many strategies to cope with the video content availability and its distribution. The majority of the studies in the field are focused on Youtube, being this the major player of video content distribution [51, 2, 19, 16]. These studies vary from analyzes of crawled data from Youtube APIs to comparisons of caching strategies from collected data of users' point of view (HTTP logs from ISP or local networks).

Dobrian *et al.* [44] study have shown the correlation between the user engagement and the video quality, being the *Buffering Ratio* (fraction of the total session time spent in buffering) and *Rendering Rate* (frames per second) the most critical metric over the total played time for short videos, the current target of our method. This characterizes the relation between quality of service and user experience and endures the importance of avoidance of SLAs violations, which minimizes buffering ratio, confirming the main metric of evaluation in our work.

Furthermore, Finamore *et al.* [52] stated that download bitrate of the video has a fundamental role in video playback quality. They measured the smoothness of the playback by the bitrate ratio, defined as the ratio between the average session download bitrate and the video encoding bitrate. The consideration of different bitrates in the dataset used as input for our method agrees with the importance of the rendering rate metric in user engagement and playback quality.

4.3.4 Studying YouTube Workloads

YouTube [143] has become the largest video sharing site on the Internet [84]. Users are able to generate and distribute video content as a free video-on-demand service. Today's estimations say that YouTube accounts for at least 60% of videos watched on the Internet [56]. Therefore, performance analysis studies of YouTube traffic and users interactions are very important for understanding the functioning of large-scale content delivery. We focus on two YouTube users' interactions features: popularity growth curves and realistic video encoding settings.

Table 4.1: Advanced encoding settings for YouTube videos used in this work.

Type	Video Bitrate	Mono Audio Bitrate	Stereo Audio Bitrate	5.1 Audio Bitrate
1080p	50 Mbps	128 kbps	384 kbps	512 kbps
720p	30 Mbps	128 kbps	384 kbps	512 kbps
480p	15 Mbps	128 kbps	384 kbps	512 kbps
360p	5 Mbps	128 kbps	384 kbps	512 kbps

4.3.4.1 On the Track of YouTube Popularity Growth Curves

Figueiredo *et al.* [51] collected and characterized the growth patterns of YouTube videos, whose datasets are currently available online [1]. They analysed three types of YouTube videos sets: videos that appear on YouTube top list, videos that were banned from YouTube due to copyrights violations, and videos that were randomly selected through API calls. They crawled once a number of videos' daily features. For each video, there are up to 100 daily measurements, or daily available samples, per feature. In this work, we are mostly interested in the measurements of *view data* feature, that depicts the popularity growth curve of a video through a array of cumulative number of daily views ranging from 0 to the total number of views.

4.3.4.2 Realistic Video Encoding Settings and SLA definitions

In order to reproduce realistic, high quality videos encodings, we consider the YouTube advanced encoding settings[2]. Table 6.3 depicts the set of high definition (HD) video encodings that we use in this work.

In this thesis, we are particularly interested in realising the shape of realistic popularity growth curves, as considering advanced coding setting and common VoD demand patterns. We aim to be able to fairly reproduce YouTube workload for evaluating our replication schemes in Chapters 5 and 6. For that, we consider combining YouTube traces[51] to well-known videos' access patterns [129]. In addition, we introduce in Chapter 6 SLA definitions that take into account Table 6.3 encoding settings. That means that whenever an user is not able to download a watch a video with its minimal bitrate, a violation happens.

[1]The Tube over Time: Characterizing Popularity Growth of YouTube Videos. http://www.vod.dcc.ufmg.br/traces/youtime/data/, January 2013.

[2]Advanced encoding settings for YouTube videos. http://support.google.com/youtube/bin/answer.py?hl=en-GB&answer=1722171, March 2013.

4.4 Content Availability and Peer-to-Peer Protocols

Peer-to-peer (P2P) networks remain one of the major means of distributing content in the Internet. In the past decade, free P2P protocols like eDonkey [62] and BitTorrent [31] became very popular Internet services as they allowed users to freely distribute content throughout the Internet.

Most of the success of content distribution using P2P depends on the content availability which contributes directly to the content popularity growth. Unlike eDonkey, BitTorrent protocol relies on a decentralised swarm-based approach, using direct download link provision [39], and content delivery incentives. This made BitTorrent the most popular P2P protocol these days [103]. For that reason, we focus on describing content availability in BitTorrent protocol.

4.4.1 Strengthening Content Availability Using Peer-to-Peer

Recently, content availability has been an interesting research topic in BitTorrent swarm-based content delivery protocol. We consider that content availability is measured by bandwidth performance, and download time. We describe two features of swarm-based protocols for improving content availability, incentive mechanisms for peers' collaboration, and the content bundling technique.

4.4.1.1 Incentive Mechanisms for Peers' Collaboration

A pure BitTorrent protocol encourages peers to contribute their bandwidth using a collaborative tit-for-tat mechanism. While it provides a straightforward incentives for improving aggregate bandwidth in swarm of peers distributing a content, studies have shown that it is vulnerable to both Sybil attacks [45] and misleading peer's cooperation, as free riding [123, 82]. To address tit-for-tat-based incentive issues and ensure content availability through fairness, Sherman *et al.* [117] introduced an auction-based mechanism that relies on the computation of a deficit counter on the peer basis. The deficit counter represents the number of uploaded bytes minus the number of downloaded bytes. Then, the mechanism prioritizes uploads to peers with lower deficit counter. Similarly, Sirivianos *et al.* [124] proposed a robust incentive mechanism that motivates a peer to collaborate with other peers even if there is not reciprocal content interests. Their enhanced version of BitTorrent, called Dandelion, rewards peers with credits that allow peers to improve content download. It also prevents Sybil attacks by enforcing cryptographic fair exchange of content requests and retrievals.

4.4.1.2 The Content Bundling Technique

To boost content availability in swarm-based peer-to-peer protocols, we may rely also on bundling technique, a common commercial strategy from economics literature [122]. Instead of delivering a single content title, such as a single film or book, bundling content allows content delivery systems to disseminate a collection of correlated contents. There exists two types of bundling.

Pure bundling, where a consumer can purchase the entire bundling or not at all. Alternatively, a mixed bundling permits customers to select parts of a package. Both can be applied to content delivery systems.

Menasche *et al.* [86] argue that the use of pure bundling in BitTorrent protocol improves the overall content availability, particularly for unpopular contents. They highlight though that it can delay those seeking exclusively popular content, resulting in a waste of bandwidth. Mixed bundling is a more common strategy for content providers. It can overcome some drawbacks of pure bundling, and also improve content availability. It permits to distribute contents according to the customers interest, e.g. an entire season of a television series, a music album, or a number of books about a specific subject.

4.4.2 Peer-to-Peer Perspectives on Content Availability

Despite the former success, the popularity of freely available P2P protocols like BitTorrent seems to be in continuous decline. In a report on Internet traffic forecast, Cisco System [58] showed that 62% of the Internet traffic in 2006 came from P2P content dissemination. More recently though, Labovitz *et al.* [77] found through an extensive analysis of inter-domain traffic that P2P Internet traffic has decreased significantly between 2007 and 2009, accounting for only 18% of overall Internet traffic. The decline in P2P Internet traffic is likely related to ISP traffic management, poor direct download link publishing (e.g. torrent files), and copyright issues.

Actually, the traffic from fetching Internet content has shifted from P2P networks to other online services, notably streaming video [9]. Nonetheless, as described in Chapter 3, emerging content delivery architectures [134, 101, 25] have taken advantage of P2P paradigm in order to enhance performance, particularly in terms of scalability and bandwidth provision. In these new architectures, the content provider seems to play a coordinator role, instrumenting the collaboration of peers to meet QoS guarantees. This suggests that peer-to-peer protocols will remain an important building block of content delivery networks for improving content availability. As detailed in Chapter 6, we argue that Internet video services and P2P protocols can successfully be combined to improve CDN performance.

4.5 Summary

Technological advances in computer science and telecommunications, including fast, widespread Internet connections, have contributed to increase the expectations of customers in online content. Today, customers not only want to be able to fetch a content, but they expect to have high download speeds. We assume that content availability is a measure of how fast a Internet content can be fetched, rather than its simple liveliness. Content providers rely on CDNs to distribute their contents to wide audiences with reduced cost and high performance. Content availability is therefore one of the main performance features of services provided through CDNs.

In this chapter, we studied how to improve the content availability through Service Level Agreements (SLA) in CDNs. CDNs provide different levels of content availability depending on

the QoS metric to be enforced. We showed that highly available content can be met through the enforcement of strict SLA contracts, where bitrate is the main Quality of Service (QoS) metric. Networks resource reservation has become the state-of-the-art mechanism for enforcing such QoS metrics inside datacenters. We also studied the availability of Internet videos as a particular type of content. We focus on video-on-demand (VoD) services, particularly YouTube workloads. We provided the main guidelines for understanding the main issues of VoD availability.

We finally show that Peer-to-Peer (P2P) networks may play a important role in improving content availability, specially in edge networks. For instance, BitTorrent protocol can be used for enhancing the aggregate bandwidth, contributing to improving the overall customers satisfaction.

Part II

Content Availability in Edge Networks

Chapter 5

Adapting Replication of a Content According to the Demand

Contents

5.1 Introduction

Content availability has become increasingly important on the Internet in the recent years. The worldwide development of high-speed internet connections has made users less tolerant of high latency and low bitrate when fetching online content. To cope with these issues, content and CDN providers must collaborate to enforce bitrate through SLA contracts. But bitrate enforcement is a QoS metric that is hard to be met, and requires changes in the content delivery infrastructure.

Most of current CDN architectures are still deployed on big, remote and centralized sites, close to the core networks [26], the so-called infrastructure-based architectures. Despite being definitively scalable architectures for content delivery, CDNs based exclusively on datacenters remain huge distributed systems that are very expensive to either build and operate. Their resource allocation efficiency relies mainly on over-provisioning. Since CDN's mechanism are agnostic to edge network resource allocation, their infrastructures are not able to provide bitrate as SLA metric to clients. Thus popular service provision through datacenter approaches makes resources allocation and QoS guarantees being imprecise and unpredictable.

One of the most important new opportunities for Internet operators is to provide distributed storage systems at the edge of the network for cloud and CDN-like users. Such approach will allow system designers to develop web services with outstanding content delivery guarantees that take advantage of very low latencies, high bitrates, and huge amounts of storage capacities. Data replication plays an important role on these new infrastructures. However, it is not an easy task to define replication schemes for edge networks that fairly adapts the placement and the number of replicas for popular content, especially if strict SLA metrics have to be enforced.

Our approach to cope with these issues is two-fold: (i) designing a novel architecture for delivering generic content in edge networks, called Caju, and (ii) providing AREN, an Adaptive Replication scheme for Edge Networks, that allocates network and storage resources to enforce minimum bitrates and prevent SLA violations properly. Thus, we organized this chapter as follows. We describe in Section 5.2 the design of Caju, our simple, general-purpose content distribution system. We present AREN, our adaptive replication scheme, in Section 5.3. We show our evaluations in Section 5.4, and we conclude with Section 5.5.

5.2 Caju: A Novel Design for CDNs in Edge Networks

5.2.1 Overview of Caju

We introduce a simple, general-purpose content distribution system, called Caju. Its design has two main goals. First, it must handle and organize resources from edge networks, allowing interoperability with datacenters. Second, it must provide mechanisms to cope with strict SLA enforcement through adaptive replication schemes.

We assume that Caju operates on sets of devices located close to customers, named federated storage domains, or simply storage domains. A storage domain is a logical entity that combines resources from both datacenters and edge networks in the last mile of the content delivery chain.

For instance, the notion of closeness among nodes of a same storage domain may be specified by a maximum latency or number of hops. As Figure 5.1 shows, content provider disseminates content throughout storage domains. We also consider that any customer is able to share their own contents, behaving as content providers. In a storage domains, devices can play either a coordinator or peer role.

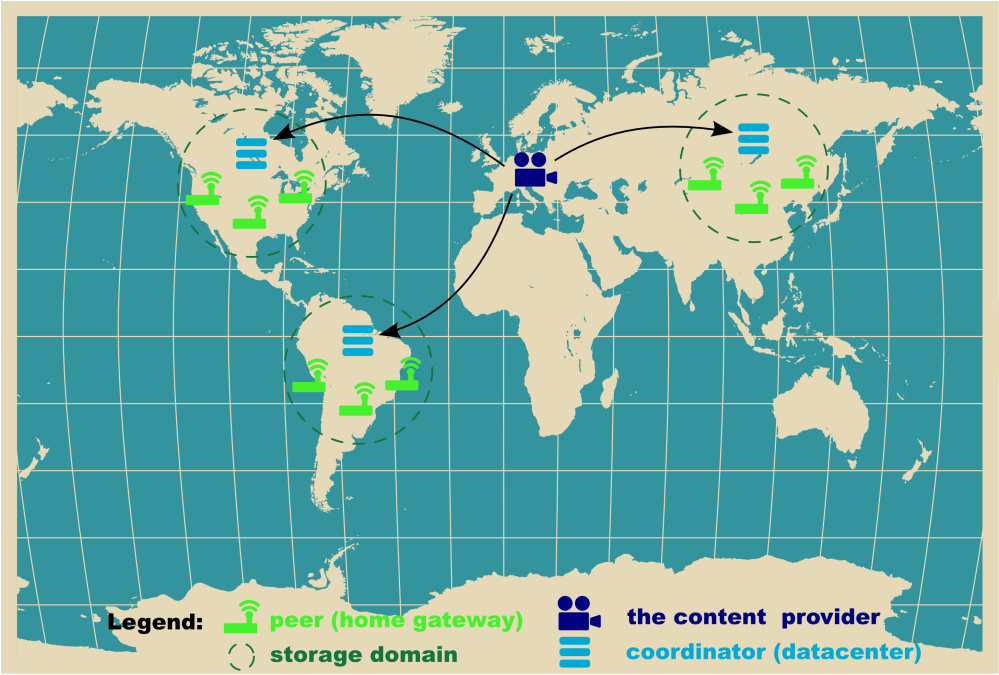

Figure 5.1: Storage domains. Combining Infrastructure-based and edge network resources to enhance content availability.

- **Coordinator** is a server or a small-sized cluster of servers deployed in the nearby datacenter. We assume that coordinator performs scheduling of content requests for the local storage domains. Normally there is a single coordinator per storage domain, which behaves similar to a *Supernode* of P2P systems. Therefore, it coordinates devices resources, keeps information about resources consumption, and inter-operates with other coordinators. Its main goal is to maintain the proper number of replicas per content across the local peers, by pre-fetching or deleting content sources. Instead of always contacting the content providers, coordinators might interoperate in logically centralized way to fetch contents that have been vanished from a storage domain. For instance, they may maintain the entire content catalogue through a distributed hash table. They also store the most recent contents in their own cache for replication purposes. Whenever a new replica is necessary, the coordinator pushes it to a randomly, uniformly selected peer. Similarly, coordinators send deletions commands to local peers. They provide cheap and high available resources dedicated to the content distribution.

- **Peers** is a set of devices located close to each other through which customers get network access, e.g. home gateways or set-top boxes connected to the same digital subscriber line access multiplexer (DSLAM). Peers contribute to storage and network resources according to their availability and load. These devices actually deliver contents to customers in a storage domain, being the main source of storage and network resources. They execute scheduling and replication commands sent by the storage domain's coordinator. Each peer contributes with a percentage of storage to the system, as in a collaborative caching. In the local cache, the Least Recently Used (LRU) policy is applied the for contents replacement.

This design takes advantage of nodes geographical position [19]. It enforces two main infrastructure properties to Caju: replication group and hop limit. The replication group allows Caju to adapt content replication for smaller sets of peers, most likely connecting customers with similar content interests. By enforcing a hop limit in storage domains, we expect to avoid jitter and to ensure low latencies, permitting Caju to improve the efficiency of network resource provision.

On top of each coordinator runs a couple of services that deals with serving customers' or peers' requests, and performs appropriate content placement and replication. The main functional blocks are depicted in Figure 5.2. Remote storage clients contact the coordinator when they need perform any request over a content. Essentially peers may perform three types of content requests: *get*, *put*, or *delete*. The coordinator maintains a catalogue of all peers and available resources, it is also responsible for scheduling the requests. On behalf of the clients, it selects proper resources for fulfilling their SLA contracts based on replications schemes.

5.2.2 The Formal Definition of Caju

Here, we provide further details about our target content distribution system, by formally describing its main services, interactions, and constraints. We consider the problem of defining a hybrid CDN design, whose infrastructure involves datacenter and edge networks resources. Thus, we define the most substantial question of this thesis, how to adapt replication of contents in hybrid CDNs according to SLA contracts.

We focus on two correlated issues: coping with resources allocation hybrid CDNs and enforcing minimum bitrates of SLA contracts using adaptive replication. Consider a set of J storage elements, or simply peers, $\acute{\mathcal{J}}_o$ that stores o. Assume that it is necessary to reserve at least the bandwidth b_o^j on each $j \in \acute{\mathcal{J}}_o$ in order to enforce SLA definitions properly. We turn our attention to the dynamics of adaptive allocation of nodes in $\acute{\mathcal{J}}_o$. We particularly aim to achieve two main goals: (i) improve network and storage usage on edge networks, and (ii) enhance consumers' satisfaction by reducing as much as possible the overall number of SLA violations.

We achieve our goals by focusing on the required number of replicas, as well as the bandwidth allocation on edge nodes. Generally speaking, how to cope with content popularity growth is at the core of the problem. We consider that content location and consistency are beyond of the scope of our thesis. Actually, we believe that the efficiency of a replication scheme in enforcing strict SLA metrics relies primarily on the scheme's capacity of tracking and dealing with con-

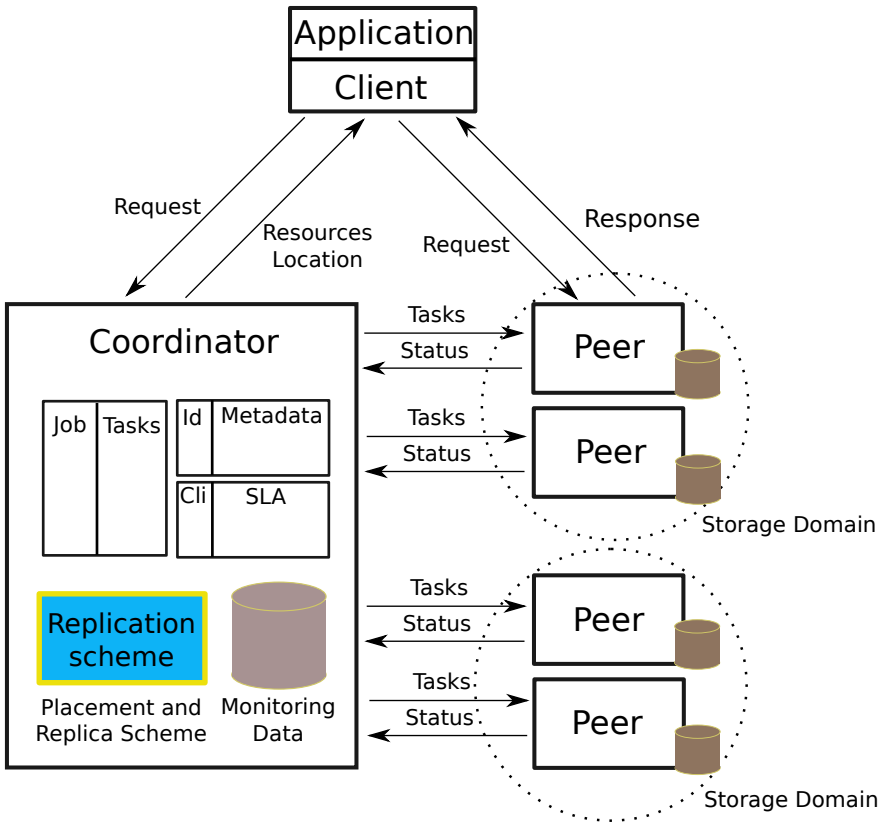

Figure 5.2: The main functional blocks in Caju design.

tent popularity growth properly. By following the content popularity growth properly, we look forward to adapting content replication degree to the its access pattern.

Caju provides a distributed content store and share services for customers. We assume that customers are able to either fetch contents from remote content providers or share their own contents. We denote the set of all possible storage's contents as $\mathcal{O}$. We consider that contents comprise a set of data blocks of fixed size, K_C, called chunks. So that, a content $o \in \mathcal{O}$ of size z_o has $\frac{z_o}{K_C}$ chunks.

We consider that M customers are able to do any number of requests R_M to the system. There are three types of client request: *get* a content, *put* a new content into the system, and *delete* their own contents. Caju provides a fourth system request type in order to *replicate* a content. It might also perform *deletions*, a fifth request type, for maintenance or control purposes.

We assume that clients are eager for quality of delivery service, and their wills are formally defined by SLA contracts. SLAs allow a customer m to choose a suitable bitrate per request λ_s, in bits or chunks, and a minimum acceptable percentage of successful requests P_s. Assuming a period of request analysis T, we consider that a customer m who did r'_m requests is satisfied with the delivery service if at least $r'_m x P_s$ requests were accomplished with λ_s rate. The content delivery service places and replicates contents throughout the system with regard to the customer

satisfaction to prevent SLA violations.

5.2.3 Content Delivery Service

We consider Caju as content delivery system deployed at the edge of network providers, that is organized in storage domains. We assume that there exists I storage domains. A storage domain i, $i \in \{1, 2, \ldots, I\}$ has storage capacity of S_i and aggregate bandwidth T_i. Each storage domain has a set $\mathcal{J}_i$ of J devices, $j \in \{1, 2, \ldots, J\}$, partitioned in two distinct classes: $\mathcal{C}_o$ for coordinator class, and $\mathcal{C}_c$ for peer class, where $|\mathcal{C}_c| \gg |\mathcal{C}_o|$. The storage capacity of a device j is denoted by:

$$s_{ij} = \begin{cases} D_o & \text{if } j \in \mathcal{C}_o; \\ D_c & \text{if } j \in \mathcal{C}_c. \end{cases} \tag{5.1}$$

where D_o and D_c are maximum storage capacity parameters. For instance, edge devices correspond to a set set-top boxes of the same model. Hence,

$$S_i = \sum_{j=1}^{J} s_{ij} \quad i \in \{1, 2, \ldots, I\} \tag{5.2}$$

The device bandwidth capacity is denoted by:

$$b_{ij} = \begin{cases} W_o & \text{if } j \in \mathcal{C}_o; \\ W_c & \text{if } j \in \mathcal{C}_c. \end{cases} \tag{5.3}$$

where W_o and W_c are maximum bandwidth capacity parameters. We assume that any device has a full-duplex, symmetric connection links. Moreover, b^u_{ij} denotes the instantaneous bandwidth consumption of the device j of i. In a same storage domain i, if j and j' are two peers from different classes, and there is not active transfers between them, their respective $b^u_{ij'}$ do not interfere with each other. Despite that, we consider that network infrastructure imposes the following condition (5.4) on the maximum aggregate bandwidth of a set of peers of i:

$$\sum_{j \in \mathcal{C}_c} b^u_{ij} \leq W_l \tag{5.4}$$

where W_l is a the maximum aggregated bandwidth provision for a set of peers that the network provider infrastructure permits. In practice, it would represent the maximum bandwidth capacity of a ISP's Digital Subscriber Line Access Multiplexer (DSLAM). Considering inequality (5.4), the maximum aggregate bandwidth of a storage domain i is denoted as follows:

$$T_i = \frac{1}{K_C}(\sum_j b_{ij}) \leq \frac{1}{K_c}(W_l + |\mathcal{C}_o|W_o) \tag{5.5}$$

where K_C is the chunk size parameter.

5.2.4 System Interactions and Performance Goals

Each customer m is connected, through its own peer device j, to a single domain i, called *home* storage domain. Any m belongs to a SLA class. The system might have one or many SLA classes, such that different levels of quality of service might be provided.

The system allows customers to do requests towards their own homes only. However, all requests might be served by devices from any federated storage domain or associated content provider, except for when customers share their own contents, that is served by customer's home storage domain. We assume that coordinators inter-operates in a logically centralized fashion for offering the following functionalities: (i) mapping edge resources and (ii) monitoring the usage of the content delivery system. The performance and detailed design issues of coordinator are beyond the scope of this thesis.

We denote the set of all R possible requests by $\mathcal{R}$. Requests are grouped in two distinct manners: by requester or by type of request. In terms of requester, there are two disjoint subsets: $\mathcal{R}_M$ for customer's requests, and $\mathcal{R}_S$ for own content delivery system's requests. When our system receives a request r that requires to transfer contents between any peers of i, it serves this request by creating data transfers from a source to a destination. As described above, a requester might come either from a customer or the own system. If $r \in \mathcal{R}_S$ the transfer is always made between two devices. For all $r \in \mathcal{R}$, let:

$$p_{j,r} = \begin{cases} 1, & \text{if } j \text{ serves } r; \\ 0, & \text{otherwise.} \end{cases} \tag{5.6}$$

be a 0-1 variable indicating if the devices $j \in \mathcal{J}_i$ provides resources to serve request r. We assume there is a function $A^i_j(t)$ that yields the current available bitrate by j in t for serving a system incoming request. Therefore, if client m requests r^m over a storage domain i in time t, asking for a λ^s_m rate, the storage system fulfil m's expectations if and only if:

$$\textbf{Constraint 1:} \sum_{j \in \mathcal{J}_i} p^m_{j,r} \cdot A^i_j(t) \geq \lambda_s \tag{5.7}$$

Therefore our system performance goal for our replication scheme is twofold. Firstly, we aim to provide the maximum number of satisfied clients as possible, reducing as much as possible the number of SLA violations. Secondly, we intend to improve the network provision reducing as much as possible system's storage usage, by adjusting properly the resource allocation over peers in order to serve $\mathcal{R}$.

5.2.5 Generic Mechanisms for Coping with Popular Content in Edge Networks

Our approach strongly relies on content placement and replication to achieve performance goals. In this section, we present an intuitive formal description of placement and replication mechanisms used in this thesis, and how they are related to our performance goals. We introduce a

generic concept of popularity for online contents, and we show some insights into its impact on placement and replication definitions. Then we present our generic mechanisms for placement and replication where content popularity is a key input parameter. We highlight that the design of our replication schemes aim to enhance content delivery system performance by providing better content availability, as described on Subsection 2.2.2. In order to simplify the model of our schemes, we consider a failure-free environment, where nodes are always available. Although fault tolerance is out of the scope of our thesis, our model is flexible enough to be combined with fault-tolerant techniques.

5.2.5.1 Popular content

We assume that a content $o \in \mathcal{O}$ has a lifetime on the storage system. During its lifetime, its popularity might evolve depending on the access patterns. We consider the content popularity as a dynamic phenomenon which is continuously measured by the coordinator. We denote $\Pi_o(t)$ as instantaneous popularity of content o during time t. All begins when o is published by a content provider or shared by a client m who performs a *put* request. We assume that, at initial time of a content's lifetime, its popularity $\Pi_o(0)$ is equal to zero. Then any client might do *gets* to this new content, modifying its popularity until its deletion.

5.2.5.2 Initial Content Placement

For all $j \in \mathcal{J}_i$, we have the following variable:

$$p^i_{o,j} = \begin{cases} 1, & \text{if } o \text{ is placed in } j \text{ on } i; \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

that indicates if a content $o \in \mathcal{O}$ is stored in device j of storage domain i. We refer to the vector $\boldsymbol{p}_o = [p^i_{o,j}]_{j \in \mathcal{J}_i, i \in \{1,2,\ldots,I\}}$ as a placement for content o, and we denote $\hat{\boldsymbol{p}}_o$ as the initial placement in particular.

We define initial placement as a result of a set of F transfers, $f \in 1, 2, \ldots, F$ required for a *put* request, where f transfers an entire content between two devices. This placement vector becomes a $\boldsymbol{p}_o$ if any subsequent transfer $(F + 1)$ modifies the initial placement vector $\hat{\boldsymbol{p}}_o$. For example, it happens when additional replicas are created or deleted. Since we study the availability of entire contents rather its chunks, we consider that a content is available on a peer to be downloaded by customers only when that peer has the entire content copy.

5.2.5.3 Keeping Track of Content Popularity Through Replication

We define r_o as the replication degree, or number of replicas, of content o in the system. Since content's popularity changes dynamically during its lifetime, we consider that our replication strategy must be adaptive in order to provide quality of service properly. Hence, an adaptive approach of defining r_o must be a function of $\Pi_o(t)$.

5.2.5.4 Replication Mechanisms for Delivering Popular Content

In this thesis, we assume two replica maintenance mechanisms for popular content: bandwidth-aware (similar to Subsection 4.2.2 definitions) based on thresholds and caching approach (as in Section 2.3 related work). These mechanisms basically define changes on the initial placement and how to create new replicas, in order to deliver storage service for popular content according to SLA constraints.

Bandwidth-aware based on thresholds. The placement and the number of replicas in this mechanism are adapted according to the available bandwidth usage of peers. The initial placement is composed of a fixed chain of transfers $\mathcal{F}_o = \{1, 2, \ldots, F\}$, where $F = r_o + 1$, and r_o is equal to a constant value R. Among these F transfers, only the first transfer has to satisfy Constraint 5.7. The remaining $F - 1$, or R, transfers create content's replicas. In this approach, peers are pseudo-randomly selected for for the initial transfer considering Constraint 5.7, and randomly for remaining transfers.

In order to adapt the number of replicas for new incoming *get* request and to cope with content popularity in our replication scheme properly, the number of replicas increases if:

$$\sum_i \sum_{j \in \mathcal{J}_i} p^i_{o,j} \cdot A^i_j(t) < \frac{\omega}{K_C} \left(\sum_i \sum_{j \in \mathcal{J}_i} p^i_{o,j} \cdot b_{ij} \right) \tag{5.9}$$

where ω is a constant and the right side of Eq. 5.9 defines our increase threshold. The placement of these new replicas must be made in order to increase the left part of Eq. 5.9. Our system decreases the number of replicas if:

$$\sum_i \sum_{j \in \mathcal{J}_i} p^i_{o,j} \cdot A^i_j(t) \geq \frac{\theta}{K_C} \left(\sum_i \sum_{j \in \mathcal{J}_i} p^i_{o,j} \cdot b_{ij} \right) \tag{5.10}$$

where θ is a constant and the right side of Eq. 5.10 defines our decrease threshold. Similar to the increase procedure, the deletion of replicas must be made in order to decrease the left part of Eq. 5.9.

Caching. We use caching as a basic and fairly efficient mechanism to adapt replication according to content popularity. A cache is the simplest way to save bandwidth, reduce latencies between popular sources and clients, and particularly avoid server "hot spots" by distributing the load. We assume that for each device j there exists a portion of storage capacity γs_{ij} statically allocated for caching, and the replica replacement follows LRU algorithm. Yet, considering a set of caching contents, our evaluation scheme is able to easily select a device that satisfies Constraint 1 (Eq. 5.7) properly. Table 5.1 presents a summary of our notations.

M	Number of customers.
R_M	Number of customers' requests.
$\mathcal{O}$	Set of contents.
z_o	Size of content o.
K_C	Constant chunk size.
λ_s	Suitable rate of bit or chunks per request, a SLA parameter.
P_s	Minimum acceptable percentage of successful requests, a SLA parameter.
i	Storage domain i, $i \in \{1, 2, \ldots, I\}$.
$\mathcal{J}$	Set of devices.
$\mathcal{C}_l$,$\mathcal{C}_\rfloor$	Devices classes: coordinator (C_o) and peer (C_c).
$\mathcal{D}_l$,$\mathcal{D}_\rfloor$	Storage capacities parameters.
$\mathcal{W}_l$,$\mathcal{W}_\rfloor$	Network capacity capacities parameters.
$\mathcal{W}_\updownarrow$	Maximum aggregated bandwidth consumption for a set of peer elements of a single storage domain i.
s_{ij}	Storage capacity of device j of storage domain i.
b_{ij}	Network capacity of j of i.
S_i	Maximum storage capacity of a storage domain i.
T_i	Maximum throughput of chunks or bits per second of a storage domain i.
$A_j^i(t)$	Current available rate of chunks from j of a storage domain i at time t.
$\Pi_o(t)$	Current popularity function of o at time t.
r_o	Number of replicas of content o.
$\boldsymbol{p}_o$	Placement mapping or vector of content o.
$\hat{\boldsymbol{p}}_o$	Initial placement vector of o.
$\mathcal{F}_o$	Chain of transfers of a request on content o.
R	Constant number of initial copies for an initial placement of o.
ω	Constant of proportionality for decreasing the number of replicas.
θ	Constant of proportionality for for decreasing the number of replicas.
γ	Constant of proportionality for cache memory size.

Table 5.1: Summary of notations

5.3 AREN: An Efficient Replication Strategy for Enhancing Content Availability

This section describes the main contribution of this thesis, our adaptive replication scheme AREN. Assuming Caju as hybrid CDN architecture, it successfully combines bandwidth-aware based on thresholds and caching mechanisms, described on Subsection 5.2.5.4 for coping with replication of contents according to their popularity growth, preventing SLA violations, and enhancing content availability.

5.3.1 Overview of AREN

AREN stands for Adaptive Replication for Edge Networks scheme. AREN replication scheme relies on bandwidth reservation and collaborative caching to provide an adaptive number of replicas for popular content. AREN provides request scheduling and content replication mechanisms to minimize strict SLA violations and edge resources usage.

5.3.2 Scheduling Network Resources through Replication for Improving Content Availability

The main concern of this thesis is to provide efficient resource allocation for meeting SLA-based customers expectations. In our case, we perform content replication for meeting this goal. Assuming that customers expect highly available contents, we are mostly interested in designing a replication scheme that allows us to ensure bitrate as main QoS metric. We assume that this QoS metric is hard to enforce, so that we talk about strict SLA contracts enforcement as a key challenge of this research. Recently, researchers have extensively studied strict SLA enforcement in datacenters networks, as detailed in Subsection 4.2.2. D3 [135, 141, 5] are remarkable example of this approach. It provides a deadline-aware control protocol that uses explicit rate control to apportion bandwidth according to flow deadline. That provides the ability to increase the aggregate throughput in datacenter environments compared to TCP. In AREN, we investigate how useful this mechanism is for instrumenting replication. Basically, we intend to keep track of network resources reservation as the main feature to adapt the number of replicas of a content.

5.3.3 Request Scheduling

Within a storage domain, AREN relies on the coordinator to track bandwidth reservation and to select nodes accordingly. Sources are selected to respond a request only if there is enough unreserved bandwidth. Scheduled sources contribute with the same amount of bandwidth, and cooperate to enforce SLAs by reserving bandwidth.

To enhance resource allocation in edge networks, AREN implements two simple scheduling policies.

- **Divide-and-conquer.** *get* requests are served by either peers or coordinator server or small-sized cluster. The *divide-and-conquer* scheduling policy gives priority to peers and uses coordinator only if there is no more spare bandwidth for reservation in the set of peers of the requested content. It permits to save mini-datacenter bandwidth for creating replicas to popular content faster.

- **Nearest source selection.** We assume that intra-domain transfers are preferable. For that reason, this scheduling policy prioritizes the selection of *get* sources that comes from the same Storage Domain of the request destination. That allows AREN to reduce the inter-domain traffic load.

5.3.4 Content Replication Strategy

After scheduling a request to an content, the coordinator updates its current aggregate bandwidth demand, and decides if it is worth creating a new replica in caching of the destination node. The coordinator computes the utility of a new replica based on thresholds. Replica utility measures the benefit of creating replicas according to estimated popularity and current bandwidth consumption of a content. We consider two thresholds for aggregate reserved bandwidth: P_{min} and P_{max}. Our replication strategy is based on two main mechanisms, namely *popular replica classification* and *replica maintenance for popular content.*

5.3.4.1 Popular Content Classification

We consider a content as popular whenever its aggregate active bandwidth is greater than a factor of the high threshold. For instance, consider a popularity factor Q, the threshold percentage P_{max}, and content o that has a single replica into a peer of network capacity of b. Let $U(o)$ be the current bandwidth reservation for content o, o is popular if $U(o) > Q * P_{max} * b$.

5.3.4.2 Replica Maintenance for Popular Content

This mechanism adapts the number of copies of popular contents regarding the thresholds and the current aggregate reserved bandwidth. We actually replicate contents according to aggregate network usage by enforcing a low and high thresholds. This makes the content replication a function of bandwidth reservation, and ensures that network and storage provision follows content demand properly, as depicted in Figure 5.3. It is performed whenever a *get* is scheduled or periodically for maintenance purposes. New replicas are created in the *get* destination when aggregate bandwidth is greater than the high threshold, and randomly removed when smaller than the low threshold. We highlight the procedure CheckReplicas, that runs on coordinators, is depicted in Algorithm 1.

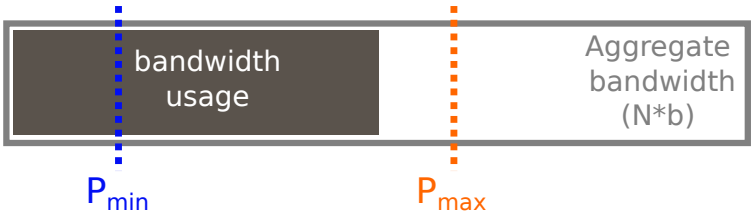

Figure 5.3: AREN bandwidth management for a popular content, illustrating aggregate bandwidth for N replicas and b available bandwidth, bandwidth reservation (bandwidth usage) and thresholds (P_{min} and P_{max}).

Algorithm 1: CheckReplicas verifies the proper number of replicas for popular content.

Input: Target content o, high threshold percentage P_{max}, and low threshold percentage P_{min}

Output: Quick-fix number of replicas $\in \{-1, 0, 1\}$.

```
1  b ← 0;
2  r ← 0;
3  for j ∈ Ĵ_o do // peers with o
4  |   b ← b + GetBandwidth(j);
   |   // get reserved bw for o on j
5  |   r ← r + GetReservedBandwidth(o, j);
6  if r > b * P_max then
7  |   return 1;
8  else if r < b * P_min then
9  |   return −1;
10 else
11 |   return 0;
```

5.4 Evaluation

In this section, we evaluate the performance of Caju and AREN in providing high content availability for content distribution networks. As described in Section 5.2, content availability is defined through bitrate in strict SLA contracts, and fault tolerance is out of the scope of our thesis. Firstly, we show how Caju enhances the performance of content distribution in edge networks. Then we explain how AREN provides high content availability using edge devices. The rest of this section is structured as follows. Subsection 5.4.1 highlights our performance goals. We present our evaluation scenario and workload in Section 5.4.2. Section 5.4.3 describes the four other replication schemes, besides AREN, evaluated in this thesis. Subsection 5.4.4 shows how efficient straightforward replication schemes are to provide highly available content in edge networks. Then, Subsection 5.4.5 shows how our threshold-based replication approach copes with challenges related to content availability faced by ISPs in edge networks.

5.4.1 Performance Goals

We assume that the primary goal of edge networks is providing Internet access to customers. Content delivery in edge networks regarding SLA contracts may represent a costly service for ISPs. Our evaluation has two main goals. (i). To verify if it is reasonable to use edge devices, including peers of customers, such as home gateways, to deliver highly available content by enforcing strict QoS metrics. (ii). To evaluate the performance of AREN, our threshold-based approach as an adaptive replication scheme. We focus on three basic evaluation metrics: number

of SLA violations, network and storage usage. We assume that a SLA violation occurs when any transfer of a consumer does not observe her minimum contracted bitrate, undermining the availability of the content for a customer, thus failing to meet her expectations. We use *happiness* or number of customers without SLA violations as a performance metric. Network and storage are also very important in our evaluations. They measure how efficient our approach is with regard to edge network resources. In terms of network resources, we analyse the number of flows and aggregate bandwidth where the higher is the better. For storage usage, we seek the exact opposite. By avoiding creating unnecessary replicas, including rarely accessed contents, we expect to reduce the burden of replication in edge networks as much as possible.

5.4.2 Evaluation Scenario

The evaluation scenario (Figure 5.4) includes 4002 nodes, arranged across two storage domains. There are one coordinator and 2000 peers per storage domain. Storage and network capacities differ accordingly to the class of device. Coordinators have 20TB of storage capacity and full-duplex access link of 4Gbps. Peers contribute 200GB each, equipped with 100Mbps full-duplex links (based on recent home gateways commercial offers [54]). Note that the two coordinators contribute to a small fraction of the total amount of overall edge resources, namely additional 5% of storage capacity and only 2% of the overall network capacity. This draws our attention to the performance of replication schemes and resources allocations towards non-expensive small-sized clusters of servers reserved for content delivery in datacenters. We also assume that peers of a storage domain are connected to the same edge network, where a maximum limit of 80% is enforced to aggregate traffic, as detailed in Subsection 5.2.4. This means that inter-domain traffic is more expensive than intra-domain traffic, particularly if you consider that there exists high latencies between any two storage domains. Edge networks are connected to the internet through the operator network that ensures inter-storage domain connectivity.

Workload was carefully set-up to match to multimedia popular content distribution, as described in recent studies presented in Chapter 4. Based on recent findings [129], we model the distribution of popularity of Internet videos as a Zipf-like distribution. Similarly, as our system deals with multimedia contents, we assume that the distribution of content sizes follows the Pareto distribution, ranging from 3 to 1600MB. Considering the work of Figueiredo *et al.* [51], we model the distribution of *gets* or popularity growth for each content using the Weilbull distribution.

Tables 5.2 and 5.3 list default values for the evaluation scenario and workload parameters respectively. SLA definitions, used throughout our evaluation, are detailed in Table 5.4. Contents are always divided in chunks of fixed size, 2MB. SLA contracts differ from each other by bitrate. Thus, we consider three SLA classes, in chunks per second: (a) 41, (b) 21, and (c) 14 chunks/s. Each customer has a SLA according to the following distribution: 40% class (a), 40% (b), and the remaining 20% (c).

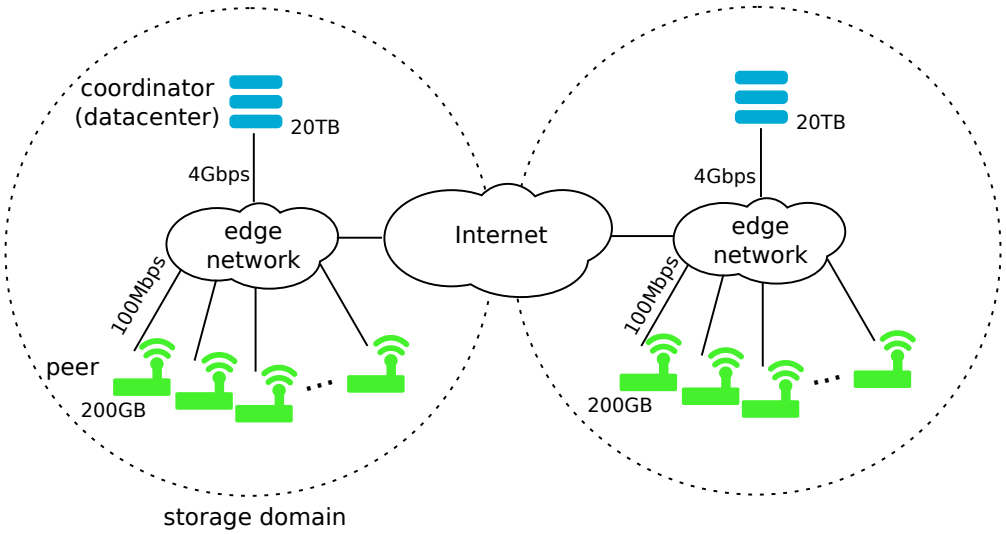

Figure 5.4: Evaluation scenario

5.4.3 Evaluated Replication Schemes

In this section, we compare the performance of AREN with four other replication schemes, detailed as follows.

Uniform replication scheme with fixed number of replicas. This is the simplest approach to replicate contents into a content delivery system, that is broadly used in current datacenter deployments, e.g. Google File System (GFS) [55] and Ceph [140]. Given a fixed number of replicas n as a parameter, we simulate a chain of content-replication of n stages just after the initial insertion (*put*). Considering the default settings of Caju's design described in Subsection 5.2.2, *get*s are randomly scheduled to balance the system load. Each request is served by at most R nodes with equal load. The actual number of sources is $r = min(n, R)$.

Non-collaborative LRU caching. Simple adaptive replication schemes based on non-collaborative caching, such as those that implements Least Recent Used algorithm, are easy to implement and deploy, and cope with popularity growth much better than uniform replication schemes. In our evaluation, a new replica is created whenever a customer, connected to a peer, performs a *get* to any content. LRU replacement is enforced regarding a static percentage of the local storage capacity for caching. Request scheduling uses Caju's default settings. Initial placement requires two replicas ($n = 2$) in different device classes of the same storage domain. Therefore, whenever we mention simply *caching* replication scheme in this thesis, we are referring to non-collaborative LRU caching.

DAR. The main goal of Distributed and Adaptive Replication (DAR) scheme, proposed by Zhou *et al.* [144], is to balance the expected bandwidth load per node. DAR algorithm intuition is replacing content replicas in the local caches based on their current bitrate. Fresh contents replace local cached contents with higher bitrate, removing the highest first. In their work, Zhou *et al.* assume that there is a logically centralized coordinator that tracks and computes the latest bitrate

Table 5.2: Default values for the evaluation scheme parameters.

Evaluation scenario	
Number of Storage Domains	2
Number of Coordinators per Storage Domain	1
Number of Peers per Storage Domain	2000
Coordinator Storage Capacity	20TB
Peer Storage Capacity	100GB
Coordinator Network Capacity	4Gbps
Peer Network Capacity	100Mbps
Aggregate Bandwidth Limit for a Set of Peers	80%
Chunk size	2MB
Number replicas	2
Maximum parallel flows per request	5

Table 5.3: Default values for workload parameters.

Workload	
Requests per Customer	uniform
Experiment Duration	1h 40min
Mean Requests per Second	100
Request Types	5% for *puts* 95% for *gets*
Content Size (follows Pareto)	shape=3 lower bound=26MB upper bound=1.6GB
Content Popularity (Zipf-Mandelbrot)	shape=0.8 cutoff=# of objects
puts (Poisson)	λ=*puts*/s
Popularity Growth (follows Weibull)	shape=2 scale $\propto$ duration

Id	QoS (chunks/s)	Customers Distribution(%)
0	41	40
1	21	40
2	14	20

Table 5.4: SLA definitions

of any content. Since this approach was initially proposed for a P2P architecture and did not handle directly strict SLA targets, we had to sightly enhance our implementation as follows. If no content with higher bitrate was found, but there exists stale contents, apply LRU as replacement policy. For DAR, we also use Caju's default settings for request scheduling and initial placement.

Unlimited. In this case, we have made an assumption of unlimited network and storage capacities at both consumer and operator edge nodes. Source nodes reserve the strict bandwidth necessary to a transfer according to the SLA contracts. It differs from our AREN approach in two points. First, it ignores spare bandwidth, keeping bandwidth reservation value as a hard limit. Second, it avoids creating additional replicas since nodes always have enough resources. We consider unlimited assumption as a benchmark case. It represents the empirical measure of the optimal bandwidth usage when bitrate is enforced as QoS metric.

5.4.4 Performing highly available content delivery

We initially measure the feasibility of delivering popular content with strict SLA contracts using Caju. We compare two approaches: uniform replication with a fixed number of replicas, and non-collaborative LRU caching.

We have evaluated the required number of replicas using the uniform replication scheme for different request rates in order to prevent SLA violations. We consider replicas stored on peers only. We consider that the mean size of contents is 40MB, and we varied the number of replicas from 1 to 10. Figure 5.5 shows the *happiness* metric, described in Subsection 5.4.1, for mean request rates of 100 (default value in Table 5.3), 200, 300, and 400 requests per second. We have observed that uniform replication schemes require high replication degree in order to cope with strict SLA definitions and popular content. At least seven replicas are required to prevent violations if the request rate is as high as 200 requests per second. For higher request rates, uniform replica is not suitable. When our evaluation scheme simulates 300 and 400 requests per second, there were 385 and 2953 violations respectively. Despite having being widely used in datacenters and storage clusters, uniform replication scheme rely on over-provision in order to distribute popular content with strict SLA definitions, hence it is not fit for edge network deployments. Therefore, we assume 400 requests per second as default rate.

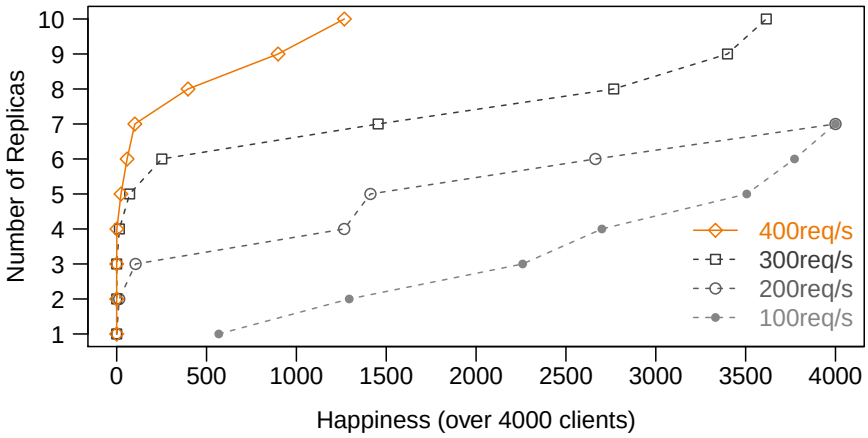

Figure 5.5: Happiness with uniform replication scheme

To avoid over-provision on edge networks, and cope better with content popularity growth, we have compared uniform replication with a non-collaborative LRU caching. We varied the number of replicas of the uniform approach from 1 to 10. For non-collaborative LRU caching, we simulated different caching sizes percentages: 1%, 5%, and 10% of the peer's storage capacity. Figure 5.6 plots an initial evaluation of storage usage and *happiness* for these two replication schemes. Even with the smallest cache percentage of 1%, caching performs much better than uniform replication. Caching consistently improves *happiness* metric by preventing violations. It allowed us to slash SLA violations from 2953, with uniform scheme, to 1. It required only

14.20TB, that is similar to a uniform scheme with 3 replicas, 12.98TB. Hence, we keep 1% of the peer's storage capacity as the default evaluation scenario setting for caching replicas.

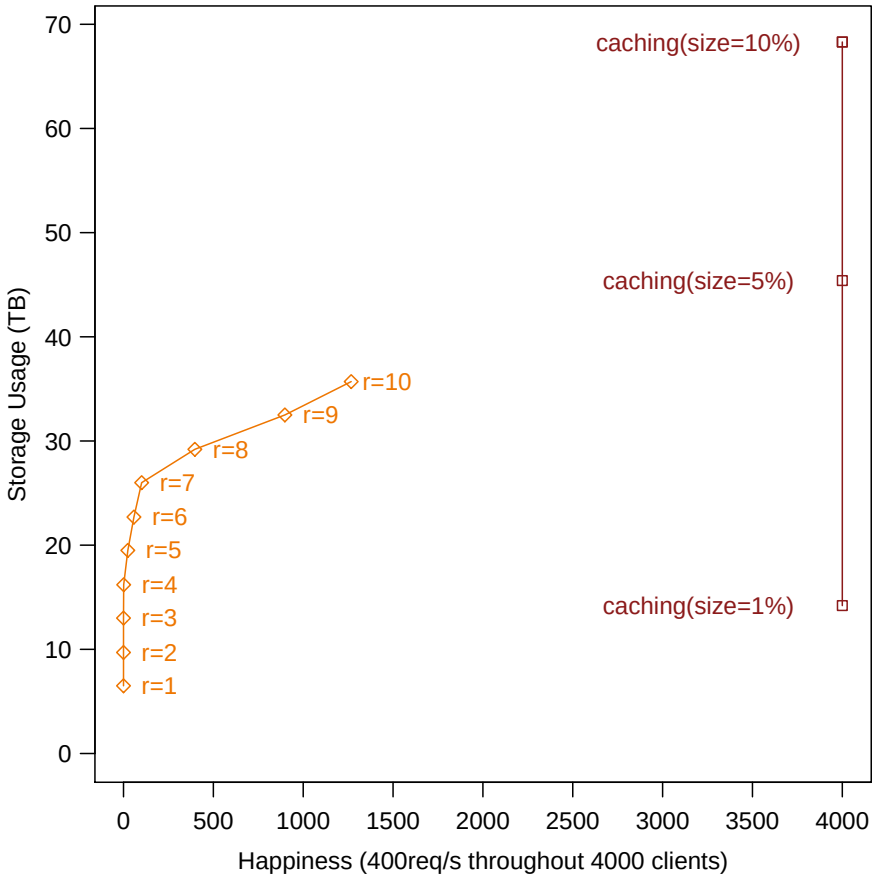

Figure 5.6: Happiness and storage usage with uniform replication scheme (r) and non-collaborative LRU caching.

In order to gather more information about the advantages of using non-collaborative caching for delivering highly available content instead of uniform replication, we evaluated and plotted in Figure 5.7 the aggregate bandwidth (throughput), flows, and violations per second. We selected results from LRU caching with local cache size of 1% of the peer's storage capacity, and uniform replication with 10 replicas. By using a non-collaborative LRU caching, we observed that the number of flows and aggregate throughput were reduced by half. We have also verified that the number of violation slashed from 2953 to 1.

These results show that (i) simple caching is much more efficient in replicating popular content on edge networks than uniform approach in terms of number of SLA violations, (ii) caching allows us to reduce network resources consumption, and (iii) it permits edge node to contribute with tiny amounts of storage capacity contribution (1% of peer's storage capacity, 2GB) in order to maintain enough replicas for improving content availability.

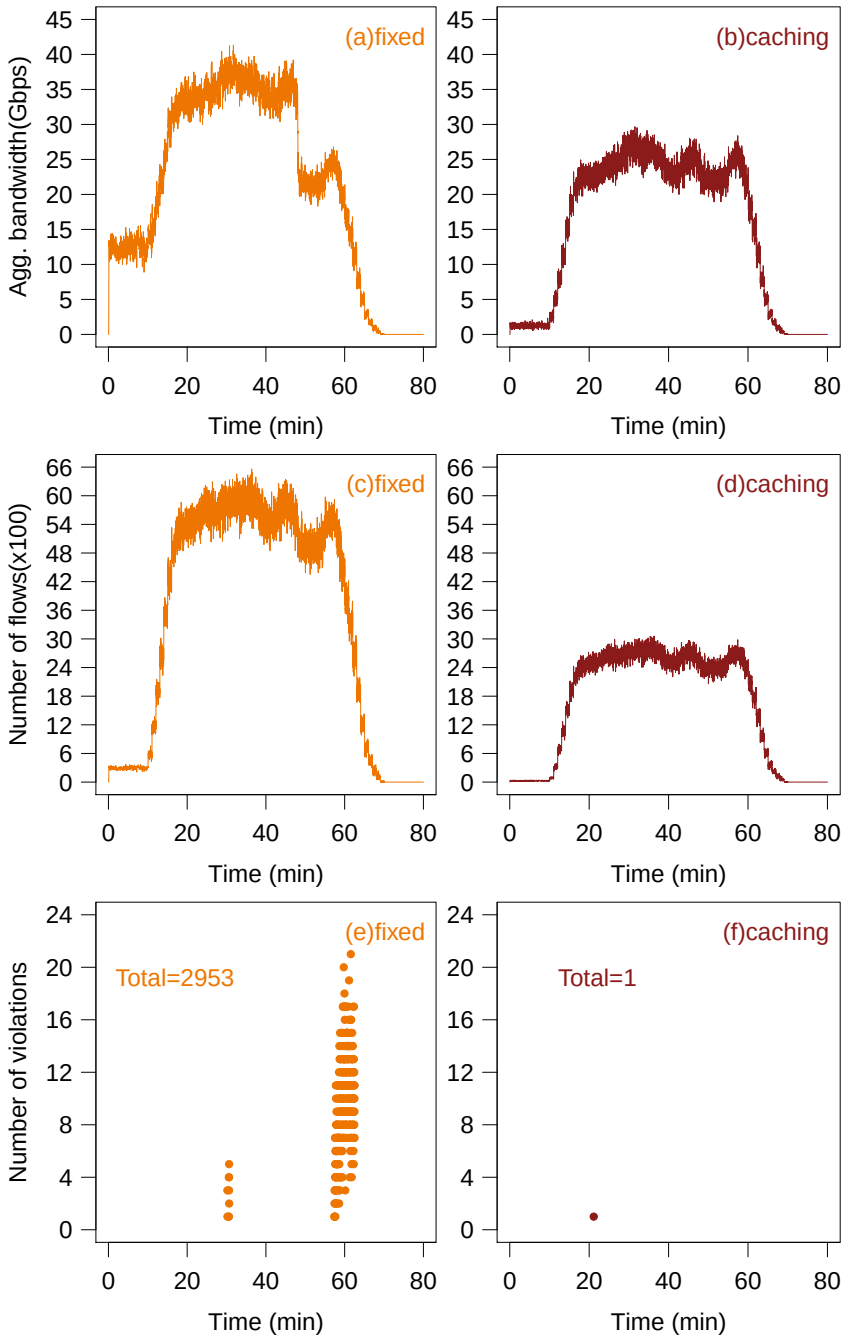

Figure 5.7: Aggregate bandwidth (a,b), number of flows (c,d), number of SLA violations (e,f) using uniform replication scheme (fixed r=10) and LRU caching (size=1%).

5.4.5 Exploring popular content delivery with AREN

Our two main aims in delivering content in edge networks are to prevent the number of SLA violations, and to improve resources utilization. We have shown that a non-collaborative LRU

caching copes with these issues quite fairly compared to a uniform replication scheme. But an increasing demand for multimedia content, especially as VoD, might overload content delivery systems, damaging its performance. We assume that replication schemes in edge networks should be able to adapt accordingly. Here, we present challenges raised by heavily loaded content delivery systems, and we show how AREN uses *collaborative caching* and bandwidth reservation to overcomes these issues.

5.4.5.1 Preventing SLA Violations under Heavy Load

We compare AREN to non-collaborative LRU caching, and a collaborative caching based on DAR replication algorithm, all described in Subsection 5.4.3. All the schemes were set to use a cache size equal to 1% of the local storage capacity and customers are able to *get* contents by setting up transfers from up to five different sources, according to number of available replicas, i.e. $min(5, thenumberofallavailablereplicas)$. Chunks size also remains unchanged, 2MB. According to the definitions of Section 5.3, we have set up AREN to enforce bandwidth reservation, and we chose the minimum threshold percentage to 5% and the maximum one to 30%. To simulate workload with higher loads, we slightly modified the default values of content size distribution from Table 5.3, by changing the shape and lower bound (the smallest content size) parameter of the Pareto distribution.

We initially show, in Figure 5.8, the *happiness* measurements when the mean content size increases. DAR and LRU caching approaches perform poorly in higher loads, while AREN is resilient to load increases. Overall, *happiness* falls sharply when the workload's mean content size increases, except for AREN replication scheme. Under the heaviest load, mean content size equal to 140MB, we observed *happiness* metric equal to 3949 for AREN, versus 2 for caching and 1 for DAR. While AREN suffered only 51 violations, caching and DAR suffered 27539 and 30071 respectively. Compared to a non-collaborative caching, AREN prevented about 99.8% of all violations. For the remaining evaluations, we assume 140MB as the default mean content size.

To shed some light on the SLA violation problem, we have analysed the *get* durations of non-collaborative LRU caching that violated SLA constraints. We present ECDF of these *get* durations in Figure 5.9. 99% of all 27539 violations last at least 4.64 seconds, with outliers up to 48 minutes. Since simple non-collaborative caching relies on random scheduling for delivering content, it lacks essential information for preventing edge nodes' overloading. It is directly related to the concurrency increasing due to transfers that last longer and, particularly for the most popular content, they are fetched from a high number of sources. AREN outperforms both caching and DAR by adapting the number of replicas according to the demand. While DAR and caching waste their caches storing unnecessary replicas, AREN keeps the minimum number of replicas for meeting SLA contracts by operating a collaborative caching. This reduces the number of concurrent uploads on source peers, prevents violations, and allows us to improve storage and network usage.

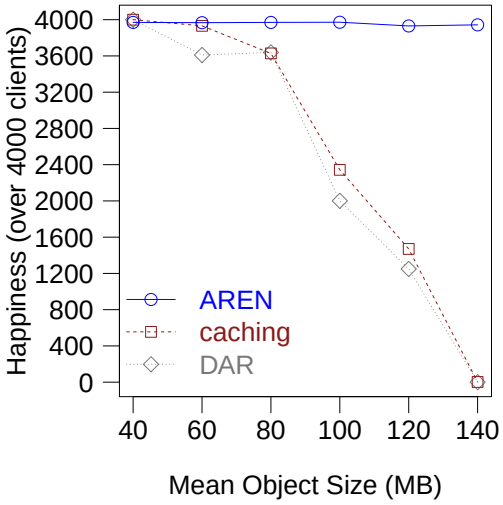

Figure 5.8: Comparing *happiness* metric for higher loads using caching, DAR and AREN.

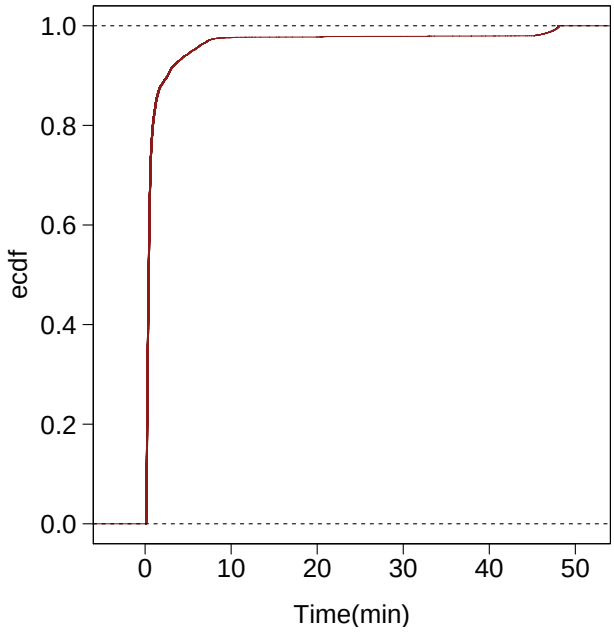

Figure 5.9: ECDF of failed *get* durations using caching. The analysis of all 27539 *get* violations shown that 99% last from 4.64 seconds to 48 minutes.

5.4.5.2 Reducing Storage Usage

We consider that peers' primary goal is not to provide storage for content delivery. Therefore, storage usage in peers has to be reduced as much as possible. Considering our heaviest loaded workload with a mean object size of 140MB, we plotted overall storage usage for DAR, LRU

caching and AREN in Figure 5.10. It suggests AREN reduce roughly 30% of storage usage by instrumenting a cooperative caching properly.

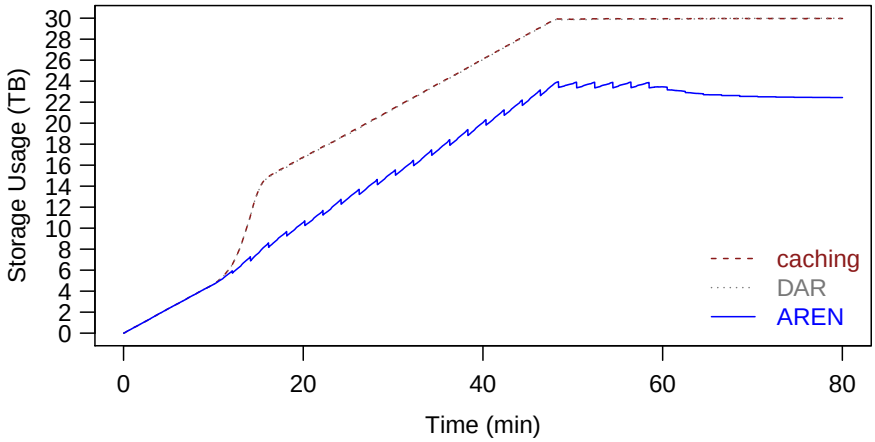

Figure 5.10: Overall storage usage for caching (non-collaborative LRU caching), DAR , and AREN.

Since all schemes of this Subsection perform the same initial placement, storage usage differs exclusively in cache usage for replication. Figure 5.11 shows the storage usage by replicas. AREN scheme provides roughly a sevenfold decrease in the amount of cache usage compared to DAR and LRU caching approaches. AREN performs much better than both DAR and LRU caching approaches because it creates new replicas for popular content only, and it is able to remove unnecessary replicas thanks to the AREN's coordinator role in monitoring and tracking aggregate bandwidth reservation. DAR and LRU caching have similar results due to a surprising performance of our DAR implementation for delivering popular content, detailed in Subsection 5.4.3. Our enhanced DAR implementation reduces SLA violations, but increases the system's storage usage.

5.4.5.3 Improving Network Provision in Storage Domains

As coordinators are located in nearby datacenters, they provide poorer network resources than edge network's peers in order to enforce high content availability. For instance, the latency among peers is minimal, and the bandwidth available for a content can be easily adapted by varying the number of peers with replicas. An efficient replication scheme must prioritise as much as possible bandwidth allocation in peers, ensuring the lowest latency and preventing jitter. In AREN, we assume that coordinators play an important role in maintaining replicas for content durability rather than its availability. Thus, we consider that the reduction of network usage in coordinators is the best scheduling approach towards better resource allocation. We present the aggregate bandwidth usage for non-collaborative LRU caching, DAR and AREN schemes in Figures 5.12, 5.13, and 5.14, respectively. AREN scheme reduces roughly 50% of aggregate upload bandwidth. AREN performs better thanks to the *divide-and-conquer* policy, described in 5.3.3, that prioritizes peers as *get* requests' sources. Instead, non-collaborative LRU caching and

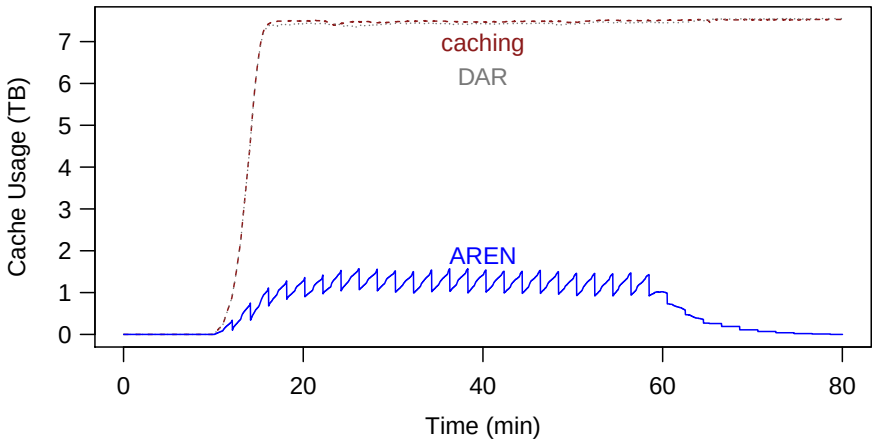

Figure 5.11: Replication's footprint. Investigating the overall caching usage for replicas amongst caching (non-collaborative LRU caching), DAR, and AREN, all using up to 1% of peers' storage capacity for caching.

DAR schemes rely on pure random scheduling that overloads coordinators upload link, providing poorer network provision. Aggregate download bandwidth has the same level for all two schemes because a common initial placement policy requires the primary copy to be stored in a coordinator. With DAR approach, we have found results quite similar to caching.

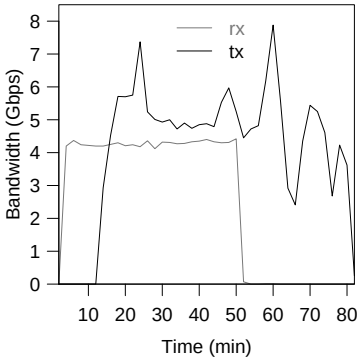

Figure 5.12: LRU caching .

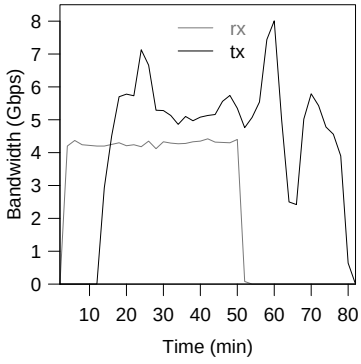

Figure 5.13: DAR.

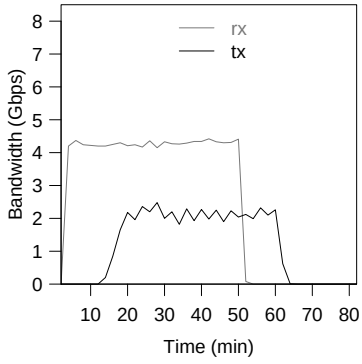

Figure 5.14: AREN.

Bandwidth in peers must be allocated efficiently. The request scheduling must handle requests properly in order to reduce the traffic burden and enhance network usage, particularly for the most popular contents. Figure 5.15 shows a box plot, including minimum value, first quartile, median, third quartile, whisker and outliers values, from upper quantile of *get* durations of the 10 most popular contents. These contents account for 1.5% of nearly four million *get* requests. AREN presents the smallest degree of dispersion amongst the evaluated replication schemes. That happens because AREN scheme is able to better schedule *get* requests through bandwidth reservation, avoiding that *get* requests for popular content either last too long, causing violations, or

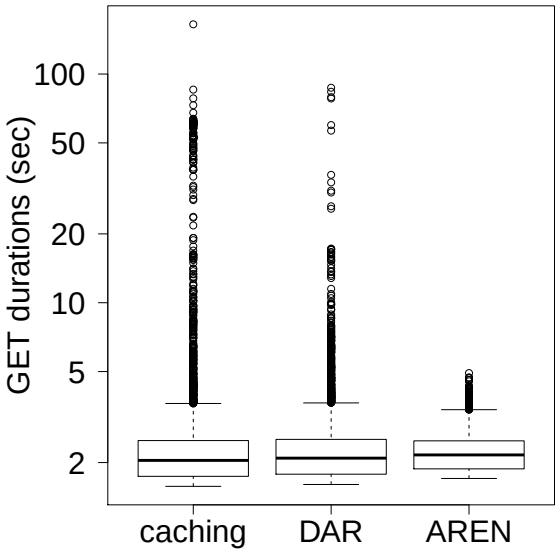

Figure 5.15: Upper quartile of *get* durations of the 10 most popular contents.

too short, wasting network resources. Overall, we have verified that 99% of all violations with caching last at least 4.64 seconds, with outliers up to 48 minutes. Since a straightforward implementation of non-collaborative caching relies on random scheduling, it lacks essential information for preventing edge nodes' overloading, and therefore provides poor resource allocation for popular content.

We analysed the number of replicas of the most popular content. Figure 5.16 plots the maximum number of replicas for the 2% most replicated contents. It shows that the vast majority of the content had a small number of copies. For instance, 98% of contents with DAR scheme had less than 21 copies. Yet, we are able to drop this number by two-thirds with AREN scheme. Our replication scheme performs still better for the most replicated content. While the maximum number of replicas using DAR and LRU reached respectively 1574 and 1740, AREN's most replicated content had only 188 replicas. This means that AREN adapts replication efficiently for the most popular contents. It also dramatically reduces the amount of required storage space for additional replicas, as well as the required number of allocated peers for each content.

We have evaluated the efficiency of peers' network provision with AREN, caching, and DAR. Our analysis focuses on the aggregate bandwidth and bandwidth allocation variance during the peak of utilization. We have compared the results with the three schemes to an assumption of unlimited network and storage capacities, described in Subsection 5.4.3. The the aggregated bandwidth is depicted in Figure 5.17. Overall, AREN performs much better than caching and DAR, quite similar to the unlimited assumption. It allows us to increase the aggregate bandwidth by almost 20%, hence achieving faster content delivery. DAR and LRU caching schemes are not able to reduce the load of peers with popular contents. So that, they undermine considerably the aggregate bandwidth of the content delivery system.

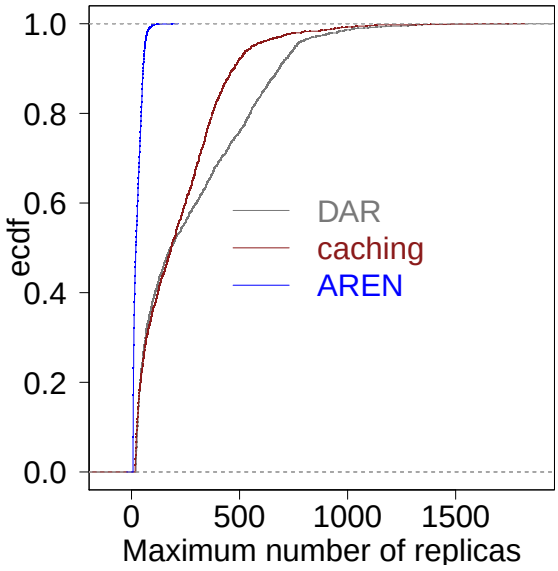

Figure 5.16: Maximum number of replicas for the 2% most replicated contents.

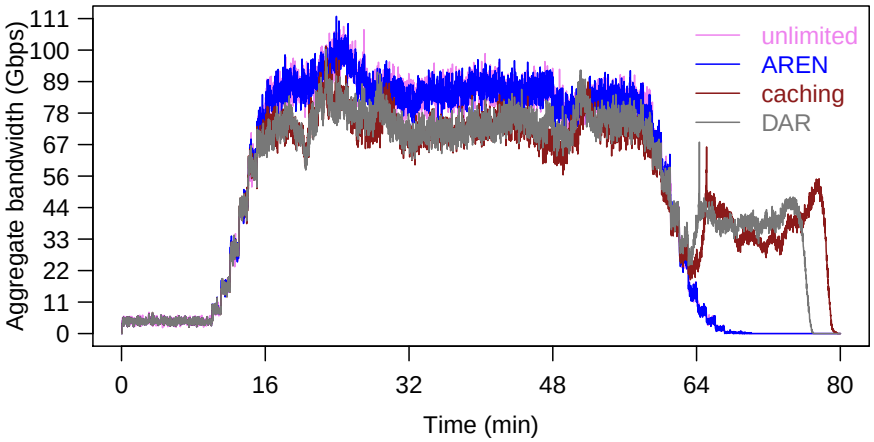

Figure 5.17: Aggregate bandwidth with caching, DAR, AREN, and an assumption of unlimited network and storage capacities.

The four graphs in Figure 5.21 show a detailed view of the aggregate bandwidth during the maximum utilization period. As expected, unlimited scheme presents the highest aggregate bandwidth picks, however AREN scheme allocates bandwidth quite closely to it. In general, DAR and non-collaborative LRU caching had similar performances. This suggests that random scheduling is not suitable for providing efficient bandwidth allocation for highly available content. A detailed analysis of bandwidth allocation, e.g. minutes 17 to 22 of Figure 5.21, allows us to conclude that random scheduling approach is not able to take advantage of spare bandwidth efficiently when the system is heavily loaded. AREN scheme overcomes this by providing necessary bandwidth

for consumers' SLA, and handling popular content replication properly. We have also observed that algorithms that balance the expected bandwidth load per node, such as DAR, does not allow us to improve the aggregate bandwidth for content availability.

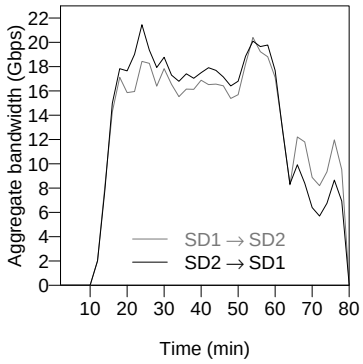

Figure 5.18: LRU caching .

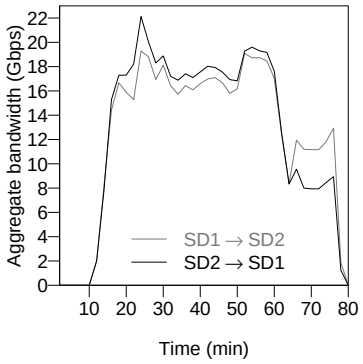

Figure 5.19: DAR.

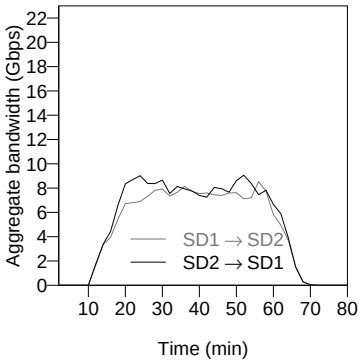

Figure 5.20: AREN.

To observe the dispersion of bandwidth allocation throughout peers, we measured the variance of bandwidth provision per second. Figure 5.22 presents four graphs with variance values per second, during the period of maximum utilization, for unlimited, AREN, non-collaborative LRU cahe, and DAR schemes. Unlimited scheme has the highest variance values. Alongside AREN, it shows that imbalance in bandwidth allocation per peer matters to improve content availability with regard to changes in popularity and bitrate enforcement. That shows the higher the imbalance in bandwidth is, the better is the the network resource provision. Surprisingly, DAR scheme presented a slightly smaller values of variance compared to non-collaborative LRU caching.

In order to control the impact of storage traffic in edge networks, transfers between nodes from different storage domains must be avoided as much as possible. AREN scheme enforces a *nearest source selection* scheduling policy, described in Subsection 5.4.3, which prioritizes the selection of intra-domain sources for requests. That allows us to reduce significantly the inter-domain traffic burden compared to a pure random scheduling. Figure 5.18, Figure 5.19, and 5.20 plot the aggregate bandwidth exchanged between the two storage domains. The enforcement of our straightforward policy in AREN scheme reduces nearly 60% of the overall traffic inter-storage domains compared to non-collaborative LRU caching. Once again, DAR performed very similar to caching.

5.5 Conclusion

In this chapter, we make two main contributions to enhance significantly content availability in CDNs. Firstly, we proposed Caju, a novel content distribution design for edge networks. Caju provides the capability to manage storage and network resources from both infrastructure-based and peers in edge networks in a collaborative manner. Based on Caju's design, we presented AREN, an novel adaptive replication scheme for content distribution in edge networks. AREN

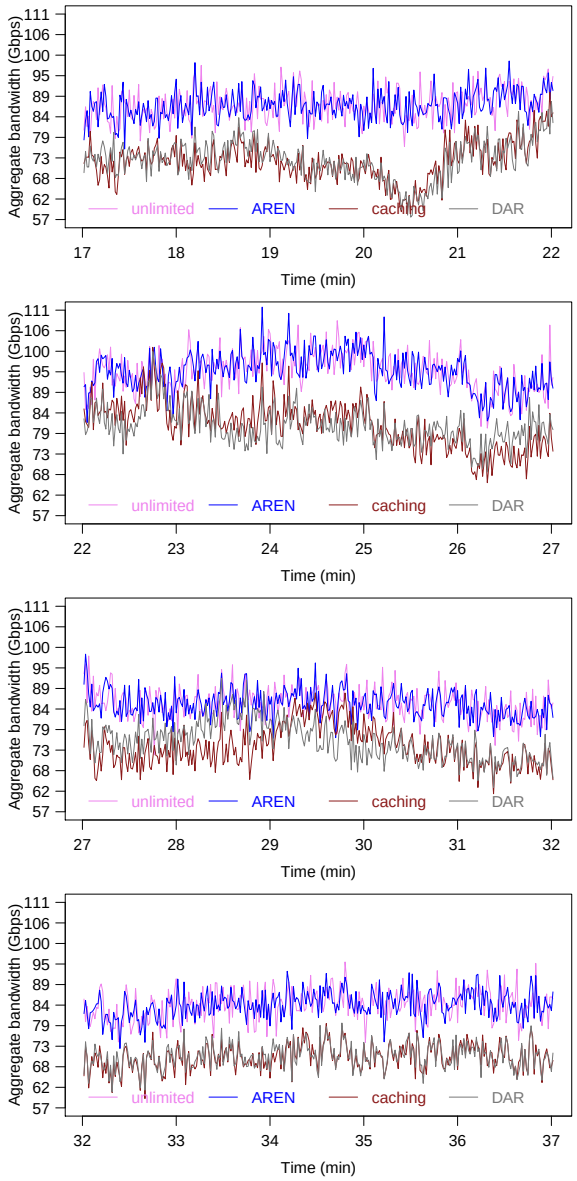

Figure 5.21: Aggregate bandwidth during the 20 heaviest loaded minutes with caching, AREN, DAR, and an assumption of unlimited resources.

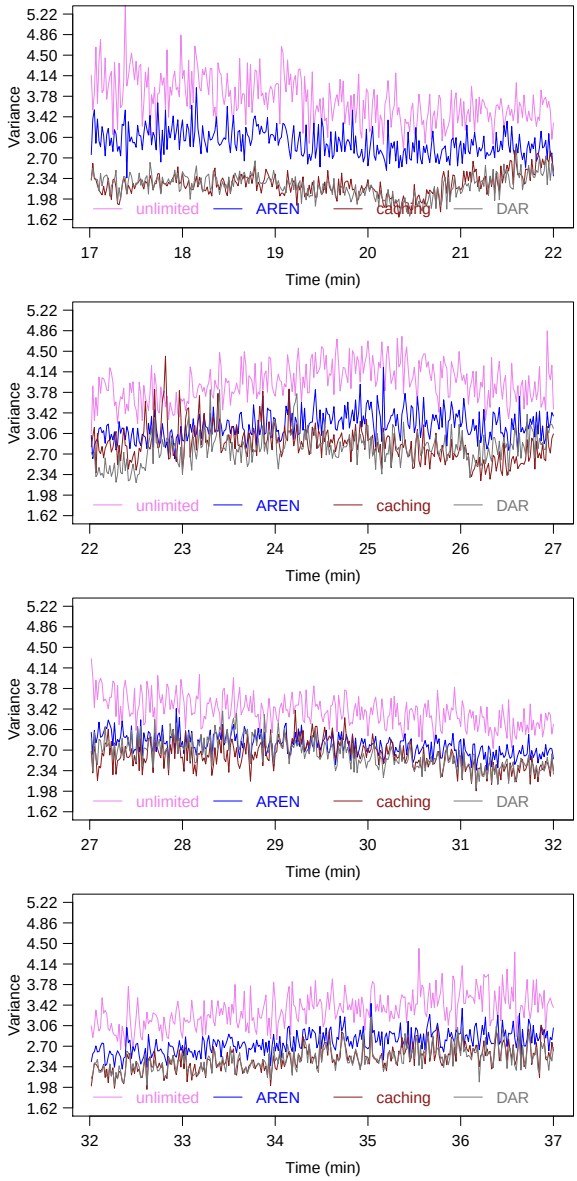

Figure 5.22: The variance of bandwidth provision during the 20 heaviest loaded minutes with caching, AREN, DAR, and an assumption of unlimited resources.

provides high content availability by enforcing bitrate of strict SLA contracts. It allows us to reduce the number of SLA violations by tracking bandwidth reservation mechanism on edge nodes and operating a collaborative caching mechanism properly. Contents are stored or removed from the the local caching according to the current bandwidth reservation. Our evaluations show that AREN consistently outperforms common replication schemes. It provides a seven-fold reduction

in the storage capacity for replicas when compared to non-collaborative LRU caching. AREN effectively prevented about 99.8% of all SLA violations when the storage system is heavily loaded. We also found that aggregate bandwidth of CDN on edge networks increases by roughly 20% the with AREN. These finds suggest that our approach provides a much better results, improving significantly content availability.

Chapter 6

Predicting Demand and Adapting Resource Allocation for Highly Efficient Content Delivery

Contents

6.1 Introduction

Replication schemes have become an important building block for content delivery network providers to improve content availability and meet consumers expectations. A *good* replication scheme must offer adaptive content replica maintenance to cope with the demand and to reduce resources usage, especially in edge networks. With these issues in mind, we introduced AREN, a novel Adaptive Replication scheme for Edge Networks, which was described and evaluated in Chapter 5.

We showed that AREN improves significantly the content availability for CDNs in edge devices, meeting QoS metrics for customers and reducing storage and network usage properly. AREN relies on a collaborative caching and a bandwidth reservation technique to adapt the replication degree and to enforce SLA contracts for customers. It applies a simple mechanism of popularity classification and content replication based on the current aggregated sum of bandwidth reservation and low/high bandwidth thresholds. Simulations with synthetic workload demonstrated that this approach provides improved results, leading to an outstanding content availability. As described in Chapter 5, it outperformed non-collaborative caching by preventing almost 99.8% of SLA violations. By reducing the total number of replicas, AREN reduces storage usage for replication and increases the aggregate bandwidth. Unlike non-collaborative caching, AREN diminishes the dependency on cache replacement policies by decreasing consistently the number of replicas.

Although AREN's results showed that it is highly efficient in replicating multimedia workloads, its deployment raises considerable issues for Internet providers in edge networks. One of the main disadvantage of this approach, that can make Internet providers reluctant to its use, is that it depends on changes on the functioning of the network stack. Efficient bandwidth reservation for meeting deadlines, like D^2TCP [136], requires major adjustments to the transport network layer to provide end-to-end bandwidth reservation properly.

To overcome this important issue, and encouraged by findings with AREN's threshold-based approach, we introduce in this chapter two novel replication schemes: Hermes and WiseReplica. Both schemes are based on flexible, robust statistical learning models. The main insight behind these schemes is to *learn* how to perform efficient content replication with AREN, and to replicate content efficiently without requiring changes on the network stack. They allow us to predict the demand of contents for providing adaptive replication from lightweight measurements of the request arrival process. We propose Hermes as a general-purpose demand-aware replication scheme, that adapts replication degree of Internet contents based on accurate predictions of their popularity. We extended and specialized our first learning model in streaming videos, then we came out with WiseReplica. Basically, it differs from Hermes in functioning and purpose. WiseReplica ranks Internet videos in order of *hotness*, allowing us to gradually allocate replicas for hungry-resources services as video on demand (VoD). The bottom line for both replication schemes remains to instrument a collaborative caching, creating and removing replicas, according to content requests. We argue that, through accurate predictions, we are able to react to demand changes promptly, and prevent SLA violations. Simulations with YouTube traces suggest that our approach is quite similar in performance to AREN, improving content availability

and reducing edge resources utilization properly.

We organized this chapter as follows. We provides a brief overview about statistical learning in Section 6.2. We describe the design of our two learning models, Hermes and WiseReplica in Section 6.3. Then we describe the replication strategy based on predictions in Section 6.4. In Section 6.5, we show the performance evaluation for both new replication schemes. Finally, we conclude this chapter in Section 6.6.

6.2 Giving A Brief Overview about Statistical Learning Modelling

Statistical learning is about learning from seen data in order to predict unseen data with minimal error. Data comprise measurements $\mathbf{x}$ represented by a feature vector with a fixed number of dimensions p ($\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$) from the input space $\mathcal{X}$. From these measurements, a learning algorithm is able to yield a function that minimizes prediction errors.

In the sample data, each $\mathbf{x}$ is coupled with a output variable y from the space $\mathcal{Y}$. We consider sample pairs $(\mathbf{x}, y)$ drawn *independently and identically distributed* (i.i.d.) from a fixed but unknown joint distribution $Pr(X, Y)$. Supposing that the seen and the unseen samples comes from the distribution $Pr(X, Y)$, we seek a prediction function $f : \mathcal{X} \to \mathbb{R}$ which is the most accurate in forecasting unseen data. This function actually belongs to a set of predefined functions space $\mathcal{F}$ and is selected by the learning algorithm. The output of the learning algorithm is $\hat{y} = f(\mathbf{x}) = I_{\{f(\mathbf{x})>0\}}$.

Learning algorithms measure the quality of the prediction by a risk function $\mathtt{r} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the proximity of the predicted target $\hat{y}$ to the ground truth y. We could define the statistical risk associated to the prediction function $f : \mathcal{X} \to \mathcal{Y}$ as:

$$\mathtt{r}(f) = P(f(X) \neq Y) = \mathbb{E}_{X,Y}\left[I_{\{f(X)\neq Y\}}\right]$$

Therefore, the learning algorithm goal is to find a hypothesis function $f^\star$ among all possible measure function in $\mathcal{H}$ for which the risk $\mathtt{r}$ is minimal, as depicted in Figure 6.1.

As $Pr(X, Y)$ is unknown, it is not possible to obtain directly the value of f at the data sample. To cope with this problem, we follow the Empirical Risk Minimization (ERM) principle [137] which states: one should choose the hypothesis $\hat{f}$ from a fixed and restrained class of functions $\mathcal{F}$ which minimizes the empirical risk. According to the Law of Large Numbers, if the the number of samples tends to infinity then the empirical risk $\mathtt{r}_{emp}(f)$ tends to the expected risk $\mathtt{r}(f)$. However, this problem is not easy to optimize due to the discontinuous parts of the indicator function $I_{\{f(x)\neq y\}}$ in $\mathtt{r}_{emp}$. To overcome this issue, a standard approach is to take a surrogate loss function of the indicator function, easier to minimize in practice by standard optimization algorithms and keep theoretical guarantees provided by the ERM principle.

In this section, we describe the two main categories of learning problems in Subsection 6.2.1, then we briefly describe the most common learning approaches in Subsection 6.2.2, and finally, we describe in Subsection 6.2.3 the common framework for learning and predicting used in this thesis.

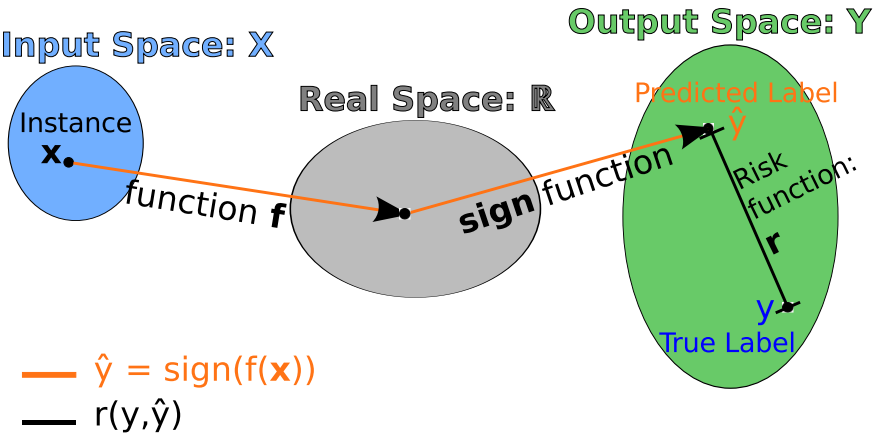

Figure 6.1: Overview of the learning task: given the input $\mathbf{x}$, a hypothesis f is learned by the learning algorithm. Then we take the sign (thanks to $I_{\{.\}}$) of the output of f and we compare it with the true outcome y by the risk function $\mathtt{r}$.

6.2.1 Categories of Learning Problems

There are two main ways of learning from data: supervised and unsupervised learning.

In supervised learning, each input measurement is coupled with a y, a label selected by an *oracle*, or any source of learning, from the output space $\mathcal{Y}$. To learn, we take N pairs $(\mathbf{x}, y)$ drawn *independently and identically distributed* (i.i.d.) from a fixed but unknown joint probability density $Pr(X, Y)$. This is true for both training and testing datasets. For instance, we consider the training dataset $S = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ of N pairs $(\mathbf{x}, y)$. Using this dataset, the supervised learning algorithm searches for a function $f : \mathcal{X} \to \mathbb{R}$ in a fixed function class $\mathcal{F}$. State-of-the-art algorithms, such as *support vector machines* (SVM) [36] or *ensemble methods* [61], aim to find $f^\star$ in $\mathcal{F}$ with the lowest empirical risk defined as:

$$f^\star \in \arg\min_{f \in \mathcal{F}} \mathtt{r}_{emp}(f) \tag{6.1}$$

where $\mathtt{r}_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} I_{\{f(\mathbf{x}) \neq y_i\}}$ is computed over the training set, and $I_{\{.\}}$ is the indicator function which returns 1 if the predicate $\{.\}$ is true and 0 otherwise. In other terms, $\mathtt{r}_{emp}$ is a quality measure relating the label to the prediction provided by the function f.

In unsupervised learning, we have N samples $(x_1, x_2, \ldots x_N)$ of a random p-vector X having probability density Pr(X). Unlike supervised learning, we do not have outputs to learn. Instead, we are interested in inferring the properties of the probability density Pr(X). This allows us to have insights into how the data are organized or clustered.

6.2.2 Statistical Learning Approaches

There are three main learning methods in statistical learning: regression, where $y \in \mathcal{Y} \subset \mathbb{R}$; classification where $y \in \mathcal{Y} \subset \{0, 1, ..., K\}$ with $K \geq 1$; and learning-to-rank approach where y

gives an indication on the target order (formally represented by a permutation σ). Learning-to-rank model shares properties of both regression and classification. As in classification approach, the output $\mathcal{Y}$ is a finite set, and like in regression there is an ordering amongst the elements of $\mathcal{Y}$. Learning-to-rank approach has been a hot topic in the Machine Learning community for the last 10 years.

In this thesis, we designed a first, general-purpose learning model for demand-aware replication using classification. Its goal is essentially to classify videos according to popularity. We propose a second learning model, which is an enhanced and specialized in Internet streaming video, e.g. video on demand. We design our second model as a learning-to-rank problem that rank videos in order of *hotness*.

6.2.3 Providing a Common Framework for Learning and Predicting, and Implementation Details

We designed a simple framework to use statistical learning algorithms in our predictive, adaptive replication schemes, depicted in Figure 6.2. Our framework has two phases: (i) learning and (ii) predicting. Each phase has its own Internet content-based workload. Learning is a preliminary phase that runs offline in a batch mode, while the prediction can go online. In this chapter, both phases are performed with data from YouTube traces. In the learning phase, we first need the training dataset, such as an *oracle* or any source of reliable information for learning. In our case, the *oracle* is AREN. Then we feed this training dataset to the learning model of our replication schemes, represented here as a generic Replication Scheme blue boxes. This allows us to identify request arrival process patterns of seen Internet content for either classifying or ranking unseen ones. Once the learning phase has been accomplished, the Replication Scheme can run in a predicting phase, as indicated in the left-hand side of Figure 6.2. In this phase, inputs comes for measurements of the the request arrival process of workload, that permit accurately prediction about Internet videos' behaviours and instrumenting replication accordingly inside storage domains.

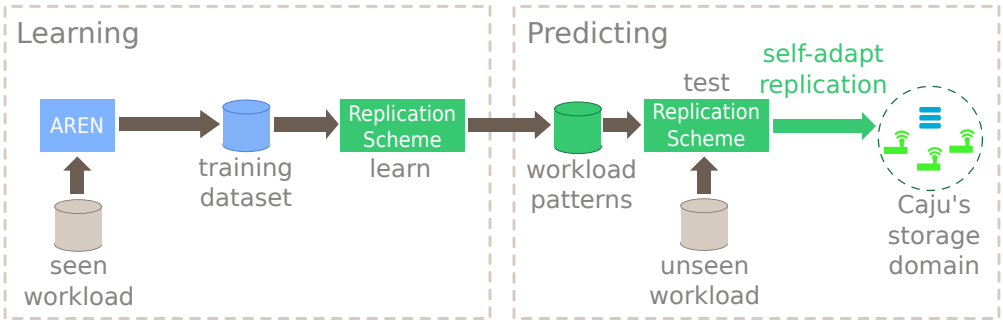

Figure 6.2: A generic framework for learning and predicting behaviour of Internet contents.

We implemented our learning model as a module of replication schemes, Hermes and WiseReplica, using Scikit-learn, a general-purpose machine learning library [99].

6.3 Designing Statistical Learning Models for Predicting Internet Content Demand

We present our two statistical learning models for predicting Internet content demand. As briefly introduced in Section 6.1, the models differ in functioning and purpose depending on their replication schemes. While Hermes is rather a scheme for adapting the number of replicas according to the demand for generic Internet contents, WiseReplica was designed specifically to replicate Internet streaming videos, particularly VoD content.

6.3.1 Input Data for Learning

One of our primary efforts towards accurate predictions was to design a set of input data to gather as much information about consumers' interactions as possible in a easy manner. Our input data comes from measurements of request arrival process. We chose this approach because it provides simple procedure to collect information of consumers' interactions. For example, in hybrid CDNs, this data can be collected from logically centralised coordinator servers that are already in charge of accountability or admission control tasks. In addition, we added labels to each line of our measurements. In the learning phase, labels allow us to identify the known data, as described in Subsection 6.2.3.

We represent the input space $\mathbf{x}$ is a content represented by 10 lightweight measurements from the request arrival process. These measurements are content size, network availability, network usage (load), current number of transfers and replicas, inter-arrival time between requests (delta), aggregate number of transfers, mean of time between requests (mtbr), life time, and average bitrate. We compute averages and means from up to the five last requests. A brief description of our input data for predictions is shown in Table 6.1.

Measurement	Description
Size	content size
Availability	available network resources on the replica set
Load	current network resources usage
Active transfers	active transfers or sessions to a content
Replicas	current number of replicas
Delta	last inter-arrival time between requests
Views	aggregate number of views
MTBR	Mean Time Between Requests of the last five requests
Life time	time since the first access request
Average bitrate	average bit transfer rate, or simply bitrate, of the last five requests

Table 6.1: Input measurements for predicting content demand.

As described in Subsection 6.2.3, we assume the measurements come from simulations using AREN and YouTube traces.

6.3.2 A Generic Demand-Aware Learning Model for Classifying Internet Content

We design a model for providing a demand-aware classifier of Internet contents for Hermes replication scheme. The model's data come from measurements of request arrival process, as described in Subsection 6.2.3. We assume that AREN (Chapter 5) provides improved results from where we can learn. Considering the measurements of the request arrival process as *inputs*, we denote the *outputs* to each dataset line regarding the content popularity. Output labels track the behaviour of AREN functioning, and allow us to classify requests. Labels permit distinguishing two main groups, *popular* from *non-popular* Internet contents, described as follows:

- **Non-popular Internet contents**: Contents with non-popular output labels are those whose access pattern of its request arrival process has not trigged any increasing on the initial replication degree. According to recent findings [129], the popularity of Internet contents follows a Zipf-like distribution, consequently most of them likely belong with this group. In AREN, they do not require any extra replica.
- **Popular Internet contents**: If during the life time of a content, its replication degree is modified by AREN, we attribute a popular label to it. In addition, we introduced further information to this group in order to capture the behaviour of the replication maintenance. Depending on the decision taken by AREN, there will be three types, or subclasses or labels, of popular contents: *increasing*, *keeping*, or *decreasing*. This allows us to interpret the measurement as a trigger for changing the resource allocation of that content, in our specific case, modifying the number of replicas.

Therefore, we model our problem in a two-step approach as follows:

- **Popularity classifier**: This learner allows us to classify contents into non-popular and popular. Since the popularity of Internet contents follows a Zipf-like distribution, popular contents can be seen as rare events. Hence, we identify popular contents as anomalies through an unsupervised learning method with binary outputs.
- **Replication classifier**: Here we consider popular contents only. There are three subclasses of replication for popular contents: increasing, keeping, and decreasing. In this case, we use a multi-class supervised learning method.

Implementation details. Our two-step classifier is based on *support vector machine* (SVM) methods [36]. According to Friedman *et al.*, SVMs are a set of robust supervised learning methods, that produce accurate, non-linear boundaries for classifiers by constructing a linear boundary in a large, transformed version of the input space. From Scikit-learn, we selected two main procedures: `sklearn.svm.OneClassSVM` for popularity classifier, and `sklearn.svm.SVC` for replication classifier.

6.3.3 An Accurate Learning-to-Rank Model for Video-on-Demand Services

The main purpose of this learning model is to capture demand growth dynamics and system resources demand of VoD services. In other words, the model must allow us to rank Internet streaming videos in order of *hotness*. This can be modelled as a learning-to-rank problem.

Given an i.i.d. sample $(\mathbf{x}, y)$ such as described in Subsection 6.2, and assuming the same inputs described in Subsection 6.3, we model the machine-learned ranking of streaming videos as follows. The supervision y associated to each input video $\mathbf{x}$ is based on four possible ordered values which gives an indication for the final target ranking. In our model, $\mathcal{Y} \in \{0, 1, 2, 3\}$, whose labels are {*cold*, *hot*, *very hot*, *viral*} respectively. It represents a natural ranking for Internet videos. Using this ranking model, we intend to provide a measure of video *hotness*, which is closely related not only to the popularity, but also to the consumption of system resources.

As in the previous learning model, AREN represents the enhanced way to serve VoD service according to video encodings and popularity, whose functioning we are very interested in learning. In this empirical approach, a video requires additional replicas only if there exists a certain number of concurrent accesses, where concurrence is measured by checking a high threshold of the current reserved bandwidth, as detailed in Chapter 5. We assume that most of the videos are ranked as *cold* videos, and they do not need additional replicas during their lifetime. As found by Szabo and Huberman [129], concurrent access are rare events in this kind of workload as well as videos that falls in one of three upper-ranking positions of our model. Thus, it provides a quite fair approach to identify the hottest videos.

Raw data from AREN permits us to easily distinguishing between two ranking positions only, cold and hot videos, i.e. requests to rarely accessed, cold videos are all those that do not trigger any replica creation, or those that resulted in deletions. However, there is a lack of information about different ranking positions of hot videos. Hence, depending on the frequency of replica creation, we add information to requests to popular videos ranking them in hot, very hot, or viral. To define these three levels of *hotness*, we run simulations with AREN, collected the distribution of replicas creation in milliseconds, and split it in three nearly equal parts by observing the 66-percentile and 33-percentile inter-creation time for new replicas. This means that the higher is the frequency of replica creation, the hotter is the video, and the higher is the ranking position. Now, collected data suit model's definitions very well.

Finally, the learning-to-rank module finds a function f from equation (6.1) with the constraint of maintaining the prediction order: $\forall i, j, i \neq j, y_i > y_j$ then $f(\mathbf{x}_i) > f(\mathbf{x}_j)$, as explained in [21]. In that case, theoretical performance guarantees are provided. Practically, the use of the mean square error $(y - f(\mathbf{x}))^2$ instead of the indicator function $I_{\{.\}}$ (which is hard to optimize because it is non-differentiable) allows us to ensure an optimal learning-to-rank algorithm.

Implementation details. We implement our model using *ensemble methods* available on SCIKIT-LEARN machine learning library [99]. According to Friedman *et al.*, ensemble learning consists of a set of very popular supervised methods, that are robust, simple to train and tune, and have a remarkable prediction performance.

6.4 Proactive Replication Strategy to Deliver Highly Available Content

We worked on two strategies for replicating Internet content. Both strategies use a learning model for adapting replication in a proactive way. Our strategies essentially differ in purpose regarding the goals of each replication schemes, namely Hermes and WiseReplica.

As a general-purpose replication scheme, Hermes enforces content availability according to customers' needs. Similar to SLA enforcement used in AREN (Chapter 5), we suppose that customers define the minimum amount of network resources to fetch any Internet content through SLA contracts. In this scenario, Hermes must cope with the replication of contents on the basis of customer-oriented minimum transfer rate. Therefore, our strategy is to prevent violations as much as possible regarding the needs of customers as individuals.

In WiseReplica, the replication goal is significantly different in two aspects. First, we consider videos rather than generic Internet contents. Second, we assume that the replication scheme must cope with the demand of each video based on its encoding setting. Unlike Hermes, the replication strategy must provide a video-oriented content provision through SLA contracts. While the SLA contracts of Hermes are drawn up between individuals and Internet Service Providers (ISPs), we assume that WiseReplica is designed to enforce contracts drawn up directly amongst ISPs and content providers. In this case, the replication strategy is to rank videos in order of demand and then adapt their replication gradually.

We summarize our replication goals and strategies in Table 6.2.

Scheme	SLA Purpose	Goal: Prevent SLA Violations	Strategy
Hermes	Customer-Oriented	On a Customer Basis	Classify and Adapt Replication Uniformly
WiseReplica	Video Encoding-Oriented	On a Video Basis	Rank and Adapt Replication Gradually

Table 6.2: A summary of the two proactive replication goals and strategies.

6.4.1 Replicating Generic Internet Content Regarding Customers' Needs

In our general-purpose replication scheme for Internet contents, called Hermes, the goal is to enforce network resources on a customer basis. We assume that customers define their network resources' needs through SLA contracts, where need is represented by a specific transfer rate. Then, Hermes must maintain replication in order to prevent violations. Hermes cope with this by simply adapting the number of replicas of Internet contents according to their popularity. Based on the learning model described in Subsection 6.3.2, Hermes classifies requests in four different

popularity-based classes, then maintains replication degree accordingly. But only popular videos require more replicas.

In this strategy, network load is a key feature that measures the demand of users, i.e. their expected transfer rates. Combined with other learning model features, it allows us to identify accurately when the replication degree of a popular content must be changed. Replica maintenance is two-fold: creation and deletion.

- **Creation**. Replica creation is straightforward. In order to cope with unexpected increasing on content demand, Hermes strategy is to create a fixed number of replicas N. So that, as soon a request to a content is classified as popular with *increasing* demand, N replicas are created in randomly and uniformly selected peers. We evaluate different values of N in Section 6.5.

- **Deletion**. In general, Hermes deletes content replicas when a request to a popular content is classified as *decreasing* popularity. We are more conservative in terms of deletions than in creations. Just one replica is deleted by classified decreasing popularity event. Similar to AREN (Chapter 5), Hermes observes the minimal replication degree of an content m before decreasing its replication degree.

6.4.2 Adapting the Replication According to the Ranking of Streaming Videos

Our utmost goal with WiseReplica is to contribute to meet increasing customers expectation on Internet streaming videos, specially video-on-demand services. To enhance VoD delivery, we assume that rebuffering is a major issue to be addressed. We propose to cope with this issue by enforcing minimum average bitrate of each streaming as the main QoS metric. In an optimistic scenario, content and CDN providers must be committed to enforce minimum average bitrate for videos through SLA contracts. Unlike Hermes, WiseReplica enforces bitrate of videos based on their encoding settings and a video *hotness* ranking, detailed in Subsection 6.3.3.

WiseReplica gradually adapts the replication of videos according to the forecasts of their hotness rank positions:

- **Creation**. To cope with SLA violations and meet customers' expectations, we consider four quite simple creation policies, namely uniform, linear, quadratic, and exponential. They are respectively defined as follows: B, Br,Br^2, and B^r, where B is a constant and $r \in \{1, 2, 3\}$ the rank positions, namely hot, very hot, and viral. We report on creation policies' performances in Section 6.5.

- **Deletion**. WiseReplica adopts the same simple deletion policy of Hermes. Whenever a video is ranked as *cold*, one replica is deleted until the minimum replication degree m is reached.

6.5 Evaluation

In this section, we evaluate the performance of Hermes and WiseReplica in delivering highly available Internet content through. We use simulations using YouTube traces to evaluate our replication schemes and compare them with other techniques under a fair scenario. We aim to study in details the how they are in meeting customers' expectations and how efficiently they allocate resources of Internet content delivery services in edge networks. Therefore, we first describe the workload in Subsection 6.5.1, then we present the simulated scenario in Subsection 6.5.2, we show the comparable replication schemes in Subsection 6.5.3, we defines evaluated SLA contracts in Subsection 6.5.4, we evaluate Hermes and WiseReplica in Subsections 6.5.5 and 6.5.6 respectively.

6.5.1 Workload from YouTube Traces

The workload is at the core of our evaluation. We define a workload that captures the main features of multimedia web content using YouTube traces.

A fair reproduction of user interactions to Internet videos is essential to evaluate the availability of Internet content properly. Hence, we study in this work a workload that combines YouTube traces [51] to well-known Internet contents' access patterns [129]. We are particularly interested in reproducing realistic popularity growth curves, considering advanced coding setting and common VoD demand patterns.

Figueiredo *et al.* [51] collected and characterized the growth patterns of YouTube videos, whose datasets are currently available online [1]. They analysed three types of YouTube videos sets: videos that appear on YouTube top list, videos that were banned from YouTube due to copyrights violations, and videos that were randomly selected through API calls. They crawled once a number of videos' daily features. For each video, there are up to 100 daily measurements, or daily available samples, per feature. In this work, we are mostly interested in the measurements of *view data* feature, that depicts the popularity growth curve of a video through a array of cumulative number of daily views ranging from 0 to the total number of views. This chapter's evaluation use their YouTube traces. Before integrating YouTube traces to our workload, we first processed their YouTube datasets to remove inconsistent measurements, such as videos with no views. Basically, we got rid of videos with small number of total views (those smaller than the first quartile) and videos with few daily measurements (those smaller than the third quartile). That allows us to pick off 20% most representative YouTube growth patterns, accounting for 21827 distinct curves. Then, for a matter of simplicity, we randomly selected, with a uniform distribution, curves from this preprocessed data to be assigned to videos of our workload.

In order to reproduce realistic, high quality videos encodings, we consider the YouTuve advanced encoding settings[2]. Table 6.3 depicts the set of high definition (HD) video encodings that

[1]The Tube over Time: Characterizing Popularity Growth of YouTube Videos. http://www.vod.dcc.ufmg.br/traces/youtime/data/, January 2013.

[2]Advanced encoding settings for YouTube videos. http://support.google.com/youtube/bin/answer.py?hl=en-GB&answer=1722171, March 2013.

we use in this work.

Type	Video Bitrate	Mono Audio Bitrate	Stereo Audio Bitrate	5.1 Audio Bitrate
1080p	50 Mbps	128 kbps	384 kbps	512 kbps
720p	30 Mbps	128 kbps	384 kbps	512 kbps
480p	15 Mbps	128 kbps	384 kbps	512 kbps
360p	5 Mbps	128 kbps	384 kbps	512 kbps

Table 6.3: Advanced encoding settings for YouTube videos used in this work.

We summarize our workload settings in Table 6.4, which lists default values and common parameters. Finally, contents, including videos, are always divided and distributed in chunks or segments of fixed size, 2MB. We would like to highlight that video encoding settings are useful only for WiseReplica replication scheme.

Workload	
Requests per user	uniform
Experiment duration	4 hours
Mean requests per second	100
Requests fractions	5% of creations, 95% of views
Video size (follows Pareto)	shape=3,between 13MB and 1.6GB
Video popularity (Zipf-Mandelbrot)	shape=0.8, cutoff=number of videos
Videos' creation (Poisson)	λ=creations per second
Popularity growth from YouTube traces	21827 distinct patterns
YouTube encoding settings (bitrates)	5Mbps, 15Mbps, 30Mbps, 50Mbps

Table 6.4: Workload's default settings.

6.5.2 Evaluation Scenario

The evaluation scenario for this chapter is the same as the one used for AREN's evaluation in Chapter 5. For a matter of convenience, we provide a brief reminder of this scenario, depicted in Figure 6.3. It includes 4002 nodes, arranged across two Caju's storage domains (further details in Chapter 5). There are one coordinator and 2000 peers per storage domain. Storage and network capacities differ according to the device role. Coordinators have 20TB of storage capacity and full-duplex access link of 4Gbps. Peers contribute 200GB each, equipped with 100Mbps full-duplex links. Coordinators contribute with a small fraction of aggregate edge resources, i.e. 5% of the storage capacity and only 2% of the total network capacity. This draws our attention to

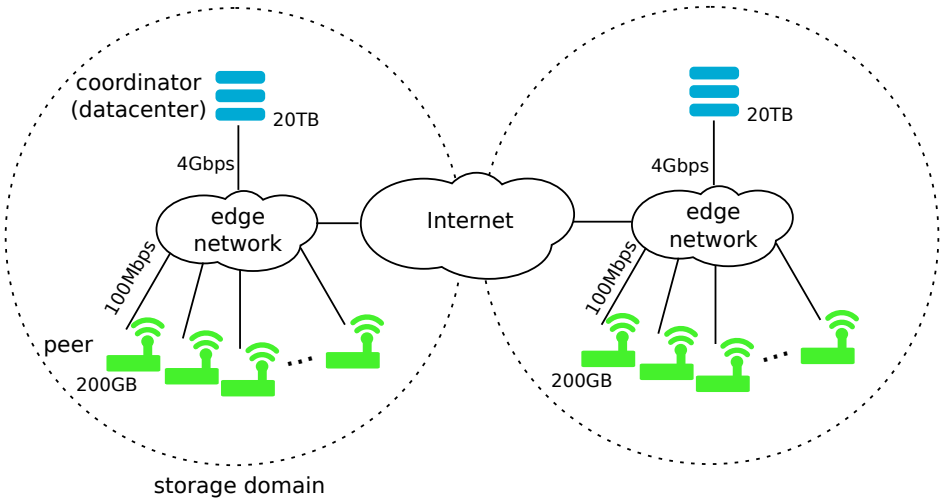

Figure 6.3: Evaluation scenario.

the performance of replication schemes towards peers resource allocation. We assume only 1% peers' storage is available for caching additional replicas, namely 2GB.

We implemented and evaluate this work using simulation. To this end, we developed a simulation tool on top of PeerSim [88] to implement storage domains in edge network and bandwidth scheduling. Our design focus on network's resource allocation accuracy for simulating bitrate enforcement and concurrent videos views properly. We have performed our simulations using servers equipped with Intel Xeon E5450 3.00 GHz, and a RAM of 4GB. Further details about our simulation's component are available in Appendix A.

6.5.3 Comparable Replication Schemes

We compare Hermes and WiseReplica with two other schemes.

- **Non-collaborative caching**. Adaptive replication schemes based on non-collaborative caching, such as those that uses Least Recent Used (LRU) algorithm, are easy to implement and deploy. A new replica is created whenever a user requests to view a video. LRU replacement is enforced regarding the static percentage of the local storage capacity for caching of 1%.

- **AREN, an Adaptive Replication scheme for Edge Networks**. AREN provides enhanced results, shown in Chapter 5. It relies on a network stack that runs a deadline-aware transport protocol, similar to Wilson *et al.* [141] work. Basically, AREN operates bandwidth reservation and collaborative caching to provide an adaptive number of replicas for videos. Then it replicates contents according to aggregate network usage by enforcing a low and high thresholds. This makes the video replication a function of bandwidth reservation, and ensures that network and storage provision follows content demand properly. Although

this empirical approach is hard to be adopted in a real deployment, it allows us to achieve enhanced results, preventing *all* SLA violations, enhancing network usage and decreasing storage usage dramatically. Here, we consider AREN as an *oracle-like* approach.

6.5.4 Defining SLA for Highly Available Content

We define SLA contracts for evaluation based on the purpose of each replication scheme. We classify them on customer-oriented and video-oriented SLA contracts, defined as follows.

- **Customer-oriented SLA contract**. As described in Section 6.4, Hermes provides general-purpose replication strategy for enhancing availability of Internet contents. In this scenario, we assume that customers will be willing to define the minimum speed of any content provision through ISP's SLA contracts. On behalf of customers, ISPs provide adaptive content replication in order to prevent violations and meet their overall expectations for Internet content. For this evaluation of Hermes, we assume that all consumers expect the same minimal QoS metric for fetching Internet contents, Hermes must cope with a SLA contract whose the minimal average bitrate is about 28Mbps, or 14 chunks/s regarding Caju's settings. We assume that this generic amount bitrate provides roughly enough resources for fairly fetching most of the current multimedia contents, including streaming ones. We consider that a SLA violation occurs whenever a customer does not observe her minimum average bitrate for any fetched content.

- **Video-oriented SLA contract**. For evaluating WiseReplica, we assume that content and content delivery providers are committed to improving the Internet video quality for customers in a content-oriented approach, as detailed in Section 6.4. In our case, WiseReplica must ensure VoD quality by avoiding rebuffering. Therefore, we consider a global, simple SLA contract drawn up to provide a minimal average bitrate according to each Internet video encoding setting. A SLA violation happens whenever the system fails to enforce the minimal average bitrate for any viewer's request.

6.5.5 Boosting Internet Content Delivery with Hermes

6.5.5.1 Predictions and Replication Performance

Hermes relies on predictions to identify popular Internet contents and enforce QoS metrics through replication. Hermes' performance depends mainly on (i) prediction accuracy and (ii) the efficiency of the replication policy. In Section 6.3.2, we explain that our two-step classifier relies on SVM statistical learning methods. To measure the prediction accuracy of each step, we vary the kernel, the main SVM parameter. We consider four kernels: *Radial Basis Function* (RBF), Linear, Polinomial (Poly), and Sigmoid. For evaluating our classifier, we use the framework described in Subsection 6.2.3.

Popularity prediction accuracy: The first step of our learning model predicts Internet contents popularity through a binary classification. We used a dataset with 286823 samples of view

requests, whose 1.31% of them belong to popular contents. Figure 6.4 depicts the *receiver operating characteristic* (ROC) curve. ROC curve is one of the most common ways of evaluating the efficiency of a binary classifier. This plot allows us to select the best classifier by measuring the true positive rate versus the false positive rate, and by computing the area under the ROC curve (AUC), where the value 1 represents the optimal classifier. Using RBF kernel, our classifier reaches an AUC of 0.97, quite close to the optimal value. Therefore, RBF kernel is the best choice for predicting popularity.

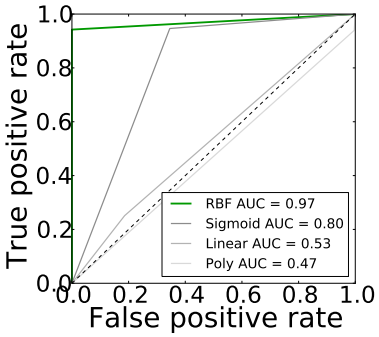

Figure 6.4: ROC curve for popularity classifier.

Replication prediction accuracy: For the second step of our learning model, the goal is to predict the replication action for popular contents in three classes. The dataset for this step contained 612754 view requests. Figures 6.5 shows total precision rates using different SVM kernels. RBF outperforms the three other kernels with the highest precision rate of 0.98, becoming our best choice. Unlike popularity predictions results, Linear and Poly kernels performed quite well, both scoring 0.97.

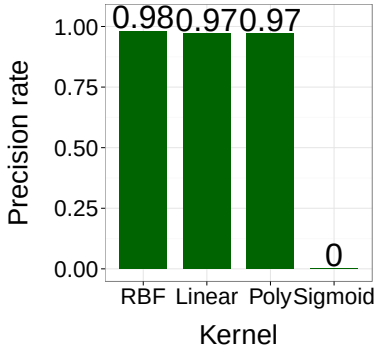

Figure 6.5: Precision rate per kernel.

Evaluating the replication policy: Whenever the learning module of Hermes predicts that a content needs more replicas, we assume that d new replicas must be created once for preventing violations. Figure 6.6 measures the number of violations for different values of d, whose values

vary from one to 13. When d ranges from seven to 10, there is no violations. This suggest that since popularity predictions are accurate, a simple replication policy should suffice. However, if d is bigger than 10, replication adds enough load to cause violations. Hence, we select d equal to seven as the most appropriate value for preventing violations of the evaluated workload.

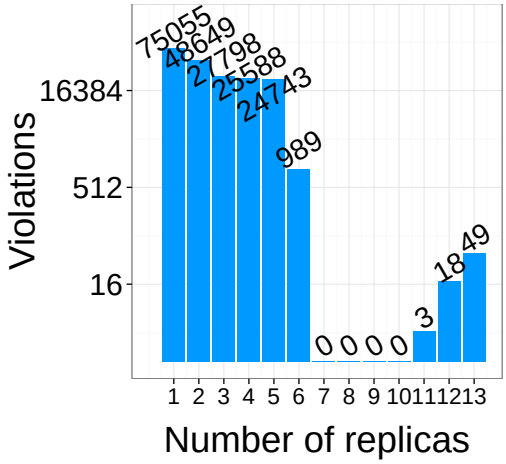

Figure 6.6: Parameter d for adjusting replication degree.

6.5.5.2 Resource Allocation Results and Analysis

We compare Hermes with a non-collaborative caching and AREN, all described in Subsection 6.5.3. We evaluate the network and storage usage, as well as the number of violations.

We aim to adapt the number of replicas to the number of views of a content, especially for the most popular ones. Figure 6.7 plots the maximum number of replicas for the 1% most popular contents. Using caching, the maximum number of replicas is high, ranging from 817 to 1377. AREN permits decreasing significantly the lower and upper limits, to 7 and 39. Hermes also reduces the maximum replica range, which is from 9 to 58. More interestingly, the shape of the replication curves of Hermes and AREN are quite similar indeed. It confirms that our predictions are accurate, and that a simple replication policy works properly.

Reducing the number of replicas implies that the systems requires less storage for replication. Figure 6.8 shows storage usage for replication by replication scheme. Although Hermes uses more storage for replication than AREN, its usage remains two orders of magnitude below a non-collaborative caching. The maximum storage usage for AREN, Hermes, and a non-collaborative caching were 3, 49, and 7956 GB respectively. Hermes creates more replicas than AREN because it does not rely on bandwidth reservation to prevent violations. Despite that, Hermes maintains replicas efficiently, keeping storage usage very low, and making cache replacement policies redundant.

In terms of violations, Hermes performance is also quite similar to AREN. Hermes prevents all violations. Each point of the Figure 6.9 represents the number of SLA violations for intervals

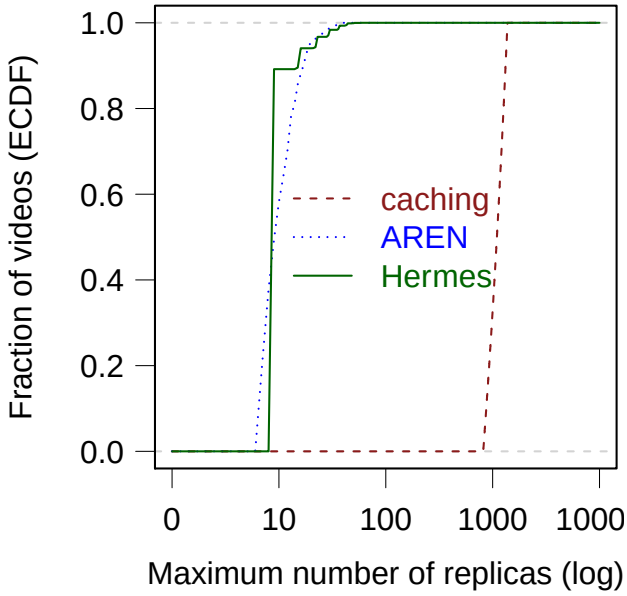

Figure 6.7: The maximum number of replicas for the 1% most popular contents.

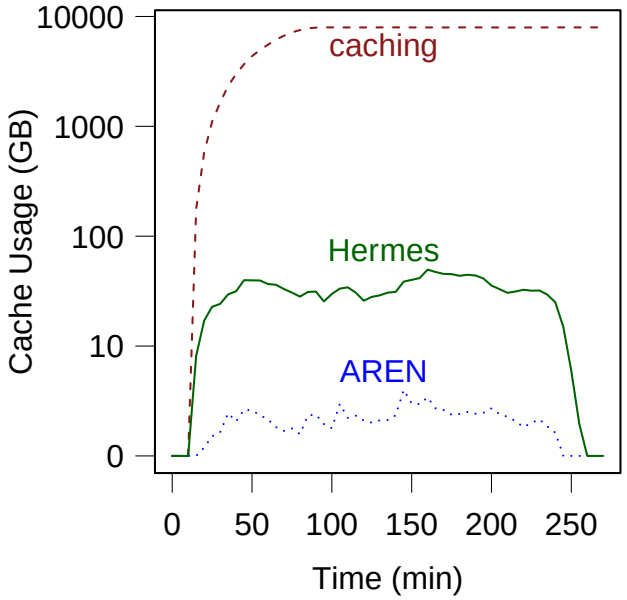

Figure 6.8: Storage usage for replication.

of five minutes. Overall, caching caused 1569 violations affecting almost one third of all viewers, AREN and Hermes had none. As AREN, Hermes prevents violations by (i) creating new copies for popular contents only, and (ii) adapting the number of replicas properly. Vertical lines in Figure 6.9 represent the first access to the three popular contents with the worst content provision

through caching. They account for 96.81% of all caching violations. The appearance of these contents puts the system under heavy load, which makes caching fails to prevent violations.

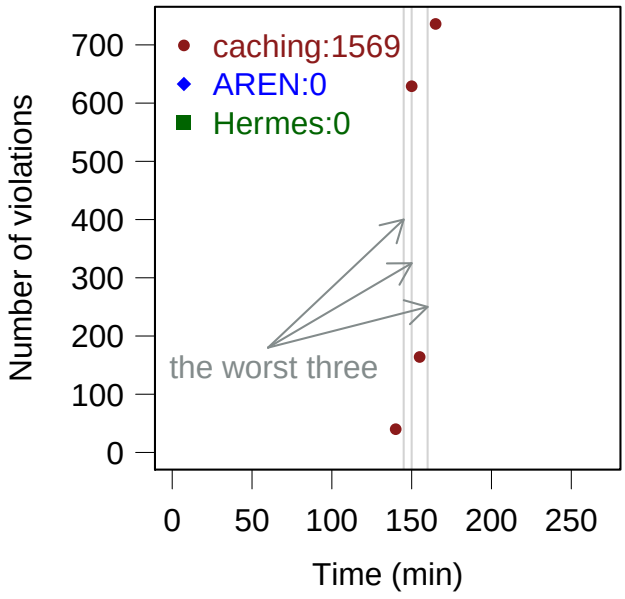

Figure 6.9: SLA violations. Vertical lines show when happened the first access to three contents with the worst content provision using caching.

Figure 6.10 depicts the average bitrate for viewers of the three contents with the worst content provision using caching. When caching was under heavy load, half of viewers experienced a very low bitrate, raging between 460Kbps and 4860Kbps. The mean bitrate with caching was 45Mbps. On average, Hermes improved this bitrate by roughly 90% under heavy load. AREN comes just behind, improving bitrate provision by 87%. This find suggests that Hermes largely outperforms caching, and provides still better than AREN under heavy load conditions.

6.5.6 Delivering Highly Available Videos with WiseReplica

The utmost goal of the performance evaluation of WiseReplica is two-fold: (i) measure the accuracy of our learning model in ranking Internet videos in order of *hotness*, and (ii) evaluate the performance of our replication scheme in meeting viewers' expectations properly. Further details about evaluation set-up are available in Subsection 6.5.2.

6.5.6.1 Performance Evaluation Metrics

We aim to evaluate the performance of two main WiseReplica modules: machine-learned ranking and replication strategy. Hence we group evaluation metrics as follows: machine-learned ranking accuracy and metrics for replication strategies in VoD systems.

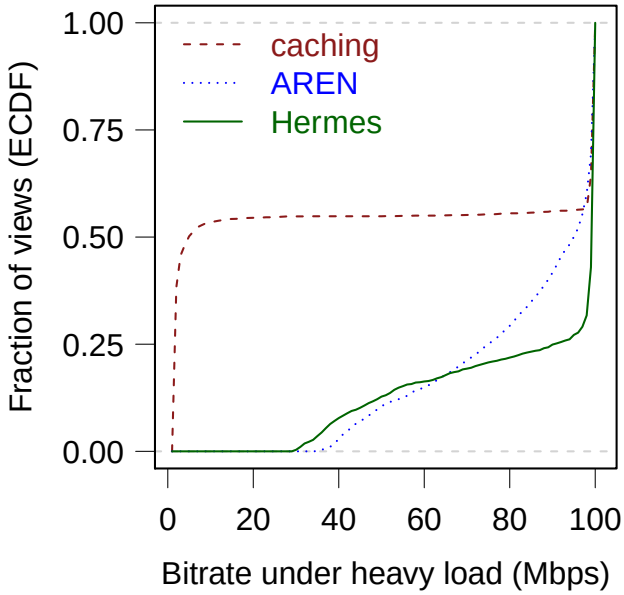

Figure 6.10: Bitrate for viewers of the three most popular contents under heavy load.

Machine-Learned Ranking Accuracy. We adopt the normalized Discounted Cumulative Gain (nDCG) criterion as the main evaluation metric for our learning model. nDCG is a standard quality measure in information retrieval for ranking problems, especially for Web search [72]. The DCG definition is:

$$DCG_L = \sum_{i=1}^{L} \frac{2^{F(i)} - 1}{\log_2(1+i)}$$

where L is the global set of ranked videos, and $F(i)$ is the rank position of ith video. To compute nDCG, we divide DCG measure by the idealized DCG with perfect order of the set L. Thus, the perfect model scores 1. Unlike typical information retrieval problems, as a ranking of web content, our model does not have the notion of *query*. Instead, we rely on nDCG robustness to measure the performance of our learning model as a global ranking problem. Since the ranking problem shares properties with both classification and regression problems, we compare nDCG to other three popular machine learning metrics: the mean square error, a standard metric for regressions; precision, for classification; and a less robust, well-known variant of nDCG, namely in this work nDCG(2). We evaluate three different state-of-the-art ensemble learning methods available in Scikit-learn library: Random Forest, Extremely Randomized Trees, and Gradient Tree Boosting. Moreover, we report briefly on the sample size for learning, number of estimators or learners of ensemble methods, measurements or features importance, and the computational overhead of our model, including memory usage and computation time for prediction.

Metrics for Replication Strategies in VoD Systems. Assuming that content and CDN providers are committed to enforcing bitrate as main QoS metric through SLA contracts, we consider SLA violation as the primary performance metric. Thus, a SLA violation happens whenever the VoD

system does not provide the minimum average bitrate for preventing rebuffering. This measures the capacity of WiseReplica capacity to meet consumers' expectations. We also investigate the impact of our replication scheme using storage domains in VoD systems. To this end, our evaluation metrics are network and storage usage. Finally, we compare WiseReplica results with a non-collaborative caching and the AREN, described in Chapter 5.

6.5.6.2 Fitting and Measuring the Accuracy of Our Ranking Model

The evaluation of our learning model comprises: ensemble method selection, number of *estimators*, sample size for learning, and inputs' relative importance. In this subsection, we aim to evaluate the most important settings and tune our model towards higher accuracy, using the learning framework described in Subsection 6.2.3.

Selecting and Fitting an Ensemble Method. Ensemble methods have become very popular in statistical learning. Their algorithms combine several *estimators* or *week learners* to provide robust learning models and prevent overfitting. We fit and evaluate our model with three methods from Scikit-learn library: Random Forest(RF), Extremely Randomized Trees(ET), and Gradient Tree Boosting(GB). We consider two distinct samples with 124 thousand lines each, one for training and other for testing. We set to 10 the number of estimators as a common setting. All other parameters have default settings. Based on four metrics detailed on Subsection 6.4.2, Random Forest fits our model better. Figure 6.11 depicts three of these metrics. Random Forest performs particularly well in nDCG score, the main metric for ranking problems. While Extremely Randomized Trees and Gradient Tree Boosting score 0.9126 and 0.4128 respectively, Random Forest scores 0.9594. In terms of precision, Random Forest slightly better, with a score of 0.9922. Extremely Randomized Trees scores 0.9899, and Gradient Tree Boosting scores 0.9502. It also outperforms the other two methods regarding the mean square error metric, scoring 0.0094 compared to 0.0122 with Extremely Randomized Trees and 0.1021 with Gradient Tree Boosting. nDCG(2) metric confirms these results. Therefore, we select Random Forest method for our ranking model and nDCG as the key accuracy metric.

Adjusting the Number of *Estimators* to Learn. According to Friedman *et al.*, Random Forest performs predictions by building a collection of *de-correlated* trees, namely estimators, and then averages them. We investigated the impact of the number of estimators in ranking accuracy, memory and computation time. We varied the number of estimators progressively from 10 to 1000, with the same previous samples. Results show that the number of estimators has a negligible impact in the accuracy of our model. While a model with 10 estimators scores 0.9594, 1000 scores 0.9569, slightly worse. One reason for this might be the number of inputs, relatively small, that is likely to require a small number of estimators. Yet, the number of estimators impacts on the model overhead, specially for computation time. As depicted in Figure 6.12, computation time ranges from 0.3 microseconds with 10 estimators to almost 26 microseconds with 1000 ones. Although the worst case still represents low overhead, the lower the better. Memory overhead is rather negligible, ranging from 30 to 32MB. Overall, our model has a quite low overhead, suitable for going online. Since there is no evidence to increase the number of estimators, we keep 10 estimators as a default, fair setting.

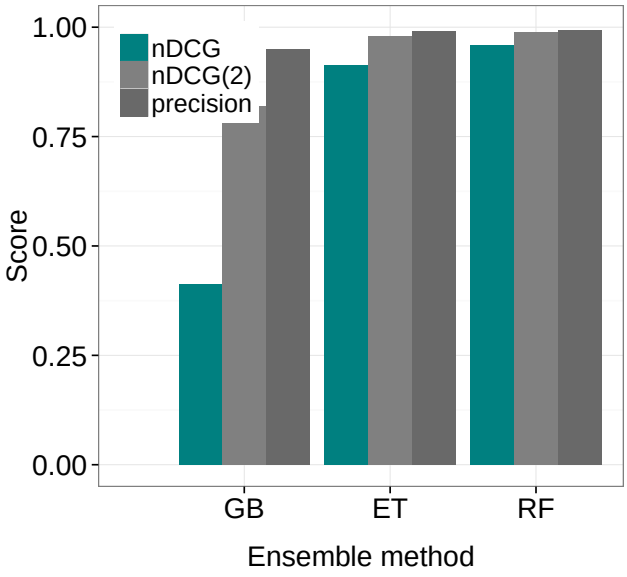

Figure 6.11: Ensemble methods evaluation: Random Forest(RF), Gradient Tree Boosting(GB) and Extremely Randomized Trees(ET).

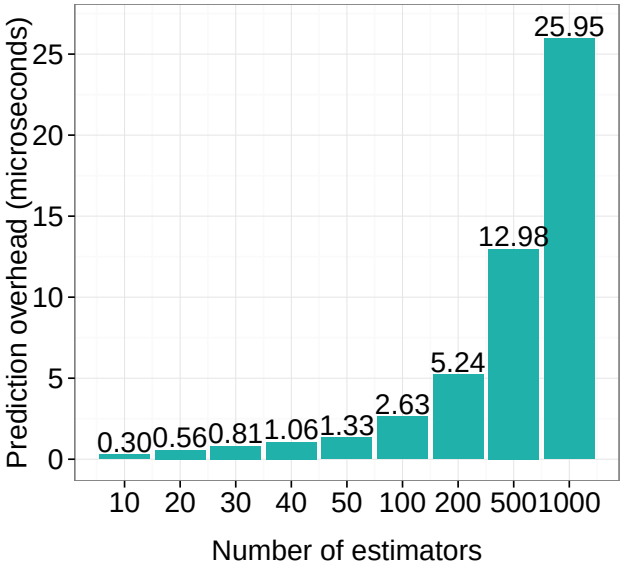

Figure 6.12: Overhead for different number of estimators of Random Forest.

Evaluating Bigger Samples for Fitting the Model. Towards a higher accuracy, we evaluate bigger samples for fitting the model. We collected more information by running longer simulations. As expected, Figure 6.13 confirms that we improve accuracy through bigger samples.

The improvement in accuracy was slight, about 0.03 as we use a sample size almost six times bigger, i.e. 683 thousand. It is quite important to highlight, though, that this has no impact on computation time of predictions. Thus, we use the biggest sample for the remaining evaluations.

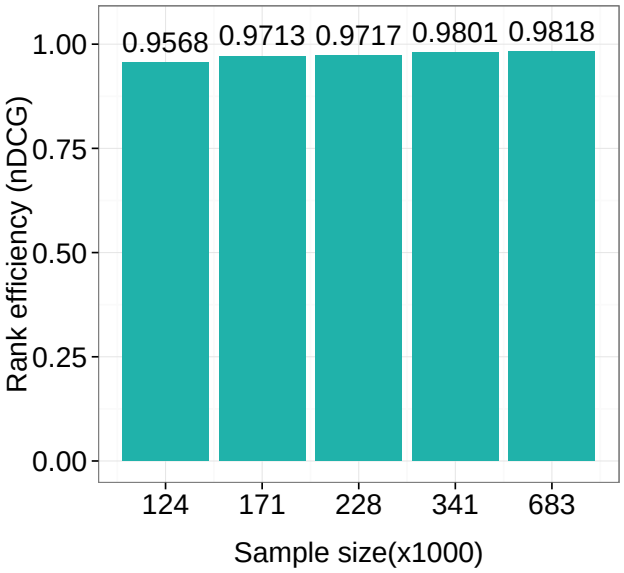

Figure 6.13: Accuracy with different sample sizes.

Analysing the Relative Importance of Model's Inputs. We were particularly interested in evaluating the contribution of each input of our model, described in Subsection 6.3.1. Scikit-learn library allows to measure the relative importance of each input for predicting the ranking position using the Random Forest method. Figure 6.14 highlights the relative importance for all 10 inputs of our ranking model. The two most relevant inputs are the current number of viewers and network availability. These inputs alone account for 99.6% of the all model's accuracy. It seems quite reasonable, since the former measures the demand for a video and the later depicts the offer of network resources, the main system feature for enforcing average bitrate. The remaining eight inputs are largely irrelevant to the ranking model's accuracy, and may be replaced or removed. Surprisingly, the number of replicas, current network load, and video size seem to be useless to our model. It is likely that network availability is a particularly good measurement, making these eight inputs rather redundant. For simplicity, we include all inputs in the rest of the work. This is harmless for the model's accuracy.

6.5.6.3 Evaluating Replication Strategies in VoD Systems

In this subsection we analyse the replication strategy used in WiseReplica. First, we evaluate four simple replication policies. Then, we compare WiseReplica with a non-collaborative caching and AREN, all described in Subsection 6.5.3. We evaluate their capacity to meet consumers' expectation by observing the number of violations. In addition, we analyse network and storage usage.

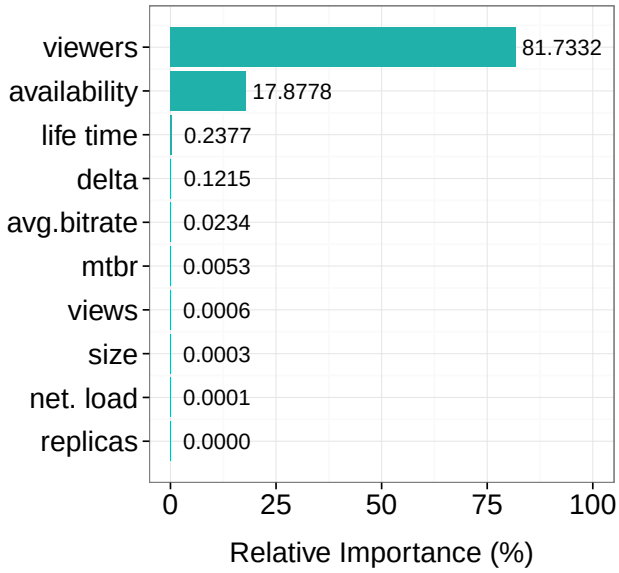

Figure 6.14: Relative importance to ranking of the 10 model's inputs.

Enforcing Simple Replication Policies on Ranked VoD. For the three highest rank position, WiseReplica enforces a replica creation policy, described in Subsection 6.4.2. It defines the replication degree growth factor. Considering mean video size of 20MB, we analyse four simple creation policies, namely uniform, linear, quadratic, and exponential. Table 6.5 show the number of violations by varying B from 2 to 6. Overall, creation policies that take into account the rank positions, i.e. linear, quadratic, and exponential, performed better. Results show that there is relatively small difference for $B \geq 3$, suggesting that our ranking model reacts promptly to modifications on network availability, preventing over-replication. However, for $B \geq 5$, it appears that replication increases the network load system load, causing few more violations. We selected the linear policy with $B = 4$ that seems to be the most resilient towards proper resource allocation, providing a fair replication degree growth factor.

Table 6.5: Replication policies.

	Parameter c				
Policy	**2**	**3**	**4**	**5**	**6**
Uniform	867	567	44	28	23
Linear	123	77	6	9	16
Quadratic	102	21	42	46	58
Exponential	118	32	19	27	28

Load Resiliency. A good replication strategy must cope with changes on the system load. We vary the global load of the system by changing the mean video size, described in Subsection 6.5.1.

Assuming the three mean video sizes 20MB, 30MB and 40MB, caching had 1814, 3864, and 7049 violations respectively, while WiseReplica had only 6, 77, and 106. Figure 6.15 compares the number of violations using WiseReplica and a non-collaborative caching. As the load of the system increases, concurrency in bitrate allocation also increases, causing more violations. WiseReplica outperforms caching mostly because it predicts and prevents useless replication. Therefore, we set 40MB as the default mean video size workload setting.

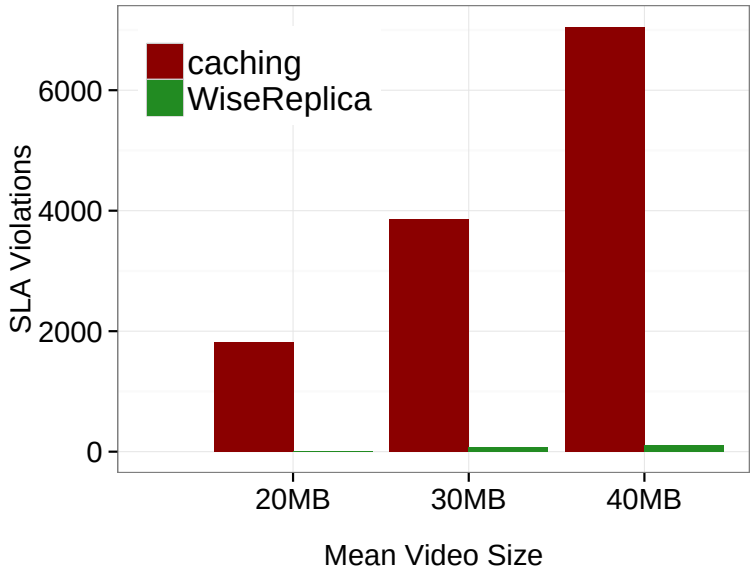

Figure 6.15: Mean Video Size. Higher loads increase the concurrence in network resources, as a result, more violations.

Benefits of Prediction on Storage Usage. We aim to adapt the number of replicas to the number of views of a video, especially for the most popular ones. Figure 6.16 plots the maximum number of replicas for the 1% most popular videos. Using caching, the maximum number of replicas is high, ranging from 816 to 1367. The AREN allows to decrease significantly the lower and upper limits, to 10 and 190. WiseReplica also reduces the maximum replica range, which is from 19 to 160. More interestingly, the shape of the replication curves of WiseReplica and AREN are quite similar indeed. It confirms that our predictions are accurate, and that a simple replication policy works properly.

Reducing the number of replicas implies that the systems requires less storage for replication. Figure 6.17 shows storage usage for replicas by replication scheme. Although WiseReplica utilizes more storage than AREN, its usage remains two orders of magnitude smaller than a non-collaborative caching. The maximum storage usage for AREN, WiseReplica, and a non-collaborative caching were 34, 85, and 7921 GB respectively. WiseReplica creates more replicas than AREN because it does not rely on bandwidth reservation to prevent violations. Despite that, WiseReplica maintains replicas efficiently, keeping storage usage very low, and making cache

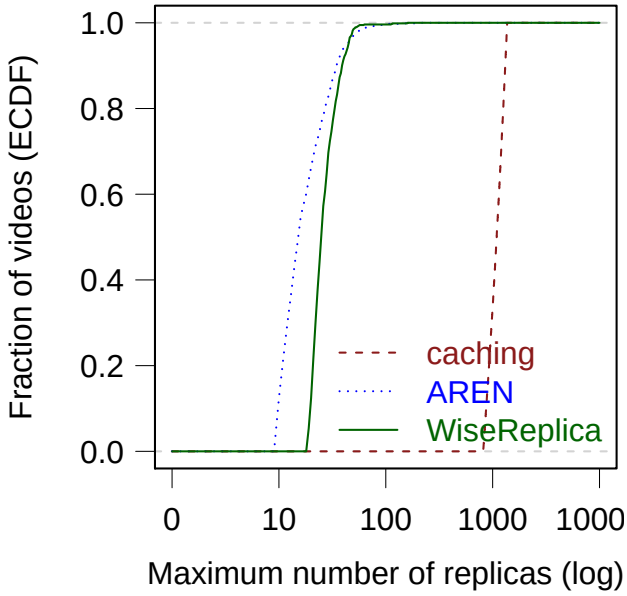

Figure 6.16: The maximum number of replicas for the 1% most popular videos.

replacement policies redundant.

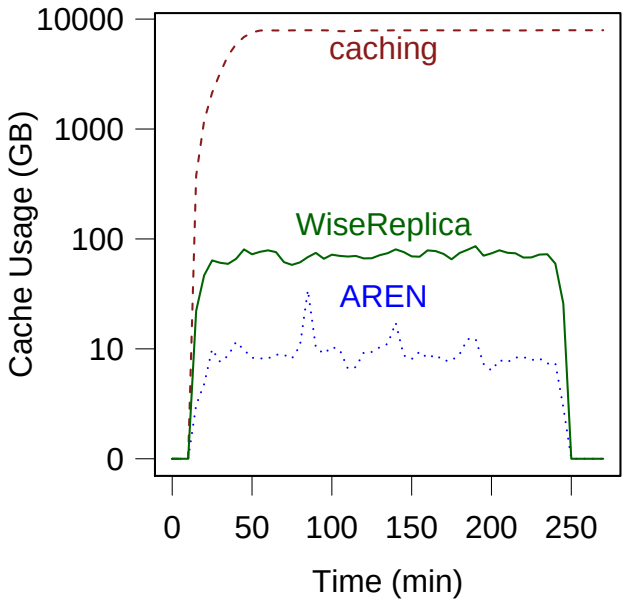

Figure 6.17: Storage usage for replication.

Enhancing Bitrate Provision for Meeting Consumers' Expectation. WiseReplica performance is also quite similar to AREN regarding preventing violations. Each point of the Figure 6.18

represents the number of SLA violations for intervals of five minutes. Overall, caching caused 7049 violations affecting 86% of all viewers, WiseReplica had just 106 violations, and AREN, evidently, none. Compared to caching, WiseReplica prevents nearly 99% of violations. It copes with violations by (i) creating new replicas for hot videos only, and (ii) adapting the number of replicas according to the rank position. Vertical lines in Figure 6.18 represent the first access to the 10 videos with the worst content provision through caching. They account for 80.62% of all caching violations. The appearance of these videos puts the system under heavy load, which makes caching fail to prevent violations.

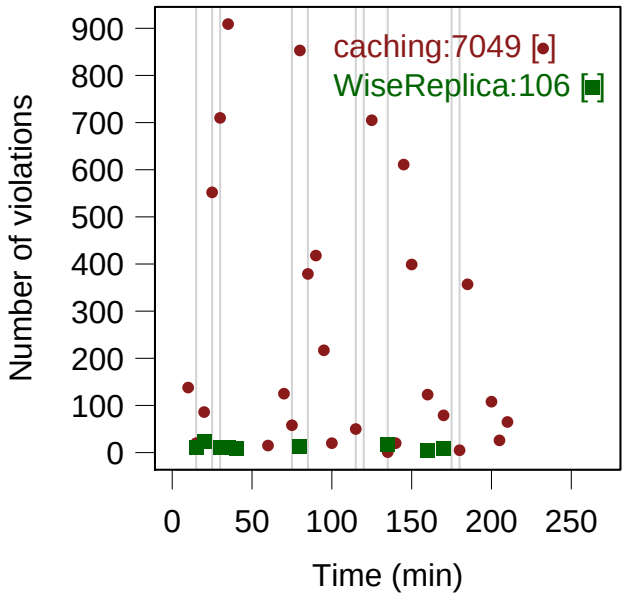

Figure 6.18: SLA violations. Vertical lines highlight the first view to 10 videos with the worst content provision using caching.

Figure 6.19 depicts the average bitrate for viewers of the 10 videos with the worst content provision using caching. When caching was under heavy load, half of viewers experienced a very low bitrate, ranging between 230Kbps and 2575Kbps. The mean bitrate with caching was 43Mbps. On average, WiseReplica improved this bitrate by roughly 85% under heavy load. Actually it performs almost as well as the AREN assumption, that improved bitrate provision by 93%. These finds suggest that WiseReplica largely outperforms caching, fairly meeting consumers' expectations under heavy load conditions.

6.6 Conclusion

In this work, we presented Hermes, an adaptive replication scheme for offering highly available Internet videos on hybrid CDNs. To adapt replication, we proposed a learning model that tracks

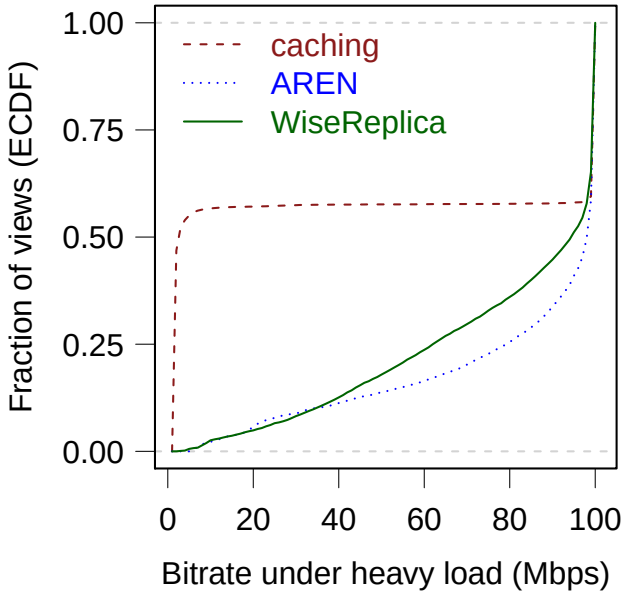

Figure 6.19: Bitrate for viewers of the 10 most popular videos under heavy load.

popularity growth curves based on lightweight measurements of the request arrival process. Simulations with YouTube traces showed that our predictions are accurate. That allowed Hermes to maintain the replication degree of Internet videos properly. Our evaluation results highlight that Hermes increases the average birate provision by roughly 90%, contributing decisively to enhance viewing experience of users. One interesting perspective on evaluating Hermes would be to develop a proof-of-concept prototype in a real testbed.

We provide two major contributions in this chapter. First, we present a SLA-based, adaptive replication scheme for meeting customers' expectations and enhancing resource allocation in hybrid CDNs. Simulations with YouTube suggest that our approach is quite efficient. Second, we propose a two accurate statistical learning models based on lightweight measurements of the request arrival process for boosting Internet content availability.

We designed Hermes for adapting replication of generic Internet contents. Its design is based on a learning model that allows us to predict the popularity of web contents for controlling contents' replication degree according to customer-oriented SLA. We propose a more specialized, tough quite interesting, approach in WiseReplica. In its design, we turned our attention to enhance the availability of Internet streaming videos. Thus, we assume that Internet Service Providers (ISPs) and content providers are committed to meet overall customer expectations on video-on-demand services through SLA contracts. In this case, the goal is to prevent rebuffering by allocating enough replicas of videos in peers of edge networks. WiseReplica cope with this problem by ranking accurately videos in order of *hotness*, and then adapting replication in a proactive way. Overall, Hermes and WiseReplica almost doubled the availability of contents. On average, network provision was improved by 87%, and storage usage for caching reduced by two orders of magnitude.

The performance of these replication schemes relies mainly on the robustness and accuracy of our statistical learning models. In addition, they are quite flexible, and can learn from different sources and big amounts of data, providing a robust framework for controlling Internet resource allocation. They provide an efficient and fast way of adapting resource allocation that goes online. One can easily extend or adapt our models for other scenarios or performance goals. Simulations showed that our predictions are very accurate indeed. in a real testbed.

Chapter 7

Conclusion

Contents

7.1 Conclusion

This dissertation developed practical methods for content delivery networks (CDNs) to enhance content availability for end customers through adaptive replication schemes. We introduced Caju, a general-purpose content distribution system, which handles resource allocation in edge networks, and allows Internet Service Providers (ISP) to interoperate with existing infrastructure-based CDNs in a hybrid design. Considering Caju's functionalities, we presented AREN, an Adaptive Replication scheme for Edge Networks, which enhances content availability according to QoS metrics. AREN provides highly available contents using network resources' reservation. It adapts the replication degree of contents in order to prevent violations, and reduce network and storage utilization.

We also placed these results in a broader context of predicting the demand for Internet contents and self-adapting their replication based on lightweight measurements from CDN's monitoring. Using statistical learning methods, we designed and evaluated a flexible framework for learning *good* resource allocation's behaviours in a batch mode. We showed that our framework is easy to use, predicts content replication on the customers' request basis, and goes online with negligible computational overhead. We extended our approach to provide highly available Internet videos, avoiding re-buffering, and meeting customers' expectations of video-on-demand services.

In summary, we believe that this dissertation provides strong arguments that content availability is one of the most important performance issues in CDNs, and particularly that hybrid

designs and adaptive resource allocation methods can contribute to improve customers' experience of getting Internet contents.

7.2 Limitations and Perspectives

Our techniques and results are not without limitations. In this final section, we present and discuss some of these thesis issues and perspectives that they suggest.

Adaptive replication schemes for hybrid, mobile CDNs. We extensively evaluated our replication schemes in edge networks composed of wired devices, such as set-top boxes or home gateways. But, as there exists an increasing availability of wireless devices with non-negligible network and storage capacities, including smartphones, tablets, and vehicular networks, it seems that a hybrid CDN design must include also mobile resources. In this context, we would be interested in evaluating the challenges of designing and evaluating adaptive replication schemes in mobile edge networks.

Flexible content delivery systems. We showed in Chapter 5 Caju, a general-purpose content distribution system for fixed edge networks. Caju was initially designed to operate in ISP devices, such as home gateways, whose communication may happen using P2P paradigm. Although we suggested in Chapter 6 that Caju's design could be extended to peer-assisted VoD systems, it remains unclear in Caju's design how to deal with P2P communication and particularly with monitoring of P2P nodes. It would be interesting to enhance Caju's design in order to provide further useful technical details to assist in the deployment of more specific content delivery systems as peer-assisted VoD systems.

Cloud services and smarter Service Level Agreement Contracts. In our dissertation, we consider two types of SLA contracts. First, considering the Hermes replication scheme defined in Chapter 6, we assume that CDN providers are committed to enforce SLA contracts assigned to customers. In this case, customers define which level of QoS they are interested in. In the second type of SLA contract, as evaluated in WiseReplica in Chapter 6, content providers define which level of QoS must be enforced by the CDN providers of any customer. In this second case, content and CDN providers are committed to enforce QoS metrics for all customers. We have likely under-explored the potential of SLA-based resource allocation to offer cloud services. We could improve our contributions in terms of cloud-oriented services by exploring better the elasticity of CDN infrastructures and performing further investigation into SLA definitions. For example, we could vary the number of SLA categories of Chapter 5, or consider other content-oriented SLA definitions in Chapter 6.

Realistic evaluation of adaptive replication schemes with a realistic environment. Our promising finds with Hermes and WiseReplica in Chapter 6 show that we can efficiently adapt content replication degree based on a number of CDN metrics. They show that replication schemes are able to promptly react to changes on the system load, reducing resource utilization and meeting SLA constraints. However, we get all results from simulations, where the reproduction of realistic system behaviours, such as churns and variations of network availability, were out of the scope of our study.

One of the most important perspectives of this work is to investigate the performance of these replication schemes by carrying out experiments in a testbed. This would allow us to validate our approach and find new research paths. In order to speed up the development cycle and deployment process, we could use BitTorrent [31] as the content distribution protocol. The aim is to make BitTorrent behaves as a reliable content distribution system based on Caju design. In this case, Hermes or WiseReplica would play a key role in controlling the number of available sources according to the *swarm* [1] size.

To reinforce our work as an important issue of content distribution networks, we could perform a deployment on PlanetLab [100]. PlanetLab would provide realistic system constraints, such as high latencies and unpredictable resources availability, which would permit us to have effective insights into our problem on a globally distributed content delivery environment.

Trances and datasets for learning better content delivery behaviours. In Chapter 6, we design and evaluate two statistical learning models that allow us to predict the demand for contents. Evaluations with state-of-the-art statistical learning metrics, specially precision and nDCG [72], suggest that our predictions are very accurate, allowing us to fairly capture the behaviours of our training data sets. But since both training and test datasets for predictions come from simulations with replication scheme AREN only, the learning task may result in biased predictions. This is likely behind the unexpected results of Section 6.5.6.2.

To overcome this issue, we could validate our learning models with different datasets. New data might come from measurements of AREN real deployment or data collected from another content delivery experiments, e.g. by using BitTorrent [31]. For data from another content delivery experiment, we should design a method to assign labels to data in order to represent popularity classes and ranking.

[1] In BitTorrent, swarm is a set of peers downloading the same content.

Appendix A

Designing and Evaluating an Accurate Bandwidth Scheduler for the PeerSim Simulator

Contents

A.1 Introduction

Cloud users expect that emerging Internet services could provide outstanding performance guarantees, e.g. enforcing deadlines for data transfers through SLA contracts. Unlike fair-share scheduling, where bandwidth is equally distributed among concurrent transfers, a deadline-aware approach, such as D3 [141], uses explicit rate control to apportion bandwidth according to the flow deadline to prevent SLA violations. In order to study the performance evaluation of such applications, we designed a PeerSim component for bandwidth scheduler that permitted us to sim-

ulate deadline-aware applications. PeerSim [88] is a highly scalable Peer-to-Peer (P2P) simulator. We have been particularly interested in PeerSim's modularity and user-friendly API. Our component allows system analysts to easily define and run simulations where accurate fair-sharing bandwidth and strict rate control enforcement are key issues. It is suitable for simulating cloud applications that require a deadline-aware control protocol on top of P2P networks. We have designed and implemented an accurate and lightweight mechanism for enforcing fair-sharing policy on data transfers and strict rate control, whenever it is required. It does not reproduce the complexity of in-depth transport protocol behaviour. Instead, we focus on proper dynamics of bandwidth sharing based on a connection-oriented approach. It allows us to apportion bandwidth resources on cloud environments. Like PeerSim, our component is simple to use and can be easily customised for many different set-ups.

This appendix is organised as follows. In Section A.2, we describe the design of our component in details by explaining its main functionalities and implemented protocol layers on PeerSim. Then we evaluate the performance of four different bandwidth scheduling approaches in Section A.3. A comparative study of these approaches and some final discussions are presented in Section A.4. Finally, Section A.5 concludes.

A.2 Component Design

In this section we outline our approach for simulating deadline-aware applications. We present the design of our PeerSim component, detailing its protocol layers, main modules, and functionalities.

We have implemented a modular component for simulating deadline-aware cloud application on top of the PeerSim simulator, whose sources and documentation are available online [1]. Figure A.1 shows the layers of our component and their interactions with PeerSim. Configurations are made through the main PeerSim API, ensuring usability.

Our component is composed by three layers: Network, Transport, and Application. Each layer provides an interface for the upper layer and allows developers to easily add additional functionalities. For simplicity, we have selected the PeerSim node, with identifier zero, for being a special node in our PeerSim component. It is called coordinator. It stores Global structures that are essential for consistency and state of the on-going simulation, such as addresses mapping and full connections' state.

A Monitoring module was implemented to provide periodical information about the state of nodes. It tracks data transfers information such as number of accomplished flows, instantaneous number of bits sent, bandwidth usage, active connections, and so on.

In this Section, a data communication between pair of nodes is named according to the layer level as connection, flow, and transfer for Network, Transport, and Application layer respectively. The following Subsections describe the main functionalities of each layer in details.

[1] The The PeerSim P2P Simulator. http://peersim.sourceforge.net, August 2013.

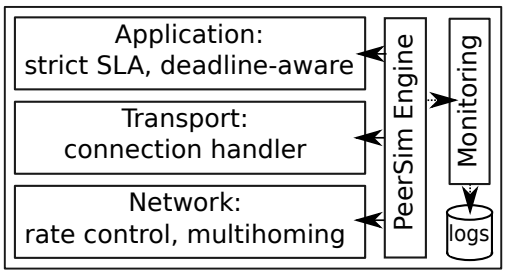

Figure A.1: A modular PeerSim component for simulating deadline-aware applications.

A.2.1 Network Layer

Network layer implements a lightweight connection-oriented bandwidth scheduler and offers two essential bandwidth mechanisms for deadline-aware applications: fair-share and strict rate control.

Accurate fair-share scheduling of bandwidth permits apportioning network resources equally and dynamically throughout active connections. It simulates the common behaviour of communications between nodes without traffic priorities. Our evaluations focus on the accuracy of this functionality in this work.

Enforcing strict rate control is the main goal of our component. It aims to provide a precise rate control mechanism for enforcing strict SLA contracts. A deadline is attributed to each connection according to the upper layer requested rate. Deadline is the maximum amount of time, in milliseconds precision, for a connection ends. During the bandwidth allocation for connections, each node verifies the requested rate and reserve bandwidth properly. Towards accurate bandwidth allocation that captures the main aspects of network dynamics, we consider that correct fair-share of bandwidth is applied for non-reserved resources, as depicted in Figure A.2.

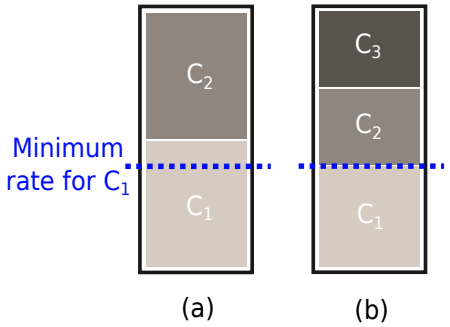

Figure A.2: Rate control enforcement and fair-sharing scheduling dynamics.

Figure A.2 (a) shows that a minimum rate of 40% of the bandwidth is enforced in connection C_1 that competes with connection C_2 for network resources. Since further resources are available, C_1 would be able to have more than required rate, finishing earlier than expected. In this particular case, fair-share scheduling ensures 50% of the bandwidth for C_2. In Figure A.2 (b), we consider,

in the meanwhile, the arrival of a third connection C_3. C_2 and C_3 do not require a minimum rate. As a result, Network layer enforces the minimum rate for C_1, that is equal to 40% in this example, and applies fair-share for the remaining resources, 60% of the bandwidth, throughout C_2 and C_3. It allows us to accurately reproduce network dynamics with rate control enforcement. Further insights into deadline-aware approach used in this work are available on Wilson *et al.* work [141].

We have also designed a multihoming scheme as part of the Network layer. Our goal is to allow users to easily define cloud-like scenarios, from machine-network virtualization to datacenter environments. Figure A.3 shows the main blocks of our multihoming network model. We have defined a Broadcast Domain (BD) as an essential block of our model. In a nutshell, it models isolated portions of PeerSim node's bandwidth. BD_0, or host domain, represents the whole bandwidth available on a node, which might simulate a datacenter uplink as well as a simple host's network interface card. Then multihoming takes place by adding additional in-built BDs, with indexes greater or equal to one, so-called guest domains. Thanks to a maximum guest domain bandwidth limit enforcement, any bandwidth values might be freely and independently assigned to guest domains. Last but not least, this module keeps and exports an address table, including BD to global pseudo-network address mapping, to ease the utilization and provide transparency of multihoming network functionality. We use this simulator for performing all experiments of our thesis. Evaluations with deadline enforcement and multihoming functionalities are available in Chapters 5 and chap:learning.

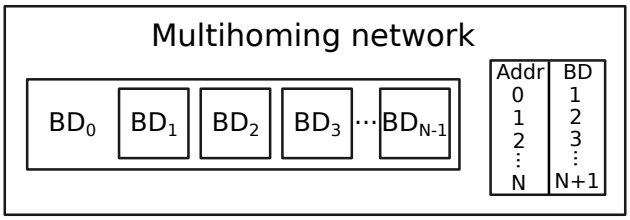

Figure A.3: A multihoming network model for cloud-like simulations.

A.2.2 Transport Layer

We propose a Transport layer to interface with our deadline-aware Network layer. We assume that data transfers are handled between pair of nodes as connection-oriented data streams, called flows. It provides two main services for upper layers: connection management and accountability.

Flow management allows us to have full control of Network connections, including addresses' mapping for multihoming, creation, and deletion operations. Whenever a new node from an upper layer is inserted in the simulation, the Transport layer handles the mapping between Application and Network addresses. There is an single instance of this mapping structure per simulation in the coordinator node. It ensures mapping consistency and proper node selection in multihoming scenarios. Transport layer provides an easy way to create flows with a couple of parameters. It stores connections identifiers returned by the Network layers for accountability or deletions. A

wide range of flow information and statistics are available, e.g. instantaneous and precise number of bits already sent can be easily retrieved from Network.

A.2.3 Application Layer

This layer provides the main interface for running deadline-aware simulations. Many logical nodes may be hosted in a single Application layer in cloud-like scenario. So that, in a single PeerSim node, we are able to easily simulate multiple logical nodes, such as multiple virtual machines or even an entire datacenter. To each logical node is assigned an application-level address, which is mapped to network-level address through Transport layer mapping. This allows users to define and to operate multihoming properly.

Along with few definitions in the main PeerSim configuration file, the Application layer requires a CSV input file with SLA contracts' definitions. Our current design allows us to define and use multiple SLA contracts. A SLA contract defines a strict transfer rate for a class of nodes. Then SLA contracts should be assigned to nodes accordingly.

Its functioning is straightforward. Application events or logical nodes' transfer requests are sent to Transport layer that interacts with Network for performing a transfer. Its behaviour depends on how the Network layer is operating. Considering a Network layer applying just fair-share of bandwidth, an event is always treated, and when the transfer finishes, Transport layer notifies Application. Then Application checks if the transfer meets its respective SLA contract on behalf of the requester. On the other hand, if Network is configured to enforce rate control from SLA contracts, before starting the transfer, an admission control process takes place for verifying if there is enough network resources in both source and destination Networks for fulfilling the Application request. If there is no enough spare resources, Network raises a connections failure message, and requester's Application layer is notified.

A.3 Comparative Evaluations of Bandwidth Scheduling on PeerSim

In this Section, we aim to evaluate the performance and measure the accuracy of different bandwidth scheduling approaches in order to provide deadline-aware data transfer with strict rate control. We have studied four scheduling approaches for bandwidth allocation on top of PeerSim simulator: the simplest bandwidth scheduler, a lock-based bandwidth scheduler, a packet-based bandwidth scheduler, and a connection-oriented bandwidth scheduler.

We have defined an evaluation scenario with 1000 nodes. For simplicity, we assume a fully meshed and connected network topology, in which each node is equipped with a full-duplex 10Mbps link. We simulate data transfers in a content distribution network where each node plays a distinct role of either content provider or consumer. Content consumers randomly select a source per request for downloading. We consider that content size follows a bounded Pareto distribution that ranges from 1MB to 1GB. We simulate one hour of content transfer. Our primary factors for performance analysis are:

- Number of content sources: that is the number of nodes that play the content provider role, whose uplink bandwidth is shared by consumers' downloads for content distribution.
- Content mean size: by changing the shape parameter of Pareto distribution, we have been able to evaluate bandwidth allocation performance with different content average sizes.
- Degree of parallelism: by degree of parallelism we mean the number of simultaneous active downloads performed by a content consumer node. During the simulations bootstrap, a number of parallel downloads is launched from each consumer at the same time, according to the degree of parallelism. When a download is accomplished, a new download is immediately performed in order to keep the degree of parallelism.

The evaluation of our simulations towards deadline-aware transfers have been measured by three simple metrics: average uplink bandwidth usage of nodes playing content provider role, average memory usage, and average computation time per content transfer. We have performed our simulations using server with an Intel Xeon E5450 3.00 GHz, and a RAM of 4GB. We describe and evaluate each approach in the following Subsections.

A.3.1 The Simplest Bandwidth Scheduler

Here, we describe an easy way to implement a bandwidth scheduler for data transfer simulations. In order to transfer data between two nodes, the source node computes the bandwidth available to the destination by simply selecting the smallest bandwidth value between the two nodes, and then computing the duration of the data transfer based on the requested content size. This scheduler provides a static bandwidth allocation between the source and destination, that extremely speeds up simulation.

Assuming an evaluation scenario where each content consumer does not perform parallel downloads, that means degree of parallelism equals to one, and a content mean size of 8 MB, we have varied the number of content sources or providers from 10 to 50 % of the total of nodes. Figure A.4 shows the average uplink bandwidth usage per content provider.

For this scenario, our implementation requires about 9.2 MB of memory on average, and the average computation time is only 0.4 microseconds per message sent. Although the simplest scheduler consumes very few computational resources, it provides highly imprecise bandwidth usage results as the number of sources decreases, and concurrent content accesses occur. As expected, the maximum average is reached when 50% of nodes play the role of content provider. This evaluation metric soars to 90Mbps when 10% of nodes are content providers because there is no bandwidth sharing policy for concurrent transfers on sources, generating highly inconsistent results.

A.3.2 A Lock-based Bandwidth Scheduler

We enhanced the previous bandwidth scheduler in order to improve bandwidth allocation precision without much increasing to computational resources consumption. We implemented a

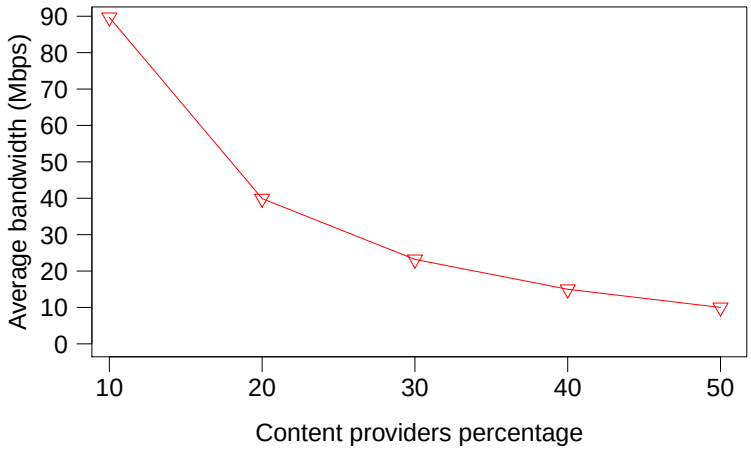

Figure A.4: Average uplink bandwidth usage of content providers for different number of content sources.

lock-based mechanism that prevents overlap of bandwidth consumptions when there are more consumers than sources. Its behaviour is quite similar to the previous approach. But, instead of starting a new transfer whenever it is requested, the content provider verifies the availability of the whole bandwidth in both source and destination. If at least one of them has already started a new data transfer, the request is queued on the content source, and FIFO policy is enforced. This simple improvement permits to avoid over-consuming in bandwidth usage of nodes. However, it undermines bandwidth allocation efficiency. Considering the worst case of the previous approach, where 10% of nodes are content providers, we show in Figure A.5 what happens with average bandwidth usage on content providers when content mean size changes. Average bandwidth usage falls sharply from 8.5 to 5.1 Mbps when mean content size is multiplied by 4. That causes an increasing amount of idle bandwidth resources despite the higher load, simulating inaccurate bandwidth apportion.

The performance of bandwidth allocation for this approach is optimal when data transfers have the same length, e.g. considering a uniform distribution for content size. Whatever the workload, parallel transfers can not be simulated properly.

A.3.3 A Packet-based Bandwidth Scheduler

This approach was based on the PeerSim bandwidth manager module proposed by Russo *et al.* [112]. In order to simulate parallel transfers and bandwidth sharing with low computational resources consumption, they introduced a bandwidth scheduler based on slots and priority sharing policy. They assume whenever a source node connect to a destination to transfer data, it allocates a bandwidth's slot for an amount of time, depending on the data size. They consider that uplink and downlink are asymmetric, with downlink greater than uplinks. In general, it permits that multiple uplink slots match into a single downlink, providing transfer parallelism with high performance. Although it is highly configurable, and simulates fairly bandwidth sharing for

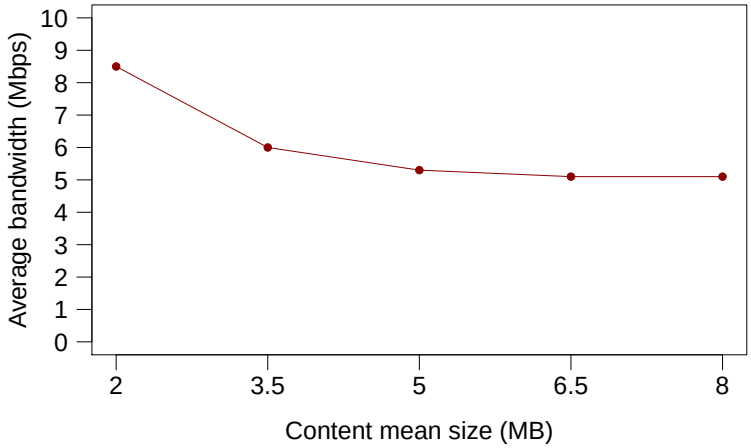

Figure A.5: Impact of content mean size on average uplink bandwidth usage of content providers.

heterogeneous networks transmitting blocks of equal size, it is hard to be successfully reused in generic scenarios, particularly when accurate fair-share scheduling is required.

We have implemented a simpler version of this approach, called a packet-based bandwidth scheduler, that is easier to configure, and enhances significantly the bandwidth scheduling mechanism. In our implementation, we promote the chunk size as a key parameter. Slots duration does not depend of bandwidth match any more. Instead, we assume that a slot is a portion of bandwidth according to the chunks size definition. For example, if a uplink bandwidth is equal to 10Mbps, and the chunks size is 1MB, the number of slots is defined by dividing bandwidth by chunks size, in this case, it would be about 10. We have also added a queue for untreated or on-going requests that permits improving significantly the scheduling of content transfer with multiple chunks. When a request does not have enough available slots on both source and destination for transmitting all its chunks, the remaining chunks are put into the queue. Remaining chunks are served following FIFO queueing and according to the slots availability. Considering a degree of parallelism equal to four and content mean size of 3MB, we have evaluated the content delivery performance by computing the average uplink bandwidth usage on content providers, and scalability through measuring the computation time per transfer of different chunk sizes. Figure A.6 shows that we are able to improve significantly bandwidth sharing accuracy by reducing the chunk size. In this case, when the chunk size is reduced from 8MB to 500KB, the average uplink usage increases from 2.5Mbps to 8.9Mbps. Yet improved accuracy pays its price, as depicted in Figure A.7. The computation time per transfer for the same range of chunk sizes is increased from 3.1 to 5.3 microseconds.

Although this approach improves the dynamics of bandwidth sharing resources, it introduces a crucial trade-off between accuracy and scalability. The smaller is the chunk size, the better is the accuracy it provides, but with an increasing computational resources consumption. Whatever the chunks size and the related computational cost, the use of packet-based, or slot-based, approach causes bandwidth allocation imprecision for incoming requests, that must wait the next free slot,

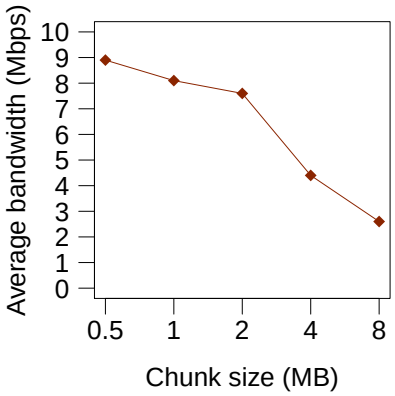

Figure A.6: Average uplink bandwidth usage on content providers with different chunk sizes.

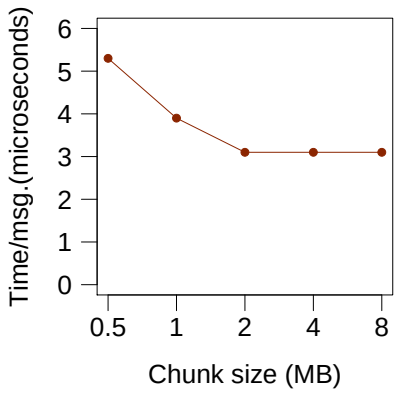

Figure A.7: Average computational time per content transfer with different chunk sizes.

or packet reading cycle, before starting. This is particularly damaging for small transfer lengths and strict rate control enforcement.

A.3.4 A Connection-oriented Bandwidth Scheduler

To overcome packet-based scheduler issues, improve accuracy in fair-share apportion, and simulate proper rate control for deadline-aware applications, we have designed and implemented a connection-oriented bandwidth scheduler in our Network layer for PeerSim, that was also initially based on the bandwidth manager module proposed by Russo *et al.* [112]. In fact, we are not interested in reproducing a wide range of realistic networking aspects, as data retransmission, packet-loss, or jitter. Instead, we focus on implementing a lightweight and accurate bandwidth scheduler that simulates the overall dynamics of bandwidth allocation on end-nodes. Therefore, we put great emphasis on enforcing fair-share scheduling for Peer-to-Peer networks. In our model, connection objects keep only the more precious information about the transfer, such as source-destination addresses, data to be transmitted, current allocated bandwidth, and remaining time. It permits computing bandwidth allocation and reproducing parallel and concurrent data transfer properly. We have run and compared the accuracy of our connection-oriented approach and a packet-based one, as depicted in Figure A.8. For a chunk size of packet-based approach equals to 500KB, our connection-oriented bandwidth scheduler performs roughly 10% better.

Since fair sharing policy is enforced properly, we have been able to implement an accurate rate control mechanism based on D3 [141], discussed in Subsection A.2.1, that permits simulating data transfer for deadline-aware applications. Despite improving consistently the bandwidth allocation accuracy, unsurprisingly it requires more computational resources than other approaches evaluated in this work.

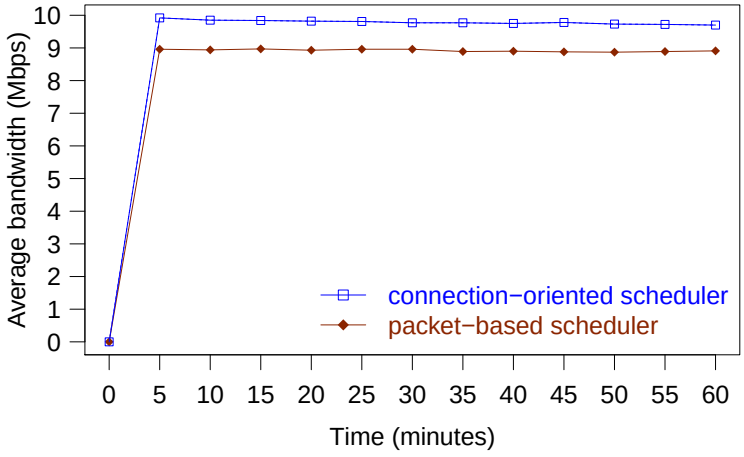

Figure A.8: Average bandwidth usage on content providers.

Table A.1: Outline of scalability and accuracy results of four bandwidth scheduling approaches.

Approach	**Accuracy**	**Parallelism**	**Computation time per transfer (μs)**	**Average memory usage (MB)**
Simplest	Lowest	Not allowed	0.5	8
Lock-based	Limited	Not allowed	2.2	162
Packet-based	Fair	Allowed	3.8	177
Connection-oriented	Highest	Allowed	46.8	431

A.4 Performance Analysis Summary

To provide a comparative study among the evaluated bandwidth schedulers, we have measured the performance in terms of scalability and accuracy over a common simulation set-up. In this common scenario, we assume the default configurations of Section A.3, we set the degree of parallelism to one, number of content providers to 100, or 10% of the total of nodes, and the content mean size to 3MB. For packet-based approach, we chose a chunk size of 500KB, half of minimum content size. Table A.1 provides an outline of evaluation results for all bandwidth schedulers. We highlight our main finds in the remaining part of this Section.

The simplest bandwidth scheduler: While the simplest approach is the easiest to implement, and highly scalable with a staggering computation time of only 0.5 μs per accomplished transfer and average memory usage of only 8MB, it performs the worst bandwidth sharing precision, and does not permit simulating parallel transfers. This approach might be useful for huge Peer-to-Peer networks that does not require fine-grained bandwidth tracking.

A lock-based bandwidth scheduler: Compared to the simplest approach, it performs a much better bandwidth allocation precision with excellent scalability in simulation duration, a still tiny computation cost per transfer of 2.2 μs. But the fact of implementing a FIFO queueing for scheduling incoming transfer requests causes a 20-fold increase in the average memory usage. Neither it offers parallelism, for us, an essential data transfer functionality. Lock-based approaches are rather suitable for scenarios with large number of nodes where data transmission in pipeline is acceptable.
A packet-based bandwidth scheduler: A packet-based bandwidth scheduler provides a quite fair scalability. Compared to a lock-based scheduler, it performs a minimal rise in average memory usage, an additional memory usage of 14MB on average, and a relatively high, but still impressive enough, increase of 70% in the computation time per transfer. Allowing parallelism in data transfers, it offers a good bandwidth scheduling mechanism for a wide range of applications. However, it exposes analysts to a trade-off between scalability and bandwidth scheduling precision through the choice of the chunks size. Assuming bandwidth allocation precision as a key feature for deadline-aware applications, this approach does not fit for purpose.
A connection-oriented bandwidth scheduler: Accuracy in simulating fair-share scheduling of bandwidth is at the core of our implementation. It performs precise bandwidth sharing thanks to connections-oriented approach. Unlike simulators that implement in-depth transport protocol behaviours, we minimize the computational cost improving scalability by maintaining only the most essential connections information. Although it causes a nearly 11-fold increase in computation time per transfer and a memory usage two and half times higher both compared to packet-based approach, connection-oriented bandwidth scheduler provides proper fair-share of bandwidth, what allowed us to successfully implement a strict rate control for simulating deadline-aware cloud applications.

A.5 Conclusion

We have designed and implemented a PeerSim component that provides accurate bandwidth sharing and rate control properly. Our component is particularly useful for predicting the performance of deadline-aware cloud services. It allows analysts to easily set-up and customize simulations thanks to the PeerSim modular and user-friendly API. We have evaluated the performance our bandwidth scheduling approach and compared it to three simpler and more scalable bandwidth approaches. As we have shown, simulation accuracy always pays its price. Our approach is not the fastest, neither reproduces in-depth transport protocol behaviours. We have rather chosen to implement a straightforward, lightweight, and accurate enough approach. We chose an approach that is based on a connection-oriented bandwidth scheduler that fits deadline-aware cloud application requirements properly.

Bibliography

[1] Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/ec2/, 2012.

[2] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang. Vivisecting youtube: An active measurement study. In *INFOCOM*, 2012.

[3] A. Adya, W. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In *OSDI*, 2002.

[4] Akamai technologies. www.akamai.com, 2013.

[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center tcp (dctcp). In *SIGCOMM*, 2010.

[6] V. Almeida, A. Bestavros, M. Crovella, and A. De Oliveira. Characterizing reference locality in the www. In *PDIS*, pages 92–103. IEEE, 1996.

[7] Amazon cloudfront. http://aws.amazon.com/cloudfront/, 2013.

[8] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *ACM SIGCOMM Computer Communication Review*, 32(1):66–66, 2002.

[9] N. Anderson. P2p traffic drops as streaming video grows in popularity. *Ars Techica, http://arstechnica. com*, 2008.

[10] M. F. Arlitt and C. L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking (ToN)*, 5(5):631–645, 1997.

[11] M. Baentsch, L. Baun, G. Molter, S. Rothkugel, and P. Sturn. World wide web caching: The application-level view of the internet. *Communications Magazine, IEEE*, 35(6):170–178, 1997.

[12] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. An analysis of latent sector errors in disk drives. In *ACM SIGMETRICS Performance Evaluation Review*, volume 35, pages 289–300. ACM, 2007.

[13] P. Barford, A. Bestavros, A. Bradley, and M. Crovella. Changes in web client access patterns: Characteristics and caching implications. *WWW*, 2(1-2):15–28, 1999.

[14] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker. Total recall: System support for automated availability management. In *NSDI*, volume 1, pages 25–25, 2004.

[15] N. Bonvin, T. G. Papaioannou, and K. Aberer. A self-organized, fault-tolerant and scalable replication scheme for cloud storage. In *SOCC*, 2010.

[16] L. Braun, A. Klein, G. Carle, H. Reiser, and J. Eisl. Analyzing caching benefits for youtube traffic in edge networks - a measurement-based evaluation. In *NOMS*, 2012.

[17] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM*, volume 1, pages 126–134. IEEE, 1999.

[18] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[19] A. Brodersen, S. Scellato, and M. Wattenhofer. Youtube around the world: geographic popularity of videos. In *WWW*, 2012.

[20] R. Brussee, H. Eertink, W. Huijsen, B. Hulsebosch, M. Rougoor, W. Teeuw, M. Wibbels, and H. Zandbelt. Content distribution network state of the art. *Telematica Instituut, June*, 2001.

[21] D. Buffoni, C. Calauzenes, P. Gallinari, and N. Usunier. Learning scoring functions with order-preserving losses and standardized supervision. In *Proceedings of ICML-11*, pages 825–832. ACM, 2011.

[22] R. Buyya, M. Pathan, and A. Vakali. *Content delivery networks*, volume 9. Springer, 2008.

[23] B. Cain, P. Rzewski, G. Tomlinson, and M. S. Day. A model for content internetworking (cdi). Internet Engineering Task Force RFC 3466. Available on: http://tools.ietf.org/html/rfc3466, 2003.

[24] L. Chang and J. Pan. Towards the optimal caching strategies of peer-assisted vod systems with hd channels. *IEEE ICNP*, 2012.

[25] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou. Utility maximization in peer-to-peer systems. In *ACM SIGMETRICS Performance Evaluation Review*, volume 36, pages 169–180. ACM, 2008.

[26] I. Chochliouros, A. S. Spiliopoulou, and S. P. Chochliouros. Developing content delivery networks. *Encyclopedia of Multimedia Technology and Networking*, 3, 2009.

[27] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi. A survey on content-oriented networking for efficient content delivery. *Communications Magazine, IEEE*, 49(3):121–127, 2011.

[28] B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, F. Kaashoek, J. Kubiatowicz, and R. Morris. Efficient replica maintenance for distributed storage systems. In *NSDI*, 2006.

[29] B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris. Efficient replica maintenance for distributed storage systems. In *Proc. of NSDI*, volume 6, pages 225–264, 2006.

[30] Cisco visual networking index: Forecast and methodology, 2011-2016. www.cisco.com, 2012.

[31] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[32] B. Cohen. The bittorrent protocol specification, 2008.

[33] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *SIGCOMM*, 2002.

[34] A. Collins. *The Detour framework for packet rerouting*. PhD thesis, Citeseer, 1998.

[35] I. Cooper, I. Melve, and G. Tomlinson. Internet web replication and caching taxonomy. Internet Engineering Task Force RFC 3040. Available on: http://tools.ietf.org/html/rfc3040, 2001.

[36] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.

[37] C. R. Cunha, A. Bestavros, and M. E. Crovella. Characteristics of www client-based traces. Technical report, Boston University Computer Science Department, 1995.

[38] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. *ACM SIGOPS Operating Systems Review*, 35(5):202–215, 2001.

[39] Direct download link. http://en.wikipedia.org, 2013.

[40] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[41] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *SIGOPS*, 2007.

[42] P. J. Denning. Working sets past and present. *Software Engineering, IEEE Transactions on*, (1):64–84, 1980.

[43] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *Internet Computing, IEEE*, 6(5):50–58, 2002.

[44] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. In *SIGCOMM*, 2011.

[45] J. R. Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[46] N. Dukkipati. *RCP: Congestion Control to Make Flows Complete Quickly*. PhD thesis, Department of Electrical Engineering, Stanford University, 2006.

[47] High-definition video delivery network. http://www2.edgestream.com, 2013.

[48] J. Evans and C. Filsfils. Deploying diffserv at the network edge for tight slas, part 1. *Internet Computing, IEEE*, 2004.

[49] L. Farina and S. Rinaldi. *Positive linear systems: Theory and applications*. Wiley-Interscience, 2011.

[50] D. Ferrari, A. Banerjea, and H. Zhang. Network support for multimedia a discussion of the tenet approach. *Computer Networks and ISDN Systems*, 26(10):1267–1280, 1994.

[51] F. Figueiredo, F. Benevenuto, and J. M. Almeida. The tube over time: characterizing popularity growth of youtube videos. In *WSDM*, 2011.

[52] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao. Youtube everywhere: impact of device and infrastructure synergies on user experience. In *IMC*, 2011.

[53] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan. Availability in globally distributed storage systems. In *OSDI*, pages 1–7. USENIX Association, 2010.

[54] Free. http://www.free.fr.

[55] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *SOSP*, pages 29–43, New York, NY, USA, October 2003. ACM Press.

[56] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM, 2007.

[57] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *PAM*. 2008.

[58] I. Global. Traffic forecast and methodology, 2006-2011. *Cisco white paper*, 2008.

[59] S. D. Gribble and E. A. Brewer. System design issues for internet middleware services: Deductions from a large client trace. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 207–218, 1997.

[60] A. Haeberlen, A. Mislove, and P. Druschel. Glacier: Highly durable, decentralized storage despite massive correlated failures. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 143–158. USENIX Association, 2005.

[61] T. Hastie, R. Tibshirani, and J. J. H. Friedman. *The elements of statistical learning*, volume 1. Springer New York, 2001.

[62] O. Heckmann and A. Bock. The edonkey 2000 protocol. *Multimedia Communications Lab, Darmstadt University of Technology, Tech. Rep. KOM-TR-08-2002*, 2002.

[63] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A measurement study of a large-scale p2p iptv system. *Multimedia, IEEE Transactions on*, 9(8):1672–1687, 2007.

[64] X. Hei, Y. Liu, and K. W. Ross. Inferring network-wide quality in p2p live streaming systems. *Selected Areas in Communications, IEEE Journal on*, 25(9):1640–1654, 2007.

[65] G. Held. *A practical guide to content delivery networks*. CRC Press, 2010.

[66] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *ACM SIGCOMM*, 2007.

[67] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 133–144. ACM, 2007.

[68] C. Huang, A. Wang, J. Li, and K. W. Ross. Measuring and evaluating large-scale cdns. In *IMC*, 2008.

[69] Y. Huang, T. Z. Fu, D.-m. Chiu, J. C. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p vod system. In *ACM Sigcomm*, 2008.

[70] Hulu venture. www.hulu.com, 2013.

[71] S. Irani. Page replacement with multi-size pages and applications to web caching. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 701–710. ACM, 1997.

[72] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *TOIS*, 20(4):422–446, 2002.

[73] S. Jin and A. Bestavros. Popularity-aware greedydual-size web proxy caching algorithms. In *ICDCS*, 1999.

[74] S. Jin and A. Bestavros. Greedydual< sup>âĹŮ</sup> web caching algorithm: exploiting the two sources of temporal locality in web request streams. *Computer Communications*, 24(2):174–183, 2001.

[75] B. Krishnamurthy and C. E. Wills. Proxy cache coherency and replacement-towards a more complete picture. In *ICDCS*, pages 332–339. IEEE, 1999.

[76] Y. Kulbak, D. Bickson, et al. The emule protocol specification. *eMule project, http://sourceforge. net*, 2005.

[77] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet interdomain traffic. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 75–86. ACM, 2010.

[78] M. Landers, H. Zhang, and K.-L. Tan. Peerstore: Better performance by relaxing in peer-to-peer backup. In *P2P*, pages 72–79. IEEE, 2004.

[79] S. Liang and D. Cheriton. Tcp-rtm: Using tcp for real time multimedia applications. In *ICNP*, 2002.

[80] Limelight networks. www.limelightnetworks.com, 2013.

[81] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A case for a coordinated internet video control plane. In *SIGCOMM*, 2012.

[82] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in bittorrent is cheap. In *HotNets*, pages 85–90. Citeseer, 2006.

[83] T. Loukopoulos and I. Ahmad. Static and adaptive data replication algorithms for fast information access in large distributed systems. In *ICDCS*, pages 385–392. IEEE, 2000.

[84] Business intelligence lowdown: Top 10 largest databases in the world, 2007.

[85] A. Mansy and M. H. Ammar. Analysis of adaptive streaming for hybrid cdn/p2p live video systems. In *ICNP*, 2011.

[86] D. S. Menasche, A. A. Rocha, B. Li, D. Towsley, and A. Venkataramani. Content availability and bundling in swarming systems. In *CoNEXT*, pages 121–132. ACM, 2009.

[87] Mirror image internet, inc. http://www.mirror-image.com/, 2013.

[88] A. Montresor and M. Jelasity. PeerSim: A scalable P2P simulator. In *P2P*, pages 99–100, 2009.

[89] Netflix: an american provider of on-demand internet streaming media. www.netflix.com, 2013.

[90] Netflix. http://en.wikipedia.org, 2013.

[91] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS*, 2010.

[92] G. On, J. Schmitt, and R. Steinmetz. Quality of availability: Replica placement for widely distributed systems. In *IWQoS*, 2003.

[93] G. On, J. Schmitt, and R. Steinmetz. Quality of availability: replica placement for widely distributed systems. In *IWQoS*, pages 325–342. Springer, 2003.

[94] E. J. O'neil, P. E. O'neil, and G. Weikum. The lru-k page replacement algorithm for database disk buffering. In *ACM SIGMOD Record*, volume 22, pages 297–306. ACM, 1993.

[95] Enabling digital media content delivery: Emerging opportunities for network service providers. www.velocix.com, 2010.

[96] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006.

[97] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson. Analysis of bittorrent-like protocols for on-demand stored media streaming. In *ACM SIGMETRICS*, 2008.

[98] M. Pathan and R. Buyya. A taxonomy of cdns. In *Content Delivery Networks*. Springer Berlin Heidelberg, 2008.

[99] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 2011.

[100] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33(1):59–64, Jan. 2003.

[101] R. Peterson and E. G. Sirer. Antfarm: Efficient content distribution with managed swarms. In *NSDI*, volume 9, pages 107–122, 2009.

[102] R. S. Peterson, B. Wong, and E. G. Sirer. A content propagation metric for efficient content distribution. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 326–337. ACM, 2011.

[103] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Peer-to-Peer Systems IV*, pages 205–216. Springer, 2005.

[104] D. Povey and J. Harrison. A distributed internet cache. *Australian Computer Science Communications*, 19:175–184, 1997.

[105] H. S. Rahul, M. Kasbekar, R. K. Sitaraman, and A. W. Berger. Performance and availability benefits of global overlay routing. In *Content Delivery Networks*, pages 251–272. Springer, 2008.

[106] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the oceanstore prototype. In *FAST*, pages 1–14, 2003.

[107] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: a public dht service and its uses. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 73–84. ACM, 2005.

[108] R. Rodrigues and B. Liskov. High availability in dhts: Erasure coding vs. replication. In *Peer-to-Peer Systems IV*, pages 226–239. Springer, 2005.

[109] P. Rodriguez, C. Spanner, and E. W. Biersack. Web caching architectures: hierarchical and distributed caching. In *Proceedings of WCW*, volume 99. Citeseer, 1999.

[110] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, pages 329–350. Springer, 2001.

[111] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 188–201. ACM, 2001.

[112] A. Russo and R. Lo Cigno. Delay-aware push/pull protocols for live video streaming in p2p systems. In *ICC*, pages 1 –5, 2010.

[113] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36(SI):315–327, 2002.

[114] B. Schroeder, E. Pinheiro, and W.-D. Weber. Dram errors in the wild: a large-scale field study. In *SIGMETRICS*, pages 193–204. ACM, 2009.

[115] E. Setton and J. Apostolopoulos. Towards quality of service for peer-to-peer video multicast. In *ICIP*, volume 5, pages V–81. IEEE, 2007.

[116] H. Shen. An efficient and adaptive decentralized file replication algorithm in p2p file sharing systems. *IEEE Transactions on Parallel and Distributed Systems*, 2010.

[117] A. Sherman, J. Nieh, and C. Stein. Fairtorrent: bringing fairness to peer-to-peer systems. In *CoNEXT*, pages 133–144. ACM, 2009.

[118] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *MSST*, pages 1–10. IEEE, 2010.

[119] G. Silvestre and S. Monnet. Performing accurate simulations for deadline-aware applications. In *MOSPAS*, July 2013.

[120] G. Silvestre, S. Monnet, R. Krishnaswamy, and P. Sens. Aren: a popularity aware replication scheme for cloud storage. In *ICPADS*, 2012.

[121] G. Silvestre, S. Monnet, R. Krishnaswamy, and P. Sens. Caju: a content distribution system for edge networks. In *BDMC*, 2012.

[122] H. Simon and G. Wuebker. BundlingâĂŤa powerful method to better exploit profit potential. In *Optimal Bundling*, pages 7–28. Springer, 1999.

[123] M. Sirivianos, J. H. Park, R. Chen, and X. Yang. Free-riding in bittorrent networks with the large view exploit. In *IPTPS*, 2007.

[124] M. Sirivianos, J. H. Park, X. Yang, and S. Jarecki. Dandelion: Cooperative content distribution with robust incentives. In *USENIX Annual Technical Conference*, volume 7, 2007.

[125] E. Sit, A. Haeberlen, F. Dabek, B.-G. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek, and J. Kubiatowicz. Proactive replication for data durability. In *IPTPS*, 2006.

[126] Akamai's sola sphere: Unsurpassed storage and delivery, optimized for media content. http://www.akamai.com/html/solutions/sola-sphere.html, 2013.

[127] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 149–160. ACM, 2001.

[128] Bittorrent. http://bittorrent.com.

[129] G. Szabo and B. A. Huberman. Predicting the popularity of online content. *Communications of the ACM*, 2010.

[130] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering what-if deployment and configuration questions with wise. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 99–110. ACM, 2008.

[131] J. Terrace and M. J. Freedman. Object storage on craq: High-throughput chain replication for read-mostly workloads. In *Proc. USENIX Annual Technical Conference*, page 59, 2009.

[132] R. Tewari, M. Dahlin, H. Vin, and J. Kay. Beyond hierarchies: Design considerations for distributed caching on the internet. *The Univertisy of Texas at Austin*, 1998.

[133] V. Tsaoussidis and C. Zhang. Tcp-real: receiver-oriented congestion control. *Computer Networks*, 40(4):477–497, 2002.

[134] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez. Greening the internet with nano data centers. In *CoNEXT*, pages 37–48. ACM, 2009.

[135] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). *ACM SIGCOMM Computer Communication Review*, 42(4):115–126, 2012.

[136] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). *SIGCOMM*, 2012.

[137] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[138] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained tcp retransmissions for datacenter communication. In *SIGCOMM*, 2009.

[139] H. Weatherspoon and J. D. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Peer-to-Peer Systems*, pages 328–337. Springer, 2002.

[140] S. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *OSDI*, 2006.

[141] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron. Better never than late: Meeting deadlines in datacenter networks. In *SIGCOMM*, 2011.

[142] N. Young. On-line caching as cache size varies. In *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, pages 241–250. Society for Industrial and Applied Mathematics, 1991.

[143] Youtube. http://www.youtube.com.

[144] Y. Zhou, T. Z. J. Fu, and D. M. Chiu. A unifying model and analysis of p2p vod replication and scheduling. In *INFOCOM*, 2012.