NNT : 2016SACLS318

Thèse de doctorat de l'Université Paris-Saclay préparée à "Université Paris Sud" et à "Télécom ParisTech"

Ecole Doctorale n° 580
Sciences et technologies de l'information et de la communication (STIC)

Réseaux, Information et Communications

Par

M. Andrea Giuseppe ARALDO

Design and Evaluation of Enhanced Network Caching Systems to Improve Content Delivery in the Internet

Thèse présentée et soutenue le « 07 Octobre 2016» :

Composition du Jury :

M. CHAHED Tijani	Professeur, Télécom SudParis	Président
M. SIMON Gwendal	Professeur associé, Télécom Bretagne	Rapporteur
M. LEONARDI Emilio	Professeur, Politecnico di Torino	Rapporteur
Mme CAROFIGLIO Giovanna	Ingénieure, Cisco	Examinatrice
M. MARTIGNON Fabio	Professeur, Université Paris Sud	Co-directeur de thèse
M. ROSSI Dario	Professeur, Télécom ParisTech	Co-directeur de thèse

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Conception et Évaluation de Systèmes de Caching de Réseau pour Améliorer la Distribution des Contenus sur Internet

Mots clés : Caching de Réseau ; Distribution des Contenus ; Optimisation de Réseau

Résumé : Le caching de réseau peut aider à gérer l'explosion du trafic sur Internet et à satisfaire la Qualité d'Expérience (QoE) croissante demandée par les usagers. Néanmoins, les techniques proposées jusqu'à présent par la littérature scientifique n'arrivent pas à exploiter tous les avantages potentiels. Les travaux de recherche précédents cherchent à optimiser le hit ratio ou d'autres métriques de réseau, tandis que les opérateurs de réseau (ISPs) sont plus intéressés à des métriques plus concrètes, par exemple le coût et la qualité d'expérience (QoE). Pour cela, nous visons directement l'optimisation des métriques concrètes et montrons que, ce faisant, on obtient des meilleures performances.
Plus en détail, d'abord nous proposons des nouvelles techniques de caching pour réduire le coût pour les ISPs en préférant stocker les objets qui sont les plus chères à repérer. Nous montrons qu'un compromis existe entre la maximisation classique du hit ratio et la réduction du coût.
Ensuite, nous étudions la distribution vidéo, comme elle est la plus sensible à la QoE et constitue la plus part du trafic Internet. Les techniques de caching classiques ignorent ses caractéristiques particulières, par exemple le fait qu'une vidéo est représentée par différentes représentations, encodées en différents bit-rates et résolutions. Nous introduisons des techniques qui prennent en compte cela.
Enfin, nous remarquons que les techniques courantes assument la connaissance parfaite des objets qui traversent le réseau. Toutefois, la plupart du trafic est chiffrée et du coup toute technique de caching ne peut pas fonctionner. Nous proposons un mécanisme qui permet aux ISPs de faire du caching, bien qu'ils ne puissent observer les objets envoyés.

Title : Design and Evaluation of Enhanced Network Caching Systems to Improve Content Delivery in the Internet

Keywords : Network Caching ; Content Delivery ; Network Optimization

Abstract : Network caching can help cope with today Internet traffic explosion and sustain the demand for an increasing user Quality of Experience. Nonetheless, the techniques proposed in the literature do not exploit all the potential benefits. Indeed, they usually aim to optimize hit ratio or other network-centric metrics, e.g. path length, latency, etc., while network operators are more focused on more more practical metrics, like cost and quality of experience. We devise caching techniques that directly target the latter objectives and show that this allows to gain better performance.
More specifically, we first propose novel strategies that reduce the Internet Service Provider (ISP) operational cost, by preferentially caching the objects whose cost of retrieval is the largest.
We then focus on video delivery, since it is the most sensitive to QoE and represents most of the Internet traffic. Classic techniques ignore that each video is represented by different representations, encoded at different bit-rates and resolutions. We devise techniques that take this into account.
Finally, we point out that the techniques presented in the literature assume the perfect knowledge of the objects that are crossing the network. Nonetheless, most of the traffic today is encrypted and thus caching techniques are inapplicable. To overcome this limit, We propose a mechanism which allows the ISPs to cache, even without knowing the objects being served.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l’Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Contents

Abstract **9**

Résumé **11**

1 Introduction **13**

1.1 Content Delivery and Network Caching 14
1.2 Research Goals 15
1.3 Evaluation Methodology 18
1.4 Thesis Organization and Contribution 19
1.5 Publications 21

2 Context and State of the Art **23**

2.1 The Ecosystem of Content Delivery 23
2.2 Content Cache Architectures 27
2.3 Policies 33
2.4 Benefits of Caching and Limits of Current Techniques 35
2.5 Cost Reduction 38
2.6 Delivery of Video Content 42
2.7 Caching Encrypted Content 45

I Cost-Aware Caching **49**

Introduction **51**

3 Optimal Cost-Aware Caching **53**

3.1 System Model 53
3.2 Optimization objectives and constraints 55
3.3 Greedy Algorithm 56
3.4 Numerical results 58

4 An Online Distributed Policy for Cost Reduction **63**

4.1 The Metacaching Algorithm 64
4.2 Che's Approximation Model 67
4.3 Simulation Results 68

4.4 Sensitivity analysis of cost-aware design 77
4.5 Implementation Constraints 83

Summary **87**

II Improving Video Delivery through Caching **89**

Introduction **92**

5 Optimal Video Caching **93**
5.1 System model 93
5.2 Representation-Aware MILP 94
5.3 Discussion 96
5.4 Model of offline AS policies 97
5.5 Numerical Results 98

6 An Online Distributed Caching Strategy for Video Delivery **107**
6.1 QImpr Algorithm 108
6.2 Simulation Setup 108
6.3 Comparison with the Optimal Solution 109
6.4 Distributed policy: Bandwidth-Utility Trade-Off 109

Summary **113**

III Caching Encrypted Content **115**

Introduction **118**

7 Analysis of Caching of Encrypted Content **119**
7.1 System Model 119
7.2 Partitioning Algorithm 120
7.3 Preliminaries 123
7.4 Consistency 124
7.5 Convergence 125
7.6 Optimality Gap 130

8 Performance of Caching of Encrypted Content **131**
8.1 Evaluation Scenario 131
8.2 Convergence 133
8.3 Sensitivity Analysis 134
8.4 Changing Content Popularity 135

Summary **139**

IV Conclusion 141

9 Summary of our Contribution 143
9.1 Cost Reduction 143
9.2 Caching of Video 144
9.3 Caching of Encrypted Content 145
9.4 Final Remarks 146

10 Discussion and Future Work 147
10.1 Cost Reduction 148
10.2 Caching of Video 149
10.3 Caching of Encrypted Content 150
10.4 Beyond the Tackled Problems 151

Acknowledgments 153

List of abbreviations 171

Abstract

Network caching is a promising technique to cope with today Internet traffic explosion and to help sustain the demand for an increasing user quality of experience. Nonetheless, the techniques proposed so far in the literature do not exploit all the potential benefits. Indeed, previous work usually aims to optimize hit ratio or other network-centric metrics, e.g. path length, latency, etc., which are relevant for the research community. On the other hand, network operators are more focused on more practical metrics, like cost and quality of experience. The usual approach in the literature is to target network-centric metrics and then, in some case, to incidentally observe the consequent benefits induced on the practical metrics we mentioned above. Our approach is different, in that we devise caching techniques that directly target the latter objectives, and we show that this allows to gain better performance.

More specifically, we first propose novel caching techniques to reduce the Internet Service Provider (ISP) operational cost, by preferentially caching the objects whose cost of retrieval is the largest. We define these techniques *Cost-Aware Caching* strategies. We formalize the problem by means of Integer Linear Program (ILP) and provide a greedy algorithm that gives the optimal solution. Numerical results show a trade-off between the classic hit ratio maximization and cost reduction and give the theoretical bound to the cost-benefit caching can bring. We apply what we learned from the ILP to devise a novel online distributed policy that can be implemented in real networks and we give a probabilistic model based on Che's approximation. By means of large scale simulation, we compare the achieved benefit to the theoretical bound found via the ILP and we show the robustness of our solution in different scenarios.

We then observe that cost minimization cannot be the sole objective of an ISP; indeed, user quality of experience is another decisive factor. We focus on video delivery, since it is the most sensitive to user experience and represents the most part of the Internet traffic. Despite the fact that video is highly cacheable thanks to its inherent redundancy, classic caching techniques ignore its relevant peculiarities, the most important of which is that each video is represented by different files, which we call "representations", encoded at different bit-rates and resolutions. We introduce the *Representation Selection Problem*, which consists in selecting the right available representations when caching videos, a problem which has been neglected so far. We describe the problem in terms of Mixed Linear Integer Problem (MILP) and we highlight some structural properties

of the optimal solution, which guide us in designing an online distributed policy, implementable in real networks, which we show, by means of large scale simulation, to effectively balance user perceived utility and bandwidth usage.

Finally, we point out that the techniques presented in the literature assume the perfect knowledge of the objects that are crossing the network. Nonetheless, this assumption does not hold anymore, since most of the traffic today is encrypted between the user and the Content Provider (CP). As a consequence, all network caching techniques are practically inapplicable by the ISPs. To overcome this limit, we propose *Content Oblivious Caching*, which allows the ISPs to cache, even if they are not able to observe the objects being served. The ISP allocates the cache storage to various content providers so as to maximize the bandwidth savings provided by the cache: the main novelty lies in the fact that, to protect business-critical information, ISPs only need to measure the aggregated miss rates of the individual CPs and do not need to be aware of the objects that are requested, as in classic caching. We propose a cache allocation algorithm based on a perturbed stochastic subgradient method, and prove that the algorithm converges close to the allocation that maximizes the overall cache hit rate. We use extensive simulations to validate the algorithm and to assess its convergence rate under stationary and non-stationary content popularity. Our results (i) testify the feasibility of content-oblivious caches and (ii) show that the proposed algorithm can achieve within 10% from the global optimum in our evaluation.

Overall, our research shows that, despite network caching has been investigated for more than 20 years, it is still an interesting and open research problem, with room for novel ideas and techniques. Indeed the continuous evolution of Internet calls for an equally continuous evolution of network caching. This is necessary to keep caching effective in improving the most relevant metrics for network operators and users and more integrated with the evolving network architectures and the Internet economic ecosystem.

Résumé

Le caching de réseau peut aider à gérer l'explosion du trafic sur Internet et à satisfaire la Qualité d'Expérience (QoE) croissante demandé par les usagers. Néanmoins, les techniques proposées jusqu'à présent par la littérature scientifique n'arrivent pas à exploiter tous les avantages potentiels. Les travaux de recherche précédents cherchent à optimiser le hit ratio ou d'autres métriques de réseau, par exemple la longueur du chemin, le retard, etc. Ceux-là sont sans doute importants pour la communauté de recherche. Par contre, les opérateurs de réseau (ISPs) sont plus intéressés à des métriques plus concrètes, par exemple le coût et la qualité d'expérience (QoE). En littérature on vise usuellement l'optimisation des métriques de réseau et, dans certains cas seulement, on constate secondairement l'amélioration des métriques concrètes dont on parlais auparavant. Notre approche est différent, car nous visons directement l'optimisation des métriques concrètes et montrons que, ce faisant, on obtient des meilleures performances.

Plus en détail, d'abord nous proposons des nouvelles techniques de caching pour réduire le coût pour les ISPs en préférant stocker les objets qui sont les plus chères à repérer. Nous appelons ces techniques "Cost-Aware". Nous formalisons le problème par l'optimisation linéaire en nombres entiers (ILP) et proposons un algorithme glouton qui calcule la solution optimale. Les résultats montrent qu'un compromis existe entre la maximisation classique du *hit ratio* et la réduction du coût. De plus, nous concevons un algorithme distribué qu'on peut implémenter dans des réseaux réels et nous le décrivons avec des modèles probabilistes. Nous comparons par simulation la performance obtenue par l'algorithme distribué et la limite théorique calculée par l'ILP. Nous montrons que la performance est robuste par rapport à différents scénarios.

Ensuite, nous constatons que la réduction du coût ne peut pas être le seul but d'un ISP; en fait, la QoE est un autre aspect important. Nous étudions la distribution vidéo, comme elle est la plus sensible à la QoE et constitue la plus part du trafic Internet. Bien que la vidéo se prête facilement au caching grâce à sa redondance, les techniques de caching classiques ignorent ses caractéristiques particulières, par exemple le fait qu'une vidéo est représentée par différentes représentations, encodées en différents bit-rates et résolutions. Nous introduisons des techniques qui sélectionnent pour chaque vidéo une de ces représentation pour la stocker dans le cache, ce qui n'a pas été étudié jusqu'à présent. Nous décrivons le problème par un Mixed Integer Linear Program

(MILP) et nous trouvons des propriétés structurelles de la solution optimale que nous prenons en compte pour concevoir une stratégie de caching distribué qu'on peut implémenter dans des réseaux réelles. Par simulation nous montrons que notre stratégie distribué est efficace pour balancer l'utilité perçue par les usagers et le débit nécessaire.

Enfin, nous remarquons que les techniques courantes assument la connaissance parfaite des objets qui traversent le réseau. Toutefois, cette hypothèse n'est plus valable, car la plupart du trafic est chiffrée. Du coup, toute technique de caching ne peut pas fonctionner. Pour surmonter cet obstacle, nous proposons un mécanisme qui permet aux ISPs de faire du caching, bien que ils ne puissent observer les objets envoyés. L'ISP repart le stockage du cache parmi les différents fournisseurs de contenu (CPs), pour que la réduction du trafic soit maximisée. L'ISP n'a plus besoin de connaître les objets demandés, mais seulement de mesurer le miss-ratio agrégé de chaque CP. Cela permet de protéger les informations de business sensibles des CPs. Nous proposons un algorithme pour l'allocation du stockage de cache et prouvons que il converge vers une allocation proche à l'optimale, qui maximise le hit ratio global. Par simulation, nous validons l'algorithme et en étudions les performances.

En général, cette thèse montre que, malgré le caching a été étudié depuis au moins 20 ans, il est encore un sujet de recherche intéressant et ouvert. Il y encore du potentiel pour des nouvelles idées et techniques. En effet, l'évolution continue d'Internet comporte une évolution continue du caching de réseau aussi. Cela est nécessaire pour que le caching soit un moyen efficace d'améliorer les métriques les plus importantes (pour les opérateurs de réseau et les usagers) et plus intégré dans les architectures de réseau et l'écosystème Internet, qui évoluent continuement.

Chapter 1

Introduction

The fundamental architecture of the Internet is not so much different than Arpanet's, from the 1960's. TCP/IP is still at the core and information is exchanged in roughly the same way. Nonetheless, Internet is no more used to exchange few kilobytes of text among public and research institutions. A large fraction of world population, with heterogeneous interests, education and goals is now on the web. One by one, all the traditional activities are moving on the Internet: personal communication occurs on social networks, Voice over IP (VoIP) and messaging services are replacing the traditional phone calls and Short Message Service (SMS), mass media are consumed more and more on the web rather than on radio and TV, citizens can communicate with public institutions using online services, etc. In other words, Internet is pervading and, in some way, shaping our everyday life. Surprisingly, the TCP/IP-based has shown to be robust to this change, by efficaciously serving as a base for an intricate set of protocols and services, which, at their turn, are combined in more complex services. Nonetheless, the pervasiveness of the Internet arises new challenges:

- *Traffic growth.* The network must be able to handle an aggressively increasing amount of traffic (in 2019 the traffic will be 64 times the 2005's [1]).
- *Economic interests.* Internet can be seen today as an economic ecosystem, in which the main objective of each player is to maximize income. Technological evolution and economic interests are tightly coupled: a new technology may change the structure of the ecosystem, making new players appear or disappear or, on the contrary, promising technical solutions may never be implemented if they risk to hurt the interests of the most influential players.
- *User experience.* The network must be able to move data fast enough so as to permit to very demanding applications, as HD video streaming, video conferencing or online games, to be sufficiently responsive. Furthermore,

Quality of Experience (QoE) is a very sensitive aspect for service providers as, in a highly competitive environment, they can only attract users if the quality of the service they provide is sufficiently high.

- *Secrecy of information.* The information that a Content Provider (CP) sends to users must be kept secret, for several reasons. First, since users spend a larger and larger fraction of time on the Internet, profiling their online activity could reveal too much information about the their real life, whence privacy is a primary concern. Second, CPs are interested in having the exclusive knowledge and control of the content they serve, in order to avoid unauthorized copies, to depict profiles of their users, compute statistics about the interest of their content, etc., which are at the core of their business.

In this thesis, we investigate how network caching can help cope with some of these challenges and how it can harmonize with them.

1.1 Content Delivery and Network Caching

Most current traffic [2] is represented by *content delivery*. In content delivery, *Content Providers* (CPs), like Youtube, Netflix, Spotify, provide pieces of content, which we also call *objects*, e.g. videos, audio files, pictures, HTML pages, etc. *Users* request objects and *Internet Service Providers* (ISPs) manage the networks for transmitting them. We distinguish content delivery from the other types of transmission that do not comply with the description above, like VoIP, instant messaging, e-mail, etc.

To emphasize their economic role, we identify CPs, ISPs and users as the three *players* in content delivery. Content delivery shows high redundancy, since the same object is requested by different users and, consequently, transmitted many times. Caching is intended to exploit this redundancy. Indeed, retrieving all the content from the origin server is inefficient, since it requires many flows from different users to the server, although all flows carry, essentially, the same pieces of information. If objects have to cross long paths, the latency is expected to be high. Moreover, many links are used for the transmission, which translates in high bandwidth utilization and risk of congestion. To avoid this, network caching consists in placing in several network locations small memories[1], called *caches*, which can store a subset of the objects. Users can thus download the requested objects from the closest location. This has several advantages: since flows terminate on the cache, without the need to reach the origin server, traffic redundancy is cut down and less bandwidth is needed to transmit the same amount of information. The latency, the risk of congestion and the load on the CP infrastructure are also reduced. For this reason caching has been widely used starting from the 1990s, when content delivery began to be preponderant, and it is still widely used. What is more, new players, called Content Delivery Networks (CDNs), have entered the Internet ecosystem. Caching

[1]small compared to the origin servers

is at the core of their business. Some of them are among the wealthiest digital companies. We point out that CDN technology is just a way, among several possible other ones, to implement caching. For the time being, it has been shown to be the most successful, primarily because it efficaciously integrates in the economic structure of the Internet. Nonetheless we should expect other innovative alternative caching technologies. For example, some proposals, like Information Centric Networking (ICN) [3], advocate a reshape of the Internet architecture, considering caching as a primitive. What said shows that network caching is, on the one side, a hot topic very relevant in the current Internet and with remarkable economic implications and, on the other side, an interesting research problem with several future perspectives.

1.2 Research Goals

In the literature, caching has been mainly intended as a tool to cope with traffic growth and metrics classically studied to this aim are hit ratio, i.e. the fraction of requests that a cache is able to serve, path length, latency, server load, etc. However, traffic growth is just one of the challenges we pinpointed at the beginning of this chapter. We claim that caching can help cope with the other challenges as well, and that none of them must be neglected when designing caching mechanisms.

We start by considering, in Part I the possible economic implication of caching, in particular the potential achievable ISP cost saving. However, only targeting cost reduction leads to the risk of a poor service offered to users. Therefore, we study in Part II how to improve user utility when serving video content. Nonetheless, our cost-aware and video-related strategies, as well as all the caching strategies presented in the literature, cannot be applied by the ISPs on most of their traffic, namely the encrypted traffic. We tackle this problem in Part III.

We observe that the three problems we consider are deeply connected each other. Since our work is a first step toward their resolution, we investigate them separately. However, it should be clear after reading this thesis that the solutions we give in the three parts can coexist and be combined, which would be an interesting direction for our future work. We will now briefly introduce our contribution.

1.2.1 ISP Cost Reduction

We focus on how ISPs can reduce their operational cost through caching in Part I. When an ISP receives a request for an object that is not stored inside its network, it has to retrieve it from some other ISP, connected through an *inter-domain link*, generating traffic on it and paying for this traffic. On the contrary, objects cached inside the ISP network can be directly served with no need to generate inter-domain traffic. In the literature it has incidentally been observed that having an efficient cache, i.e. high hit ratio, brings inter-domain traffic reduction, which is trivial. We go further. We first show that there is a trade-off between hit ratio maximization and cost minimization. In other words, if we are able to maximize the hit ratio, we are at the same time losing a part of the potential cost saving. The root cause of this trade-off is the price heterogeneity of the inter-domain links: while classic caching is blind to prices, if we want to minimize cost, we must tend to preferentially cache objects that lie behind the most expensive inter-domain links.

We propose cache strategies that directly target cost minimization and we show that saving are remarkably higher than with classic caching. In other words, moving cost minimization from being a side effect of hit ratio-maximizing caching to being the final goal makes a crucial difference. In our opinion, maximizing hit ratio is interesting from a research point of view, but equivalent to doing caching for the sake of caching. On the contrary, we claim that the real objectives of a production network are different and more practical, cost minimization being one of these, and we show that tailoring caching directly to them has a tremendous beneficial impact.

1.2.2 User Experience Improvement in Video Delivery

We investigate how we can improve user experience in video delivery through caching in Part II. Video transmission is highly cacheable and redundant. Indeed, if we consider a video encoded at a certain bit-rate and resolution, it is practically immutable, differently, for example, from a dynamic web page, which often changes. This means that video traffic is potentially highly cacheable. Moreover, popular videos are requested many times by many users, whence the redundancy. Furthermore, video delivery constitutes most Internet traffic and this fraction is expected to increase up to 80% in 2019 [2]. These factors make caching of videos particularly interesting.

This explains the success of Content Delivery Networks (CDNs) (Sec. 2.2.3), which are particularly used to replicate video content. However, we claim that video caching has not reached full maturity, yet, for two reasons: i) the algorithms used by the different CDNs are not publicly available, as they are considered business sensitive information, and thus their efficiency cannot be openly evaluated and improved and ii) scientific research (with only some recent exceptions [4, 5]) has treated caching and video delivery as two orthogonal problems, studied by different communities.

The most currently adopted standards in video delivery, like DASH [6] and WebRTC [7], rely on *Adaptive Bit-Rate (ABR) Streaming* [8] (*Adaptive Streaming* in short), which represents a single video with different files. Each file is a different *representation* of the same object, at different *quality*, i.e. different resolution and bit-rate. When a user requests a video, she does not explicitly specify the quality representation, which is instead decided on the fly by a control algorithm based, among other factors, on the network conditions. While the assumption behind all classic caching techniques is that one request can be satisfied by one and only one file, this does not hold anymore for videos, in which we have also to choose the quality representation we want to serve.

As a consequence, while classic caching decisions can be decomposed in the object selection problem (which object we want to replicate in the caches) and the replica placement problem (in which network location we want to place its replicas), we add a new dimension, namely the *representation selection* problem, i.e. which all the available representations we want to cache. This is the main conceptual contribution of Part II. Again, our goal is not to optimize classic network-related metrics. Our objective is to maximize user utility, considering that the higher the representation quality served, the higher the user utility. We show that, to this aim, it is crucial to store the right representations of the right objects at the right locations. We give guidelines for helping in these choices and we provide caching strategies, which approach the optimal allocation.

1.2.3 Caching with Encrypted Transmissions

We tackle the problem of making caching feasible with encrypted transmission in Part III. We examine the case in which the ISP owns a cache infrastructure and aims at maximizing its efficiency. Classic caching assumes perfect visibility of the objects and their requests: the cache owner, the ISP in our case, must observe the objects that are transiting in its network in order to pick some of them and cache them. Moreover, it has to understand the requests in order to retrieve the desired objects from caches. Nonetheless, more than 50% of transmissions are encrypted [9] and this percentage is expected to increase, as the IETF Internet Architecture Board (IAB) recommends "protocol designers, developers, and operators to make encryption the norm for Internet traffic" [10]. Before transmitting the objects, Transport Layer Security (TLS) tunnels are established between the CP server (or some surrogate server) and the users, which are completely opaque to the ISP, thus making impossible any caching. As a consequence, ISPs cannot enjoy the advantages of caching, in terms of bandwidth saving, cost reduction, improved quality to their customers, etc.

To solve this limit, we propose caching as a service offered by the ISP to the CPs that opt to subscribe. The ISP partitions its cache in segments, and assigns each to a different CP. Each segment is exclusively managed by the respective CP, by means of a proxy process running in the cache infrastructure but remotely controlled by the CP. All the content of the CP is stored in an encrypted form in the respective segment, so the ISP and the other CPs cannot look into it, thus preserving the sensitive business information of CPs. The CP proxy is the only one to decide what to store in its segment, to see which objects are being requested and to retrieve them. The only task of the ISP is to decide the allocation, i.e. the size of each segment, in order to maximize the overall hit ratio. Indeed, more cache space should be ideally conceded to CPs that can exploit it better, i.e. the ones that have very popular objects. The problem is that the ISP must compute the allocation without being able to observe the request flows, consequently having no knowledge about the objects and their popularity. The ISP can only measure the miss rate, i.e. the amount of requests that are not satisfied by the cache segment of each CP. We propose an iterative algorithm, based on perturbed stochastic subgradient methods, which measures

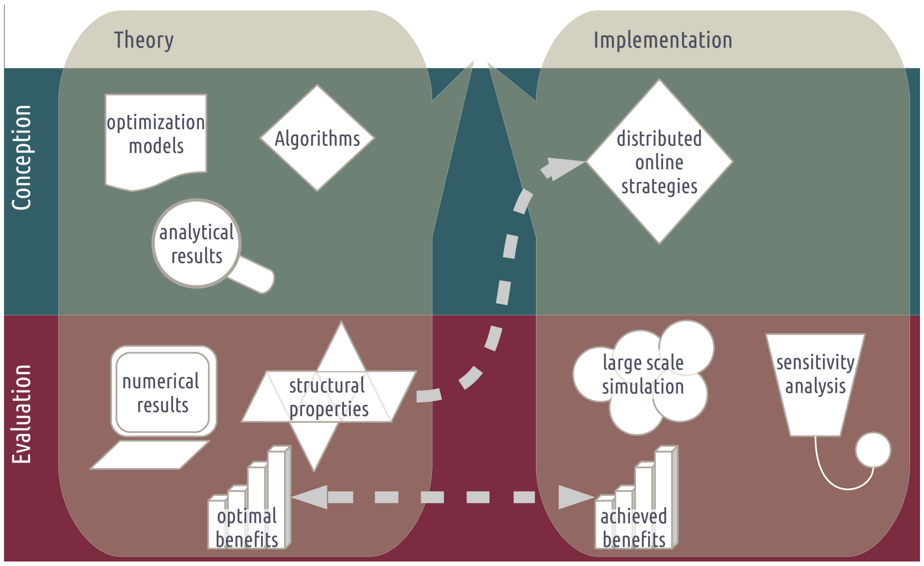


Figure 1.1: Methodology used in this thesis.

only the miss intensity and continuously adjusts the allocation. We analytically prove that the algorithm converges to an allocation that is boundedly close to the optimum. The convergence is robust to the unavoidable noise in the miss intensity measurement, which is due to the fact that each iteration is of finite time length and thus the number of requests that concur in the observed miss intensity is finite, thus preventing from reconstructing the average miss intensity of the single CPs.

An attentive reader may ask why we are maximizing the overall hit ratio in this case, while we have claimed that it is not the most relevant metric. We observe, in regards to this, that we provide here an architecture design and an algorithm that can be reused, with the necessary modifications, to optimize also other metrics, e.g. the cost. We target hit ratio here, just because it is still the reference metric in the literature, and because the focus is on the method (architecture and algorithm) more than the objective itself, in this case.

1.3 Evaluation Methodology

The benefits of network caching can be studied with different methodologies, spanning from the most theoretical to the most realistic. Since each methodology gives a different viewpoint, in our work we aim to use them in a harmonic way. Our approach is top-down. As depicted in Fig. 1.1, when proposing a cache strategy, we usually start by formalizing the scenario and the objective in an optimization framework. In Part I we provide an algorithm that computes the optimal solution in linear time, while in Part II we run the IBM CPLEX solver. We use the numerical results obtained either by the algorithm or by the solver to establish the theoretical performance bounds, the possible trade-offs between objectives and the structural differences with classic caching. Since offline policies are hardly implementable, we then propose online policies in Part I and Part II. In Part I, we give a probabilistic model of our online policy and then verify the performance through simulation, either in Octave or ccnSim [11], which is a simulator of network of caches. We compare our policies with state-of-the-art strategies present in the literature and with the optimum, which we use as a benchmark to seize the possible margin for improvement.

In Part III we used a different methodology. Our first goal is to mathematically prove the convergence of our algorithm, leveraging stochastic optimization techniques, namely stochastic subgradient methods. Also in this case, to verify that our algorithm works in realistic situations, we present a simulation campaign.

Overall, in our work we start from theory to end in simulation. Despite its crude simplification, theory gives us important hints that help us understand the problem and give us guidelines for the design of real policies. The final goal is, in any case, to propose implementable strategies, which we evaluate by means of simulation of realistic scenarios.

1.4 Thesis Organization and Contribution

This thesis is organized as follows.

State of the Art Chapter 2 reviews the network caching architectures and strategies conceived in the literature and implemented in production network. It pinpoints the limits that our proposals aim to tackle and discuss the related work.

Cost Reduction In Part I we present novel techniques to use caching as a mean to reduce the ISP inter-domain cost.

- Chapter 3 formalizes the problem. First, it provides the optimization model, which selects the objects to cache in order to minimize the ISP inter-domain cost. Second, we present a linear time greedy algorithm that gives the cost-minimizing solution. We provide numerical results running instances of the algorithm to get theoretical bounds on the achievable saving and useful guidelines.

- In Chapter 4 we propose an online distributed strategy for cost reduction. In more detail, it is a metacaching policy, which stores objects with a probability proportional to the price of the link behind which they can be retrieved. Our strategy is simple enough to be implemented in a network device working at line speed. Then, we theoretically model it through Che's approximation. By means of simulation, we first verify that results are compliant to the prediction of the model. Then, in simple and also realistic networks, we show that gain is remarkable and we evaluate how far we are from the optimum. We also provide a sensitivity analysis to show the impact of policy tuning and scenario characteristics. Overall, we show that results are consistent in all the considered scenarios.

We also underline that during this part of our research activity, we released version 0.3 of ccnSim, an efficient and scalable open-source caching simulator that we enriched with cost-aware caching functionalities and that we make available at [11].

Caching of Video In Part II we deal with caching of videos and we introduce the *representation selection* problem, which consists in selecting the quality representation to cache, which has not sufficiently been considered previously.

- In Chapter 5 we present a Mixed Integer Linear Program (MILP), which decides which objects to cache, at which quality, at which locations and also the routing. The objective is to maximize user utility, which is directly related to the video quality served. We run a solver to get numerical results, which we will use as theoretical bounds. The main guideline that we obtain is that, at optimum, we must not cache different representations of the same objects, but just the right ones. In particular, the quality of cached objects must decrease with their popularity.

- In Chapter 6, we leverage what we have learned to design an online caching strategy that mimics the optimal solution and that we simulate in realistic large scale scenarios.

Caching of Encrypted Content In Part III we propose an architecture and an allocation algorithm to allow ISPs to cache despite encryption.

- In Chapter 7 we first give a conceptual view of our architecture and the system model. Second, we introduce the underlying assumptions and we present our partitioning algorithm, whose goal is to find the optimal allocation, i.e. the one maximizing hit ratio. We analytically show that the algorithm converges boundedly close to the optimal allocation.

- In Chapter 8 we give further implementation details and we evaluate the performance of the partitioning algorithm in simulations written in Octave. First, we check the convergence and then we evaluate the impact of the algorithm parameters as well as the sensitivity to the scenario characteristics. We also simulate the realistic case of time-varying content popularity and verify that, by appropriate tuning of the algorithm, we are able to assure the performance.

1.5 Publications

Journal Papers

Authors	Title	Journal	Covered by
A. Araldo, D. Rossi, F. Martignon	Cost-aware caching: Caching more (costly items) for less (ISPs operational expenditures)	IEEE Transactions on Parallel and Distributed Systems (TPDS), 27(5):1316-1330, 2016 [12]	Part I
V. Catania, A. Araldo, D. Patti	Parameter Space Representation of Pareto Front to Explore Hardware-Software Dependencies	ACM Transactions on Embedded Computing Systems, 14(4), 2015 [13]	-

Conference Papers

Authors	Title	Conference	Covered by
A. Araldo, G. Dán, D. Rossi	Stochastic Dynamic Cache Partitioning for Encrypted Content Delivery	ITC, Wurzburg, 2016 [14]	Part III
A. Araldo, F. Martignon, D. Rossi	Representation Selection Problem: Optimizing Video Delivery through Caching	IFIP Networking, Vienna, 2016 [15]	Part II
A. Araldo, D. Rossi, F. Martignon	Design and Evaluation of Cost-aware Information Centric Routers	ACM SIGCOMM Conference on Information-Centric Networking (ICN), Paris, 2014 [16]	Part I
A. Araldo, M. Mangili, F. Martignon, D. Rossi	Cost-aware caching: optimizing cache provisioning and object placement in ICN	IEEE Globecom, Austin, 2014 [17]	Part I

Workshop Papers

Authors	Title	Workshop	Covered by
A. Araldo, D. Rossi	A per-application account of bufferbloat: Causes and impact on users	TRaffic Analysis and Characterization(TRAC), Nicosia, 2014, **Best Paper Award** [18]	-
A. Araldo, D. Rossi	Dissecting bufferbloat: Measurement and per-application breakdown of queueing delay	ACM CoNEXT Student Workshop, Santa Barbara, 2013 [19]	-

Posters

Authors	Title	Venue	Covered by
A. Araldo, F. Martignon, D. Rossi	Enhancing Content Delivery through Caching	LINCS Workshop, Paris, 2016	Part I, Part II, Part III
A. Araldo, F. Martignon, D. Rossi	Enhancing Content Delivery through Caching	Digicosme Research Days, Paris, 2016	Part I, Part II, Part III
A. Araldo, M. Mangili, D. Rossi, F. Martignon	Analysis and Design of Next Generation Information Centric Networks	Forum STIC Paris-Saclay, Paris, 2014	Part I
A. Araldo, D. Rossi	Dissecting Bufferbloat: Measurement and Per-Application Breakdown of Queueing Delay	Traffic Monitoring and Analysis Workshop, London, 2014	-
A. Araldo, M. Mangili, D. Rossi, F. Martignon	Performance Analysis of Secure, Next Generation Content-Centric Networks	Forum STIC Paris-Saclay, Paris, 2013	Part I
A. Araldo, D. Rossi	Dissecting Bufferbloat: Measurement and Per-Application Breakdown of Queueing Delay	ACM CoNEXT 2013	-

Chapter 2

Context and State of the Art

In this chapter we review the network caching architectures (Sec. 2.2) and strategies (Sec. 2.3) currently used in production networks, as well as the propositions in the literature. At the same time, we introduce the terminology that will be used throughout the thesis. We then underline the benefits and limits of current approaches (Sec. 2.3.3) which trigger our research. We then describe the main players in the Internet that are impacted by caching (Sec. 2.1.1). Finally, the last three sections summarize the previous work closely related to the three problems tackle in the thesis, namely cost reduction by caching, caching of video content and caching of encrypted content.

2.1 The Ecosystem of Content Delivery

In this section we describe the Internet ecosystem starting from a very basic picture and incrementally enriching it up to include all the actors we will encounter in the rest of the thesis. We will later refer to this picture to understand the advantages that content caching bring to each of them.

2.1.1 The Main Players in Content Delivery

The goal of the Internet is exchanging pieces of information, which can be web pages, audio files, videos, etc. Information can be exchanged under two paradigms: *point to point* and *point to multi-point*. According to the former, information is produced by one user and transmitted to one other user only. Examples of point to point communication are e-mails, instant messaging services, Voice over IP (VoIP), etc. Differently, according to the point to multi-point paradigm, an entity called Content Provider (CP), is responsible for hosting and delivering the information to many users. Note that the CP does not necessarily correspond to the entity generating the information: Youtube and

Facebook are examples of CP hosting content generated by users; other examples are Neflix and Spotify which host content generated by movie production companies and music labels, respectively. Note that throughout this thesis we will use the terms information and content almost inter-changeably. The only subtle nuance that we will have in mind is that we will implicitly intend *content* as information that is interesting for more than one user. In this thesis we will focus on *content delivery*, implicitly referring to the point to multi-point paradigm, in which information *is* content.

Transmitting content means moving bits across network links, which are maintained and operated by other entities called *Internet Service Providers (ISPs)*. Each ISP owns an infrastructure, basically composed of internal links and routers, interconnected to the other ISPs' infrastructures by means of external links, which we call *inter-domain links*. ISPs, CPs and users are the three vital players in content delivery, in that it is conceptually impossible to remove one of them from the picture.

To make the picture of the ecosystem complete, we cannot avoid to also include *Content Delivery Networks (CDNs)*. A CDN is a network of servers distributed across the Internet which is paid by one or more CPs to distribute their content. Even though CDNs are today among the biggest companies in the Internet (examples are Akamai and EdgeCast) and are absolutely necessary to make Internet work, they are just a technological solution and, as for all technologies, nothing assures that it will keep being adopted and will not be supplanted by some other techniques in the future. In other words it is possible to think of a picture of the content delivery ecosystem without CDNs, and, for this reason, we do not consider them as vital as the other three players, e.g. users, CPs and ISPs, without which that picture would not be possible.

2.1.2 Economic Relations in the Internet

While Internet was originally just a network of computers, now it is a much more complex phenomenon that can be viewed under different perspectives. It is shaping society [20, 21], influencing politics [22] and psychology [23]. We emphasize the point of view of Internet as an economic environment. In general, "Money plays the largest part in determining the course of history" [24]. This is particularly true in the Internet, in which the technological evolution is tightly coupled to the economic interest of each actor. Under an economic viewpoint, the main goal of a CP is not providing content but maximizing its profit, as well as the main goal of an ISP is not transferring data but also maximizing its profit. The adoption of a new technological solution does not directly depend on its efficiency, evaluated from an engineering point of view, but only on the profitability for the biggest Internet players. In most cases, more efficient systems bring, as a consequence, more profits for most players and engineering and economics go hand in hand. But, as we will see later, this is not always the case. For this reason, in this thesis we will consider, when possible, the compliance of our proposals to the current economic relations, which we will briefly summarize in this section and depict in Fig. 2.1.

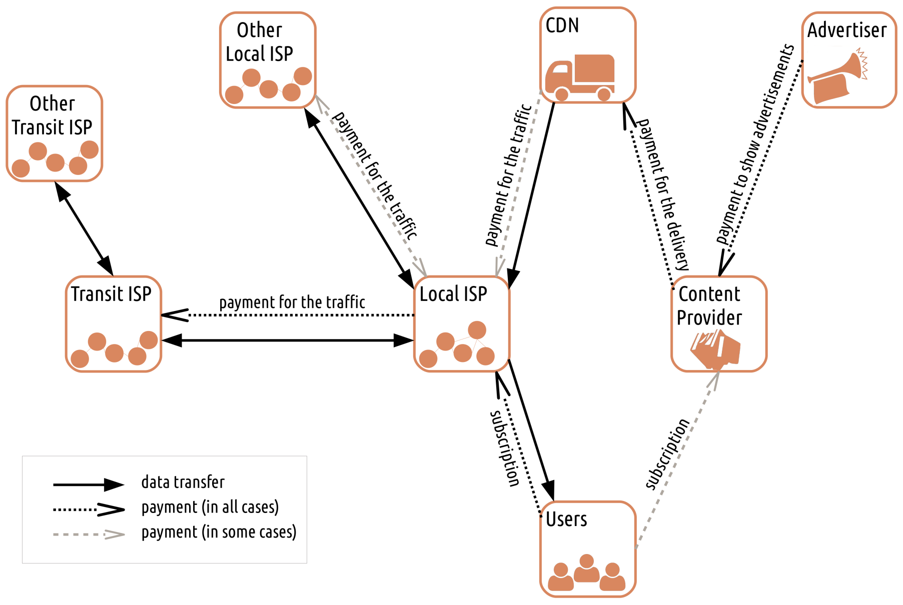


Figure 2.1: Economic relations in the content delivery ecosystem.

ISP income and cost According to the taxonomy in [25], we distinguish between *Local ISPs* and *Transit ISPs*. The latter are only connected to other ISPs, their role being only forwarding data. The former are also connected to users and CPs, and give connectivity to them. The link between two ISPs is called *inter-domain link* and may be ruled by different kinds of agreement. A *settlement-free peering agreement* consists in exchanging the traffic without any payment. On the contrary, with a *transit agreement* one ISP pays the other to receive or send data. The connection between an ISP and a CP (or a CDN) obeys to the same kind of relations. Usually CPs or CDNs pay ISPs to send and receive data, but peering agreements are becoming more and more common [26], particularly for big CPs like Google or Netflix and big CDNs like Akamai. In some case, the ISP receives also a fee from CPs or CDNs to allow them locate their racks inside the ISP network (Sec. 2 of [27]). Local ISPs additionally receive payments from the users subscribed to their services, either adhering to a *usage based billing* or to a *fixed subscription fee*. The former, in which users pay proportionally to the traffic they generate, was mainly used in previous years, while the latter, in which users pay a fixed fee periodically, e.g. every month, independently from the traffic, is prevalent today [28, 29, 30]. Summarizing, ISPs income comes from other ISPs, CPs, CDNs and, in case of local ISPs, from users as well.

CP income and cost While ISP income is tied to the traffic exchanged, the CP revenue comes from

- users, who can
  - pay every time they access an object (*pay-per-view payment*), or
  - pay a fixed fee periodically, say per month
- advertisers, i.e. third party enterprises which pay the CP to show advertisements to the users consuming the content.

In both cases, the revenue perceived by the CP is directly related to the *number of views*, i.e. the number of times the content of the CP has been requested. Moreover, advertisers typically pay the CP more in case users click on the advertisement message and visit the relative page [31]. Therefore *targeted advertising*, which consists in trying to show the user an advertisement that is likely interesting for her, has an impact on the revenue. Counting the number of views and of clicks on advertisements and profiling users (to facilitate targeted advertising) are core functionalities for CPs and a caching architecture can be considered feasible only if it does not impede them. As concerns the CP cost, it is constituted by the payment to the connected ISPs for the traffic generated and, possibly, the payment to the CDNs to subscribe to their content delivery service, the installation cost of CP racks inside ISPs networks, if any.

CDN income and cost CDN income totally comes from subscribed CPs. On the other hand, CDNs pay ISPs to exchange traffic, unless settlement-free peering agreements are active and in some case also for the installation of the racks.

Conflicting goals between ISPs and CPs or CDNs The complex economic interactions between ISPs, CPs and CDNs have not only an impact on their revenues [32], but also on the effective persistence of the structural principles enforced in the Internet by regulators [33] and on user experience [27]. This motivates taking these interactions into account when studying content delivery. Frequent disputes arose in recent years (see Sec. 3 of [34]) between ISPs and CPs or CDNs. The latter, encouraged by peering agreements, tend to devour ISP bandwidth, as they aim to transmit large objects, e.g. high quality videos, to as many users as possible. To keep user experience high, ISPs continuously increase the bandwidth available in their infrastructures, increasing the capital expenditures and, on the other hand, are not sufficiently rewarded for their investment, since most of users pay a monthly fee that does not depend on the traffic generated. Moreover, once more bandwidth is available, CPs and CDNs tend to greedily use it by sending larger objects, e.g. videos at higher quality, and enriching the range of the online services they provide. This “traffic bloat”, together with a more and more aggressive competition in the market, is one of the possible reasons of the telecommunication company revenue decrease despite the investment increase (see page 2 of [35]). Content caching can help counteract these issues.

2.2 Content Cache Architectures

Caching is a technique intended to use a memory hierarchy in an efficient way. To start introducing the nomenclature used later, we will assume that information is divided in *objects*, which can be individually requested. The set of all objects is the *catalog*. Let us consider a memory hierarchy composed of two memories, which we denote with *base memory* and *cache memory.* Note that we are not necessarily referring to the memory hierarchy in a computer architecture, but, in general, to any form of memory, including memory distributed over a network. The base memory is large, able to store all the catalog, but the cost of access is high. Cost may denote access delay, energy consumption, monetary cost or other metrics of interest. On the contrary, the cache memory is small but guarantees a fast access. Caching consists in storing in the cache a subset of the catalog. Every time we access an object, first we look into the cache. If it is present, we say that a *cache hit* has occurred and the cached copy is accessed at low cost. Otherwise, the object is accessed at the base memory and we say that a *cache miss* has occurred in this case. Observe that accessing an object may mean reading or writing (modifying) it, but in this thesis we will consider just the first case. The constrained space available in the cache memory limits the amount of objects we can store there, and thus they must be chosen carefully. This choice is determined by a *caching policy* (or, equivalently, *caching strategy*). Cache efficiency is usually measured in terms of *hit ratio*, i.e. the fraction of requests that are served by the cache instead of the base memory, which straightforwardly also indicates the amount of traffic reduction guaranteed by the cache. The efficiency of a cache depends on the redundancy in the request sequence: if redundancy is high, i.e. there are few objects that are requested many times, traffic reduction is effective, since storing those objects prevents a large amount of requests to be forwarded toward the base memory.

Caching was introduced in computer architecture, where the memory hierarchy is composed, going from the largest to the smallest, of hard disk, main memory and processor cache. The concept started to be adopted in the context of content delivery in the Internet in the 1990s [36]. *Content caching* [37, 38, 39] consists in deploying, in different locations of the network, caches which can store a part of the catalog. The permanent copies of the objects are assumed to be stored in some servers (or clusters of servers) called *origin servers*, managed by the respective CP. The set of origin servers represent, overall, the base memory.

Content cache can be deployed in different segments of the network. The placement of a cache has an impact on its efficiency, which has been considered in the literature [40, 41]. This problem is not the main scope of our research and we limit ourselves to roughly observe that when a request flow traverses a node equipped with cache, it is "reduced", meaning that it is relieved from a part

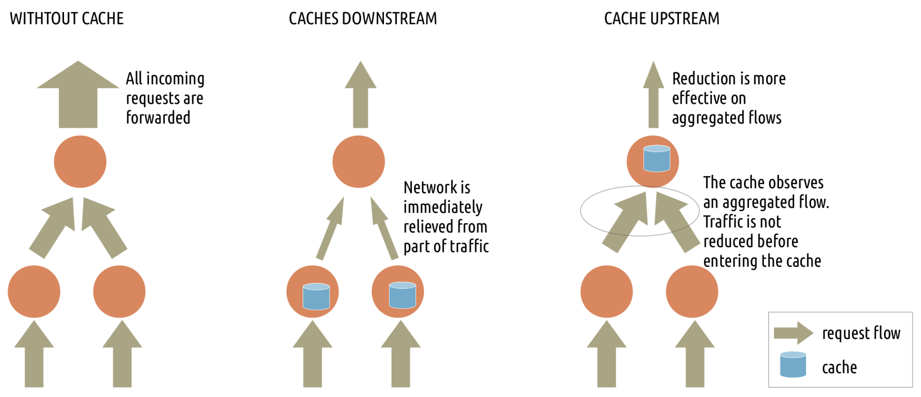

WITHTOUT CACHE
CACHES DOWNSTREAM
CACHE UPSTREAM
All incoming requests are forwarded
Network is immediately relieved from part of traffic
Reduction is more effective on aggregated flows
The cache observes an aggregated flow. Traffic is not reduced before entering the cache
request flow
cache

Figure 2.2: Flow reduction and aggregation.

of requests, and thus the size of the outgoing flow is smaller than the ingoing. As depicted in Fig. 2.2, if we place the cache close to users, the request flows are reduced immediately after their source point and therefore smaller flows will occupy the rest of the network. On the other hand, if caches are further from users, they can benefit more from *aggregation*, meaning that they can intercept flows coming from different users and, since the aggregated flow typically shows more redundancy than the single ones, cache efficiency is expected to increase. The drawback is that flows are unreduced until they encounter a cache, which implies a higher bandwidth utilization of the links that are crossed before a caching node.

2.2.1 User-Side Caches

All modern Internet browsers are equipped with a software cache, which intercepts the requests that can be thus satisfied locally in the user machine without generating any traffic in the network. Some enterprises or organizations also deploy proxy caches [36] in their Local Area Networks (LANs). A proxy cache is a software (like Squid) running on a generic server or a specialized hardware device, like CacheBox. It intercepts the request flows of all the users. Since the resulting aggregated flow typically shows more redundancy than the single flows, hit ratio is expected to be higher than with browser caches and thus they are more effective in reducing the traffic crossing the border of the LAN. On the other hand, they cannot reduce the traffic inside the LAN, as browser caches do. To use a proxy cache, the devices in the LAN must be configured in order to direct all the traffic to the proxy. This configuration can be manually or automatic, based in the latter case on the Web Proxy Auto Discovery Protocol (WPAD) [42].

2.2.2 CP-Side Caches

Reverse proxy caches are placed in the LAN of the CP. They are specialized devices, like ProxySG, or software running on a generic server (like NGINX), which act as a front end for the CP, meaning that they intercept all the requests directed to the it. A reverse proxy cache typically stores the most accessed objects of the CP in a fast volatile memory, to speed the content retrieval by avoiding the access to hard drives. Additionally, it can store the results of frequently executed computations, e.g. dynamic web pages or results of database queries. Note the traffic reduction is only provided inside the CP LAN and not in the Internet.

2.2.3 Network Cache

User-side and CP-side caches are not sufficient to cope with Internet traffic deluge, since the flow aggregation seen by the former is limited, whence their limited efficiency, and the latter does not reduce Internet traffic at all. Network caching, which this thesis is focused on, overcomes this limitation by placing caches in the Internet.

Mirror Servers In the early nineties [43], popular CPs started to deploy different servers, called mirror servers, at different locations (possibly different continents), offering the same service and the same content. Server mirroring is still used now, especially for software downloads. Its peculiarities are that each CP handles its own servers and that user explicitly selects the specific server she wants to contact. To be precise, mirror servers cannot be truly defined as caches, since they replicate all the original catalog, instead of just a part. We decided to include it in our discussion as they constitute one of the first example of content distribution and replication system.

Transparent Proxy Cache A transparent proxy cache system is a server or a systems of servers deployed by the ISP, in a way such that clients are unaware of their presence, and thus they do not need any specific configuration, neither manually nor automatic and they can operate as they were communicating with the origin server (whence the adjective "transparent"). A transparent proxy cache can be a specialized device, like CacheFlow, or a software running on a generic server, like CacheMara. Its work principle is detailed in [44]. Users send their requests toward the CP server, but the ISP routes them to its transparent proxy, which instantiates two communications at the same moment: i) a transparent proxy-client communication, in which the transparent proxy acts as the CP server and ii) a transparent proxy-server communication, in which the transparent proxy acts as a client. Then, the transparent proxy behaves as a man in the middle, relaying the data exchanged between the client and the origin server, allowing them to perform exactly the same operations as they were directly communicating each other. Since the transparent proxy has the complete visibility of the traffic, it can decide to cache some of the transmitted objects. Given that it intercepts the aggregated request flow of different users of the ISP, its efficiency is supposed to be larger than a client-side proxy server. In addition, traffic reduction occurs both inside the ISP network and on the inter-domain link. Transparent caching can be deployed in a much more distributed fashion, at the very edge of the network, as in the recent *small cell caching* proposals [45], intended for mobile networks. In order to improve transmission efficiency, mobile ISPs are deploying more and more antennas. The increased spatial density of the antennas permits to reduce the distance between the user device and the transmitting wireless source, thus limiting energy consumption and interference and improving throughput. Therefore, in the most populated areas, in addition to the classic macrocells (with a radius up to 35 Km), we find small cells (from 10m radius to few hundreds meters), which can be named femtocells, picocells, metrocells or microcells depending on their size [46]. Authors of [47] propose to equip the small cell base stations with a transparent cache, in order to serve user requests locally, thus saving traffic on the backbone.

Redundancy Elimination (RE) Also called *byte caching* [48, 49, 50, 51] , it was introduced in 2000 by authors of [52]. RE middleboxes are co-located with routers, can recognize online if there are identical portions of packets that are repeatedly sent and exploit this redundancy. Let us consider a link, with RE middleboxes at its ends. If a redundant portion is frequently sent on the link, both middleboxes associate to it a short identifier and save this association in a table. Then, in all following transmissions, a packet containing this portion is modified before the transmission on the link, replacing the portion with the index. When the modified packet arrives at the other end of the link, the receiving middlebox replaces the index with the actual portion, so that the original packet is reconstructed. Since indexes are shorter than the portions they replace, RE allows to transmit the same amount of information by using less bandwidth.

Content Delivery Networks (CDN) A CDN [53, 54] is a network of servers, called *surrogate* servers, distributed across the Internet and replicating the content of one or more CPs that subscribed to its service. Each user request is served by one of the CDN servers holding a valid replica. In the simplest case, the selected CDN server is the closest to the user, but selection can be based also on other criteria, e.g. load balancing, transmission cost, etc. [53]. A CDN provider may be a third party entity, like Akamai and EdgeCast, that is delegated by the subscribed CPs to distribute their content and receives payment for this service. Some big CPs have their own CDNs, like Google. Recently, also ISPs are deploying and offering CDN services [55, 56, 57, 58, 59]. Starting to appear in 1998 [53, 60], CDN providers are now among the biggest actors in the Internet and deliver the most part of the traffic generated by the biggest CPs [61]. We do not want to describe exhaustively the CDN operation, which, by the way, varies from company to company and is not completely public, being part of the business sensitive information. Here we limitedly underline the basic common aspects. More details can be found in the already cited work and, as far as modern commercial CNDs operation is concerned, in Sec. 2.1 of [62] and references therein. The CDN is transparent to the client which operates as it was communicating with the original CP server. The following mechanisms must be implemented to permit CDN operation:

- *Content transfer*, i.e. allowing CP to upload to the CDN storage the objects whose transmission is delegated.
- *Redirection*, i.e. redirecting to the CDN infrastructure the requests coming from users and originally directed to the origin server.
- *Object selection*, with which the CDN decides which objects should be replicated across the available surrogate servers
- *Replica placement*, with which the CDN decides in which of the available surrogate servers the objects should be stored.
- *Request routing*, with which the CDN routes each of the requests arriving at the CDN infrastructure to one of the surrogate servers holding the requested object (if none, the request is routed to the origin server).

Redirection can be implemented with two different mechanisms:

- *DNS redirection*. To be easily reachable, a CP always registers its domain name, e.g. `youtube.com`, to a domain name registrar [63], associating its name to an IP address. In the classic scenario, without any CDN, the associated IP address is the origin server's. If DNS redirection is used instead, the CP must modify [64] its association in the registrar, inserting the IP address of the CDN. Doing this, all the requests directed to the origin server will be automatically sent to the CDN.
- *URL rewriting*. This is the most diffused technique for CP-CDN redirection. Most of Internet content is accessed through HTML pages, which can be typically described as a basic skeleton enriched with links that point to other embedded objects (like images, videos, etc.). When URL rewriting technique is used, the CP delegates to the CDN the delivery of the embedded objects only, which are usually larger than the skeleton itself. To this aim, the CP modifies the links present in the skeleton to make them point to the CDN. As a consequence, when browsers retrieve the HTML skeleton (directly from the origin server), they transparently download the embedded objects from the surrogate servers.

Observe that object selection, replica placement and request routing are three tightly coupled problems that arise not only in the context of CDNs, but in general in all types of cache networks, including ICNs, which we describe below.

Finally, just to give an idea of the complexity of CDNs, we write here what reported by [65]: "Akamai's CDN currently has over 170,000 edge servers located in over 1300 networks in 102 countries and serves 15-30% of all Web traffic.".

Information Centric Networks (ICN) ICN [3, 66, 67] is a new network paradigm, alternative to the current IP network model. It has attracted a lot of interest in the research community but, to the best of our knowledge, has not been deployed in any production network, yet. A classic IP-based network is "host-centric", meaning that the communication is point-to-point, exchanging packets from a host machine with address `A` to another with address `B`. Since most Internet traffic consists in several users requesting a limited subset of objects, irrespective of their location, the point-to-point model, which is heavily based on the location of the communicating hosts, is not suitable [68] to exploit the inherent redundancy in such traffic. To overcome this limitation, in the ICN design routers are directly able to:

- read the name of the objects, instead of (or in addition to) the address of the server
- cache objects in an embedded cache before forwarding it.

Moreover, while the datagram exchanged in IP networks is the packet, i.e. a sequence of bytes going from one source to a destination, in ICN we have two datagrams: the *interest packet*, carrying a request for a named object, and the *data packet*, carrying a portion of an object. This adds new potential functionalities to the network:

- Requests can be routed based on the object name rather than the server address and can be satisfied by replicas placed at the closest (or best, based on different possible criteria) network location. This can be realized without the need of the redirection procedures implemented in CDN, which may degrade performance (see Sec. 2.4.2).
- Caching is a primitive of the network architecture.

Differently from CDN, ICN caching nodes are expected to be routers equipped with ultra-fast memory, capable of handling traffic at 10 Gbps and beyond [69, 70, 71, 72]. Consequently, these memories are more scattered in the network and smaller, up to few TB, resulting in a much more distributed cache system than current CDNs.

Different proposals can be covered under the umbrella of ICN: DONA [73], Content Centric Networking (CCN) [68], Named Data Network (NDN) [74], Conet [75], etc. They differ based on strategies regarding:

- *Naming*: each object must be assigned a name, which will be used by network devices to route requests, cache objects and retrieve them from caches. There can be different possible naming strategies: names can be human readable or not, the name space can be hierarchical or flat, etc.
- *Request routing*: can be mediated or not. In the first case, the network is equipped with a *rendez-vous system* [73], which intercepts all the requests and translates the requested object names into the address of the physical location where a replica can be found. In the second case, no location

address is used, and all the routing is solely based on the name of the objects.

- *Object authenticity*: in current networks, the information authenticity is guaranteed by secure communication channels, e.g. using Transport Layer Security(TLS), established between the client and the server (either origin server or CDN surrogate server). Once the secure communication is established, the client assume that the server will send authentic data. On the contrary, given that in ICN objects can be distributed in different locations, new mechanisms have been proposed to allow a client to verify the authenticity of objects, without secure communication channels. Authenticity is based on a secure *binding* between an object and its name, assuring that the object a user is receiving is exactly the one that she has requested. This can be done by *direct binding* or by *indirect binding*. The former consists in adding a hash of the object to its name, while in the latter a hash of the object, signed by the CP, is added to the object.

2.3 Policies

The policies by which we can manage a network of caches can be classified in two families: *offline policies* and *online policies*.

2.3.1 Offline Policies

Conceptually, offline policies assume an entity, which we call *network controller*, which decides replica selection (what to cache) and replica placement (which network location to cache in) and periodically updates these decisions. Observe that, although not directly related to caching, routing must be decided by the network controller, to determine the replica to send requests to. Offline policies determine these decisions, usually based on the results of optimization problems. In other words, we assume that at a generic time instant we have facilities, i.e. link capacities and caches, and a set of user requests. The goal is to find the facility allocation that optimize the objective function, which can be the hit ratio, the path length, the latency, etc. This approach, which has been named as *snapshot approach* [76], is based on this instantaneous picture of the system. It is not realistic, since the instantaneous picture of the system is never available, as we never exactly know in advance what users will request. In addition, offline policies are in many cases not feasible due to the excessive complexity of the optimization problems that must be solved. Moreover, they require the information about the network state to arrive to the controller, which adds delay to the content delivery. This information can also be difficult to reconstruct in a consistent way. All these problems have been largely observed (for example in Sec. 2.4 of [62]).

Nonetheless, offline policies have been widely studied by the scientific literature [77, 78, 79], since they permit to obtain a theoretical description of the system, which, although not realistic, can help in understanding its behavior.

Moreover, if we divide the time in finite length slots, we can predict [80], with a certain degree of precision, a picture of the system in the future slots, based on the observation of previous ones and we can compute the offline solution on this prediction. For this reason, when approaching a problem, we first leverage offline policies (Chapter 3 and Chapter 5), before investigating online strategies (Chapter 4 and Chapter 3).

2.3.2 Online Policies

In online policies, each node of the cache network takes its decisions, locally and independently. Every time an object arrives at the node, it decides whether to cache it or not, before forwarding it toward its destination. The policy ruling this decision was defined by authors of [81] as *metacaching policy*, also called *decision policy* or *insertion policy*. If the object is accepted, there may be no space in the cache to store it. In this case, one of the previously cached objects must be evicted. The selection of the object to be evicted is dictated by a *replacement policy*. In addition, when a request is received, the node must decide where to forward it. This is determined by a *forwarding policy*. While we do not want to repeat here a taxonomy of related work about these policies, already covered in a complete and clear manner by recent works (Sec. 2 of [82], Sec. 2.3 of [38], Sec. 5.2 of [83], [84]) and also by less recent ones [85, 86], we limit ourselves to describe the ones that we consider and extend. While most of the literature investigates replacement policies, we embed our strategies in metacaching policies.

The simplest metacaching policy is *Leave a Copy Everywhere* (`LCE`), which accepts to store all the objects that arrive at the node. An alternative is *Fixed Probabilistic Caching* (`Prob`) [81, 87], which accepts every arriving object with a fixed probability $\bar{\psi} \in [0, 1]$, called *insertion probability*. This policy acts as a filter, as it manages to store only sufficiently popular objects, which arrive often to the cache and, after several trials, end to be accepted. At regime, `Prob` has a better hit ratio than `LCE` and, if we ideally make the acceptance probability tend to 0, we will find in cache exactly the most popular objects (Theor. 1 of [88]). However, it is important to underline that during the initial transient phase performance is poor, since we refuse to store a subset of objects, thus losing the related traffic saving. In other words, `Prob` caches objects only after they are observed many times ($1/\bar{\psi}$ times on average), and thus there is a long transient dominated by cache misses. Moreover, the smaller the acceptance probability, the longer the transient, meaning that an effective filtering effect can be obtained only after a long time.

As for the replacement policies, we will leverage *Least Recently Used* (LRU), which evicts from cache the least recently requested object. Other finer policies have been proposed but they are typically too complex to be implemented at line rate, e.g. 10 Gbps, which is the expected object arrival rate in a real network [69, 70, 71, 72]. We remark that, while other replacement policies exist, like Least Frequently Used (LFU) and Last Recently/Frequently Used [89], which are more complex than LRU but still simple, most of the literature still refers to LRU.

For the sake of completeness, we also point out that Instead of evicting objects every time room is needed for new ones, *Time-to-Live* (TTL) *based caches* [90, 91] evicts objects a certain amount of time after they were inserted.

2.3.3 Cache Partitioning

Cache may be partitioned in different segments, each one handled separately by either an offline or online policy. The problem of cache partitioning has first been considered in the context of CPU cache [92, 93, 94, 95]: CPU cache is partitioned among different competing processes or it is divided in two segments, one for the instructions and one for the data.

On the other hand, cache partitioning is not frequently applied in network caches. Some exceptions are [96, 97]. Each segment of the partition is dedicated to a different application in [96]and [98], while in [97] the ISP manages a cache and partitions it, assigning each segment to a different CP.

In any case, the problem of finding the allocation, i.e. deciding the size of each segment, must be solved. In Part III we will describe the novel cache partitioning technique, based on perturbed stochastic subgradient method, we first proposed in [14]. Moreover, we will consider cache partitioning when dealing with video transmission (Chapter 5), using each segment for storing one specific video quality only.

2.4 Benefits of Caching and Limits of Current Techniques

This section discusses the advantages of content caching for the Internet players we pinpointed in Sec. 2.1.1 and then underlines the limits of current technology.

2.4.1 Benefits

The root cause of all the advantages of caching can be explained in simple terms by observing that caches are typically closer to the users than the origin server and thus the length of the path (usually expressed in number of hops) the data have to cross is reduced. This has many straightforward consequences:

1. The propagation delay and the total processing time of the routers is reduced, and thus the latency experienced at each user request.

2. The total bandwidth consumed per each transmission, computed as the sum of the bandwidth used at each link, is reduced as well. As a consequence, we can transmit more data in the network with no need to increase the link capacity. This also reduce the probability of link congestion.

3. The load on the origin servers is reduced.

These advantages are particularly evident when *flash crowds* occur, meaning that in a short time window a restricted set of objects attracts a very large number of requests, which happens very often in the Internet [99].

Advantages for users Consequence (1) has a direct positive impact on the user experience. Thanks to (2), users can consume more content and at a better quality, e.g. video or audio (we will analyze this in Part II), which improves the user experience, as well. As observed by [100], reduced latency and increased amount of content available can even make new services possible, e.g. tele-medicine, extremely interactive applications. They can also change the way current applications typically work: leveraging fast access to remote information, we can move more and more data and processing from the client to the server, allowing thinner clients on local machines, able to run with less resources and at low power, which is particularly interesting given that information will be consumed on tablets and smartphones more than on classic desktop computers [1].

Advantages for CPs Being able to serve more content at a higher quality to users, CPs can attract more views (see Sec. 2.1.2) and increase revenue consequently [100]. CPs also benefit from cost reduction. Indeed, caching relieves origin servers from a fraction of load, thus allowing to increase the amount of content served without updating the server infrastructure, which brings a capital expenditure saving. Furthermore, CPs typically pay for the traffic they generate on the ISPs (see Sec. 2.1.2) they are connected to. Since caching reduces this traffic, operational expenditure saving is possible as well.

Advantages for ISPs We have to distinguish between local ISPs and transit ISPs. As already observed, caches deployed in a local ISP can improve the quality of experience perceived by the subscribed users, thus making the ISP more attractive. Since a part of traffic is directly served from caches inside the ISP network, the inter-domain traffic is reduced, which brings operational expenditure saving (we will see how to maximize the saving in Part I). Moreover, more and higher quality content can be served without updating the capacity of links, which brings capital expenditure saving. On the contrary, transit ISPs have no incentive in caching [101], since they do not usually pay for the exchanged traffic (either ingress or egress traffic) and often, are even paid for it, in which case retrieving content from other ISPs constitutes a revenue.

2.4.2 Limits of Current Techniques

The research we conduct starts from the observation of the limits of current caching techniques and moves a step forward toward their resolution. The limits to take into account are not exclusively technical. As already discussed in Sec. 2.1.2, even if a caching infrastructure guarantees the best performance, it will hardly be adopted if it hampers the economic interests of the big players.

For this reason, we will take into account in this section also the potential incompatibility issues of the caching solutions with the business relations in the Internet.

CP loses control over content All caching architectures imply that the CP loses control on the replicated content, since when a user accesses a replica, she does not need to contact the origin server of the CP (unless specific mechanisms are put into place). This impedes [34] *access control*, which consists in limiting only to certain users the access to certain objects, and *trackability*, i.e. recording the accesses to the different objects from the different users. Note that access control permits to demand payment from users, while trackability is vital for preserving the income from the advertisers (Sec. 2.1.2), and thus they are core functionalities. While some research has been conducted on these issues [102], only CDNs solved them in practice and this is the main reason of their success. Access control and trackability are integral part of the service offered by CDNs [103, 104]. In practice, when CPs delegate content delivery to CDNs, the latter take the responsibility of performing those functionalities.

However, issues may arise related to possible malicious or erroneous misuse by CDNs of these delegated functionalities, e.g. erroneous report of statistics, which may also translate into an erroneous fee demanded to the CPs, etc. To the best of our knowledge these issues have not been studied by the scientific community. In practice, CPs suppose that CDNs are trusted parties, and these issues do not arise by assumption, which, in our opinion, does not offer sufficient guarantees. In Part III, we will present a solution that overcome these problems by giving CPs the complete control over their replicas.

Encryption As already discussed in Sec. 1.2.3, current caching techniques does not work with encrypted content. The reason is that they require the visibility of the objects transmitted, which is broken by encryption. We add here that even object-unaware techniques, like Redundancy Elimination, which do not require such knowledge, are inapplicable since encryption generate different streams at each transmission of the same object, nullifying the redundancy. CDNs circumvent the problem by establishing an encrypted channel between the client and the surrogate server. This is made at the expense of CPs, which must i) lose the control of their content delegating it to third party CDNs, ii) accept the risk of letting third party CDNs incarnate themselves when transferring data to users. In Part III, we propose an alternative cache architecture that can co-exist with encryption, without the issues i) and ii).

Permeation As already said, due to encryption, the only effective caching solution is CDNs. CDN surrogate servers are often directly connected to local ISPs [105] and in some case local ISPs let CDNs deploy servers inside the access network [106]. We say that CDNs *permeate* up to the "edge" of the network, i.e. the part of the network that it is close to users, and this allows to reduce immediately the traffic (see Fig. 2.2). However, in mobile networks, especially

Table 2.1: Metrics optimized by caching algorithms in the literature.

Hit ratio	[108, 109, 110, 111]
Number of hops	[108, 112, 41, 109, 82, 113]
Latency	[114, 115, 108, 116, 117]
Link load	[108, 59, 118]

in the envisaged 5G networks [107], the edge is even closer to the users: each ISP has a high number of small cells (Sec. 2.2.3), each of very small size, and it is infeasible for any CDN to host their nodes there. The only entity that can cache in small cells is the ISP which owns them. Unfortunately, classic ISP transparent caching cannot be applied on encrypted traffic. The architecture we propose in Part III solves this problem and allows the ISP to deploy caches up to small cells.

Classic caching goals are not the most interesting As already abundantly stated, almost all the caching strategies presented in the literature aims to maximize hit-ratio. In our opinion, this is not an interesting goal in itself and, on the contrary, is like aiming to cache for the sake of caching. The rest of previous work focuses on optimizing classic metrics like the number of hops, the latency or the link load. Tab. 2.1 summarizes the objectives pursued by caching in the literature. We claim that other interesting goals, which have not been sufficiently investigated, can be achieved. In Part I, for example, we propose new caching strategies that aim to minimize the inter-domain traffic cost of ISPs and we show that saving compared to classic caching techniques is remarkable. In Part II we focus on another important objective not sufficiently studied in the caching literature, namely the improvement of the video quality provided to users.

Caching is not tailored for multimedia We have already observed in Sec. 1.2.2 that classic caching, which would be most beneficial on video transmission, is not tailored for it. This discussion can be extended in general to every type of multimedia file, e.g. audio, pictures, etc. We attack the problem in Part II, where we propose caching techniques specifically tailored for video delivery.

In what follows, we summarize the related work specifically related to the three problems we tackle in this thesis: cost reduction, video delivery and caching of encrypted content. We summarize the methodologies in Tab. 2.2.

2.5 Cost Reduction

We discuss in this section the previous work related to the cost-aware caching problem, which we tackle in Part I. We mainly discuss the work that take into

Table 2.2: Methodologies used in the related work (more details in Sec. 2.5, Sec. 2.6 and Sec. 2.7) and in this thesis.

	Cost Reduction	Delivery of Video Content	Caching of Encrypted Content
Online distributed caching	[119, 120, 121, 122], Chapter 4	Chapter 6	
Game Theory	[123, 124, 125, 126, 127]		[97, 128]
Optimization	Chapter 3, [120, 129, 27]	[5, 130, 131, 4, 76, 27], Chapter 5	[132]
Stochastic Optimization			Part III
Economic Models	[101, 104]		[34]
Probabilistic Models	[133, 134], Sec. 4.2		
Control Algorithms		[135, 136, 137]	
Trace-driven Analysis		[138]	
Coding		[130]	
Recommendation		[139]	[140]
Architecture		[62]	[55]
Security Mechanisms			[141, 142, 102]

account the cost in the caching decisions, but not only. For instance, in the last subsection we refer to approaches that address cost reduction via other techniques, which are orthogonal to caching. We consider them, because they could be combined with our cost-aware caching to increase even further cost reduction. We also include in our discussion work that is not in the context of computer networks, but that can be easily applied to them, with the opportune modifications. In addition, we also consider works in which the notion of cost is not related to a monetary value (which is our target), because the techniques they propose still have some validity replacing their notion of cost with ours.

2.5.1 Cost-Aware Replacement Policies

Several cache replacement policies that take into account a cost associated to objects have been proposed in the literature. Already in the 90s, Young [121] devises "Greedy Dual" in the context of k-server problem, which can be seen as a generalization of the caching problem (see Sec. 1 of [143]). In particular, [121] focuses on the *weighted caching* problem, where the meaning of "weight" is "cost", whose meaning, in turn, is not precisely defined and is left as an abstract quantity. Observe that in the early 90s, caching was intended as "paging", i.e. moving pieces of data from the disk to the main memory, or as storing fonts in a printer, while network caching was not yet envisaged. A more efficient version of this algorithm called "GD-Wheel" has been recently published [122], in the context of memory-based key-value stores, which are used to store the results of some computation, e.g. the results of database queries. There, "object" means one of such results, and its cost reflects how difficult it is to compute it again. While the aforementioned works fall outside the computer networks research area, in the late 90s, Cao and Irani [119] extended GreedyDual and applied it to proxy caches, which we can intend as user-side proxy caches (Sec. 2.2.1) or transparent proxy caches (Sec. 2.2.3) or reverse proxy caches (Sec. 2.2.2). The cost can mean in their case the download latency, the object size, the congestion status of the link used to download the object or even the price paid to use that link.

The contribution of Part I distinctly differs from the above works. First, we specifically focus on the monetary cost of inter-domain traffic, providing results on the realistic saving of an ISP. Second, [121, 122, 119] propose replacement algorithms based on complex computations that would be impossible at line speed. On the contrary, in Chapter 4 we propose a metacaching policy that is lightweight and easily implementable in an ICN-router.

2.5.2 Cost-Awareness in CDN

In two papers [129, 27] published after our first articles on cost-aware caching [17, 16], the authors investigate cost and caching from an inverse viewpoint with respect to what we do in Part I. In their model, a CDN can store objects to different locations, each inside a different ISP. Different locations imply different cost, depending on the agreement between the CDN and the ISPs. Simplifying, the goal of the CDN is to find the placement that guarantees the minimum overall cost for the CDN while satisfying all users request. Differently from them, our goal is minimizing the ISP cost, instead of the CDN cost. While their CDN model implies "pieces" of cache distributed in different external ISP networks, we look only at the cache space inside one ISP and we aim to manage it in order to minimize the cost. From a technical point of view, the optimization model of [27] is a generalization of ours (Chapter 3). First, we only consider cost, while they also consider user utility (even if the exact meaning is not clearly specified). Second, while in the model we provide in Chapter 3 we only need a decision variable to establish whether to cache an object or not, in [129] we must also decide in which node, i.e. in which ISP networks, to cache.

2.5.3 Economic Considerations in ICN, P2P and the Cloud

The economic implications of caching are considered by [101, 123, 124, 125, 104], in the ICN context. In more detail, [101] models the economic incentives of different network players (including regulators) to deploy (or support) distributed

ICN storage. In [104], the economic feasibility of ICN is evaluated, contrasting it with client-server, peer-to-peer and CDN models.

Roberts et Al. [133] explore the memory-bandwidth tradeoff: installing a cache infrastructure implies a certain additional capital expenditure increase and, at the same time, an operational expenditure decrease, thanks to the inter-domain traffic reduction. Their goal is to understand what is the amount of cache to deploy in order to reduce the overall cost, fixing LRU as replacement policy and `LCE` as metacaching policy (Sec. 2.3.2). This investigation is extended and applied specifically the mobile networks in [134], also considering `Prob` (Sec. 2.3.2). A recent work [120] investigates the same trade-off in a cloud environment, where storage resources can be allocated or deallocated on the fly. Contrarily to [133, 120, 134], we assume a fixed amount of cache space and we study cache policies that can exploit it efficiently, which makes our study complementary.

The most relevant works for what we investigate in Part I are [123, 124, 125], concerning ICN and [126], concerning P2P. Again, our approach is different. First, while they consider ISPs as atomic entities and focus on an inter-ISP view, we study the problem of cost saving from an intra-ISP perspective, and propose a scheme that ISPs can use to manage their own networks. Additionally, [124, 125] investigate new pricing models for ICN networks without looking at the caching policy to be used. Our research is orthogonal: we focus on a novel cache strategy and we study its impact on the pricing model currently used in the Internet. Authors of [126, 123] study how ISPs can reduce the transit traffic by sharing their cache content exploiting settlement-free peering links: while this reduction is blindly computed among all transit links, in Part I we instead explicitly exploit price heterogeneity, which we show to have an important impact in practice.

2.5.4 Other Techniques to Reduce Cost

For the sake of completeness, we point out that other techniques, different from caching, have been proposed to reduce the inter-domain traffic cost. Some of them [62, 144, 145] are related to live streaming. Since this traffic is not cacheable by nature, we do not report on that. We just observe here that the control architecture of [62], composed of local and global control, is interesting for us, as we can embed our caching decisions in it, as part of our future work.

Castro et Al. [127] propose a collaboration between ISPs to aggregate their traffic in order to benefit from economies-of-scale. While in our work we consider ISPs as autonomous and non explicitly cooperating entities, an interesting research direction could be to evaluate the margin for improvement we could have by combining cost-aware caching with a collaboration like the one proposed in [127].

2.6 Delivery of Video Content

Video streaming over the Internet has become a mainstream research topic in recent years: as such, several works focused on the problem of ensuring an efficient *video streaming* in communication networks. Similarly, *caching* has attracted a surge of attention in recent years through popularization of Content Distribution and Information Centric Networks. However, as already discussed, there is still lack of a unified viewpoint to alleviate the huge increase in required bandwidth and to guarantee satisfactory Quality of Experience (QoE) for users.

2.6.1 Classic Caching versus Classic Video Delivery Impairment

We recall that in what follows we will implicitly refer to Adaptive Streaming (see Sec. 1.2.2), as our video delivery method. To confirm the impairment stated above, classic caching directly applied to video streaming not only is inefficient, but may even be harmful, since it may cause rate fluctuations [135], which hamper QoE. Another example of classic caching vs. classic video streaming impairment is given in [138], which, by means of trace-driven simulations, finds that an ICN cache deployment may not always lead to relevant QoE improvement in video delivery. Yet we argue that such results understate the benefits achievable via caching, since they are obtained by applying representation-blind policies, which consider homogeneous objects, all encoded at a single quality. In Part II, we instead leverage the possibility to serve different quality representations to maximize user satisfaction, respecting capacity constraints.

2.6.2 Control Mechanisms

QoE maximization has been tackled in the classic literature on video delivery [136, 135, 137] by proposing control mechanisms that intelligently share bandwidth among different users. While typically this kind of work assume that the source of transmission is unique [137], more relevant for our work are recent papers that take into account the existence of multiple sources (like caches and repositories), like [135, 136]. However, these works evaluate control algorithms under a given content allocation, whereas in Part II we look for the allocation guaranteeing the best QoE.

2.6.3 Optimization Approach

Authors of [5] consider caching of videos in a heterogeneous network, assuming that users can specify the minimum video quality they are willing to accept and the network provider goal is to minimize delay and cost while providing at least that quality. Our viewpoint is different, since we directly measure user satisfaction in terms of quality provided, rather than delay, and our goal is not just to satisfy a minimum requirement but to send videos at the maximum possible quality. A similar viewpoint is adopted in [144], which however does not take caching into consideration, since it focuses on live streaming The closest work to ours is perhaps [4], which employs caching, transcoding and routing functions to minimize the networking cost in a video distribution context. A two-step iterative approach is proposed, where, first, storage and computing resources are allocated optimally, then the routing is configured in the second phase. However, the model does not explicitly account for the utility perceived by users downloading different video representations, which is the focus of our study in Part II.

Maille et Al. [27] explore the consequences of replica placement on the QoE. Their scenario includes one CDN and two ISPs and three CDN cache locations, one inside each ISP and another, intermediate, outside but close to both. QoE can assume 3 values: by assumption, if an object is cached inside an ISP, the users of that ISP will enjoy a high QoE, while QoE is medium if the object is retrieved from the intermediate cache and basic if it is not cached and it must be retrieved from the origin server. The authors do not consider the bandwidth needed for a transmission as well as the link capacity constraints. On the contrary, in our model (Chapter 5) QoE, which we represent as user utility, depends on the bit-rate that we are able to send to users, which is constrained by the link capacities available in the network. Thanks to the realism of our description, we can show that user utility not only depends on which node we cache, as in the assumptions of [27], but also on the representation selection, the routing and the topology.

2.6.4 Alternatives to Adaptive Streaming

In our work, we only consider Adaptive Streaming, as defined in Sec. 1.2.2. For the sake of completeness we briefly report on some alternatives, which, to the best of our knowledge, are not widely deployed in production networks, mainly due to their complexity.

Some work has evaluated the benefits of using layered video coding, e.g. Scalable Video Coding (SVC), with caches. With this coding, each chunk of video is composed of a base layer and then other additional enhancement layers. The base layer is sufficient to reproduce the video at low quality. If more bandwidth is available, the client can download also the enhancement layers obtaining a better quality. The results presented in the position paper [146] show the advantage of SVC over Adaptive Streaming in terms of cacheability. SVC is also considered by [5], while [130] introduces a new layered video encoding. Observing that layered video coding still have a high complexity which slows down its adoption, differently from [146, 130, 5], our enhancement is obtained using the currently most deployed technology, i.e. Adaptive Streaming, in which there are different alternative representations of the same video, at different quality, and one of them must be served to the users. Moreover, the context of our model is a multi-AS environment, where the capacity of multi-hop paths limits the rate of transmission (thus, the served quality), whereas in the wireless context considered by [130] the limitation is due to the channel condition.

Another alternative to adaptive streaming is *on-the-fly transcoding* which allows to store just one of the available representations of a video and then transform it to a lower quality, if it is more suitable for transmission. Since this transformation is computationally expensive, such approaches have been investigated by the research community [147, 148], but are difficult to implement in real networks. To the best of our knowledge, big Content Providers and Content Delivery network still use Adaptive Streaming, i.e. they "pre-encode streams at various bitrates applying optimized encoding recipes" [149].

Although SVC and on-the-fly transcoding show potential for future development, they are not widely deployed in the current Internet. For this reason, we just focus in this thesis on Adaptive Streaming.

2.6.5 Impact of Different Quality Representations

The fact that a single video can be represented at different qualities has an important impact on users' experience, which [131] and [76] study in a CDN and wireless scenario, respectively, investigating what is the subset of video quality levels to make available in order to maximize QoE. Both make some simplifications of the network settings. The former only considers the capacity limitedness of the link connecting the user to the ISP, while we consider the capacity of all the links inside the network. Therefore, while the topology is ignored in [131], we take it into account and we are able to evaluate the instantiation of a feasible path, i.e. a path between the user and the copy of the requested object, in which the capacity constraints of all links are met. This evaluation is not possible in the model of [131]. On the other hand, we observe that [131] takes into account other details that we could also embed in our model as future work, like the capacity constraint of the link connecting the user to the ISP and the heterogeneity of devices from which users consume content. As for [76], it only considers one cache and one video.

We overcome the limitations of the work above by considering in Part II realistic networks, taking into account the limited storage at each node and the limited bandwidth on each link. More importantly, we assume that the set of the available quality levels is already established and look at the problem from a network viewpoint seeking to select the representations to store among the available ones.

Also Mukerjee et Al. [62] take into account the fact that a video can be represented at different bit-rates and propose an architecture to distribute content in a CDN topology. As we do in Chapter 5, they aim to route videos in order to respect link capacity constraints while maximizing the overall video bit-rate (which translates in video quality). However, they do not consider caching, which is at the core of our study, and they suppose that users specify the desired bit-rate, while in Part II users only request the video and the network tries to serve it at the best quality possible.

2.6.6 Client-Based Decisions versus Network-Based Decisions

From a technical point of view, current video delivery control algorithms require the client to take all the decisions, e.g. the quality representation to download, based on its measured metrics. Kleinrouweler et Al. [150] and references therein pinpoint the limits of this set-up, due to the limited view of the network that a client can have. Authors of [150] propose a *Network-Assisted* Adaptive Streaming, in which network elements communicate with the clients to help them in taking their decisions. In line with these findings, we advocate giving the entire control of these decisions to the ISP, since it can more easily know the state of the network and the resources available. This is not unrealistic since, in either case, users do not make any explicit choice most of the time [151], so that the selection mechanism, be it done in the Web browser of their personal device, or at the proxy in the ISP premises, is completely transparent to them. To support the plausibility of this switch, we point out that a recent IETF draft [139, 152] describes a *Server and Network Assisted DASH (SAND)*, currently in an experimental phase, in which video-aware network elements take the decisions and give directives to users, accordingly. In line with our findings, moving from the client to the network itself the responsibility to choose how to satisfy user requests is expected to be beneficial, particularly in presence of caches.

2.7 Caching Encrypted Content

The problem of permitting ISP to cache despite the fact that content is encrypted by CPs, which we tackle in Part III, has not largely been investigated in the literature. It has been mainly addressed by security mechanisms (Sec. 2.7.1). Another branch of work relevant to the problem investigates, in a larger sense, how to share the management of a cache system between the main actors in content delivery, namely CPs, CDNs and ISPs, which we discuss in Sec. 2.7.1. We also refer to some work close to ours as far as the methodology is concerned in Sec. 2.7.3

2.7.1 Security Mechanisms

In order to contrast our work of Part III with a broader literature and with current state of the art, it is useful to abstract it in the following terms: we want a *cache owner* to be able to cache even if the content and the requests are known only by the respective CPs. In our case, the cache owner is the ISP, while in the CDN environment it is a CDN [153] and in some work a cloud provider [141]. Note that the problem remains conceptually the same in the three cases. In the current Internet, the problem is partially solved by *Digital Rights Management* (DRM) technologies. CPs distribute to the cache owner infrastructure objects encrypted with a symmetric key. Then the key is distributed only to the authorized users, in a secure way. These techniques are used nowadays by the main CPs (Netflix, Spotify, etc. [153]), but have several issues:

- *Additional messages issue.* They typically require different messages between the user and the CP, which slows down the delivery.

- *Request visibility issue*: They also require the cache owner to read and understand the names of the objects requested by users, which we already explained to be a sensitive information for CPs, which would hardly be willing to provide it to ISPs. Therefore, our goal is to keep opaque not only the objects, but also the requests.

A similar approach is presented in [141], in which a CP arranges its users in groups, each group having particular access rights. A symmetric key is associated to each group, with which the CP encrypts content before sending it to the cache owner. This symmetric key is also distributed to the members of the group via some secure channel. As a consequence, the cache owner cannot read the content. When an authorized user requests for an object, it receives an encrypted copy from cache, if present, and can decrypt it with the symmetric key. The problem that arises with this technique is that a new symmetric key must be issued every time a user joins or leaves a group. Moreover, all the cached objects that were encrypted with the old key, must be encrypted again. Another issue is the request visibility issue already mentioned above.

Interesting, though not much extended, a branch of literature aims to provide confidentiality and trackability in ICN. The mechanism proposed in [142] requires the user to contact the cache owner to request for an object, then the cache owner to contact the CP to ask if the user is entitled to receive it. These communications must occur at every download. It should be clear that the same issues underlined above raise.

Mangili et Al. [102] devise a specific encryption mechanism to be implemented by CPs. In addition to the previous underlined issues, the authors also show a trade-off between the security strength and the cache efficiency: it is impossible to have high hit-ratios while assuring the maximum trackability and confidentiality. Moreover, the user is forced to receive all the chunks of the object before decrypting it. Therefore the mechanism is effective for downloading software or big pieces of data, but is not trivially applicable to video delivery, which, by the way, represents most of today traffic (Sec. 1.2.2). Indeed, users start to watch videos immediately after some chunks are received, without waiting to download them all, which is not possible with the mechanism of [102].

In the mechanism proposed by [154], to assure an object is only accessed by a user, it is encrypted with its public key. If the authorized receivers are more than one, a key pair is associated with this group of users and the object encrypted with the public key of the group. In addition to the request visibility issue, it is clear that managing the key pairs for all the possible groups is very complex, particularly considering that such groups are volatile, with new users continuously joining and other leaving.

2.7.2 ISP/CDN/CP Responsibility Share in Cache Management

A set-up similar to what we consider in Part III can be found in a recent work [97], which proposes to share a cache managed by an ISP among different CPs in order to achieve fairness. Partitioning of the storage is done using a pricing scheme based on the value that each CP gives to cache space. Unlike [97], our proposed algorithm aims to maximize the cache hit rate, and does not involve payments, which may make its adoption less controversial considering the disputes among ISPs and CPs in recent years [34]. To the best of our knowledge, our work is the first to propose an effective way to maximize the efficiency of an ISP-managed cache, by partitioning it among multiple CPs, without being aware of the content that the CPs serve and without involving payments.

Closer in scope to ours are worth mentioning a set of recent works that target ISP/CDN/CP cooperation [128, 132, 55, 155, 34]. These not only show that ISPs have strong incentive in investing in caching to reduce the traffic on their critical paths, but also show that the other Internet actors, i.e. CPs and users, would benefit from ISP in-network caching. The game theoretical study in [128] shows that caches are inefficient when operated by CPs, since CP content placement and ISP traffic engineering are often not compatible. Solutions are proposed in [132, 155], which however require ISPs to share with the CP confidential information, such as topology, routing policies or link states, and as such are arguably highly impractical, since ISPs are typically not willing to disclose information about the state of their network, i.e. congestion, available bandwidth, etc. [34] Conversely, [55, 34] foster an ISP-operated cache system, but requires the ISP to be able to observe every object requested by the users, which is arguably equally impractical since CPs purposely hide this confidential information via HTTPS. In contrast with these previous works, our solution does not yield to any leaking of business critical information. Furthermore, our solution is not limited to a single CP, unlike [132].

Finally, from a technical viewpoint, our work is aligned with recent industry efforts in the Open Caching Working Group (OCWG) [140]. The mission of the OCWG is to develop standards, policies and best practices for a new layer of content caching within ISP networks, which can coexist with HTTPS and provide shared access to storage for many CPs. Our work fits the OCWG requirements and as such is, we believe, of high practical relevance.

2.7.3 Work Related to our Methodology

As concerns the methodology, our partitioning algorithm is based on stochastic optimization. A stochastic optimization based algorithm, although different from ours, is also used in [93] to partition a CPU cache among competing processes. Clearly, the cache workload created by CPU jobs, the cache size and all the other characteristics of the CPU cache scenario are completely different from the network cache case we target in this manuscript. This makes their algorithm inapplicable to our study, as we realized in an early stage of our investigation, simulating it and remarking that the time needed to converge to a good result was intolerably long.

Part I

Cost-Aware Caching

Introduction to Cost-Aware Caching

In this part, we investigate caching strategies, which we define “cost-aware”, aimed at optimizing a very relevant and practical objective for ISPs, i.e. inter-domain traffic cost minimization. We contrast them with the classic strategies aimed at maximizing hit ratio.

In Chapter 3 we formulate two optimization problems under ideal assumptions, namely a problem whose objective is to maximize hit ratio and a problem aiming at minimizing cost. Numerical results show that there is a trade-off between the two objectives and that, as a consequence, classic caching techniques, aiming at the first objective, fail to achieve all the potential cost-saving that caching can bring.

In Chapter 4 we remove the idealized assumptions above and we propose an online distributed policy, implementable in real networks, that approaches the optimal cost-minimizing solution. We give a probabilistic model of the policy. By means of large scale simulation, we compare the achieved benefits with respect to the state of the art classic caching techniques, which are blind to cost, and with respect to the theoretical optimal cost-minimizing solution found with the optimization. We also evaluate the robustness of our online policy when changing scenario, we study the sensitivity with respect to internal parameters and we discuss implementation constraints.

Overall, we find that embedding cost-awareness in network caching brings a sizeable saving for ISPs.

We summarize in Tab. 2.3 the notation used throughout this part.

Most of the content of this part is extracted from [17, 16, 12].

Table 2.3: Summary of the notation used in this part.

Variable	**Meaning**	**Place of definition**
Sets:		
$\mathcal{L}$	Set of external links	
$\mathcal{O}$	Catalog	
Parameters of the optimization model:		
λ_i	Request rate for object i	(3.23)
K	Cache size	Sec. 3.4.2
$\pi^{(l)}$	Price paid by the ISP for every object crossing the external link l	Sec. 3.1
$\Xi_i^{(l)}$	Binary variable indicating whether object i is retrievable through link l	Sec. 3.1
Decision variables:		
$\lambda^{(l)}$	Rate of requests crossing external link l	(3.2)
$f_i^{(l)}$	Fraction of demand for object i directed to external link l	Sec. 3.1
π_i	The price of the link which gives access to object i	(3.12)
x_i	Binary variable denoting if we cache the object i	Sec. 3.1
Parameters of the evaluation scenarios:		
α	Skew parameter of the Zipf popularity distribution	(3.23)
$\mathbf{s} = (s^{(1)}, \ldots, s^{(N)})$	Split vector; $s^{(l)}$ is the fraction of objects that is behind external link l	Sec. 4.3.2
π	Price ratio, i.e. the ratio between the expensive and cheap links	Sec. 3.4.2
Metrics:		
h_i	Cache hit ratio for object i	Sec. 4.2
H	Hit ratio	(3.1), (3.20)
C	Cost	(3.3), (3.15)
CF	Cost fraction	(3.26)
PS, AS	Potential Saving and Achieved Saving	(4.10), (4.11)
Online algorithm parameters:		
$\psi(\cdot), \bar{\psi}$	Popularity based module in the metacaching policy and its default value	Sec. 4.1.1, (4.1)
$\beta(\cdot), \bar{\beta}$	Price based module in the metacaching policy and its average	(4.2), (4.3)
$\overline{\psi\beta}$	Average acceptance ratio	(4.4)
M	Normalization factor	(4.5)
W	Simplification factor	(4.7)

Chapter 3

Optimal Cost-Aware Caching

We start our investigation of the potential cost-benefits of caching leveraging optimization techniques. Despite the ideal assumptions, they consent to formalize the problem of cost in a systematic way, to understand the structural differences in the caching choices that occur when minimizing cost instead of classically maximizing hit ratio and to seize the potential benefit that a cost-aware policy can bring in theory.

The chapter is organized as follows. In Sec. 3.1 we describe the system model and in Sec. 3.2 we provide two Integer Linear Programs (ILPs), one maximizing hit ratio and the other minimizing cost. Among all the possible optimal solutions, we focus on a subset with simple structural properties in Sec. 3.3, which are computable in linear time by simple greedy algorithms. Finally, we discuss numerical results obtained by the model to show the cost-hitratio trade-off and the potential benefit of cost-aware strategies.

3.1 System Model

Fig. 3.1 illustrates the model adopted in this chpater: an ISP serves a rate λ_i of requests for an object i belonging to the catalog $\mathcal{O}$. Tab. 2.3 summarizes the notation used throughout this chapter. To serve these requests, the ISP may need to retrieve the object through one of its available *external links* (we use the set $\mathcal{L}$ to denote them), connecting it to other ISPs, or CPs or CNDs, paying a related cost.

In case the ISP is operating caches, some of these requests can be served within the ISP network. Denoting with x_i a binary variable indicating if object i is cached, we can define the *hit ratio*, i.e. the rate of requests served by cache,

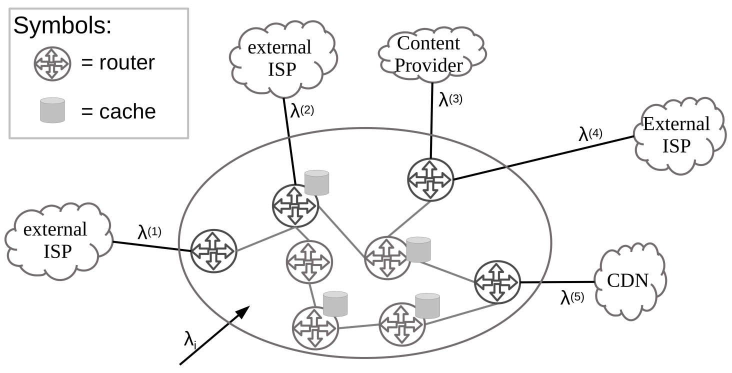


Figure 3.1: Model of the ISP. The ISP is connected to third party networks through external links $l_1, l_2, \ldots, l_5$.

as:

$$H \triangleq \frac{1}{\sum_{i\in\mathcal{O}} \lambda_i} \sum_{i\in\mathcal{O}} \lambda_i \cdot x_i. \tag{3.1}$$

The maximization of the cache hit ratio, irrespective of the link through which the requests exit the ISP network, has usually been the objective of ICN research. In contrast, we argue that the primary goal of an ISP is to minimize the cost associated to external links' utilization. In other words, by installing a limited amount of cache storage within its network, the ISP may not want to blindly maximize the hit ratio independently of the object cost: rather, the ISP aims at caching objects that lead to a larger cost saving, i.e., objects that are accessible through the most expensive links.

To capture this objective, we first specify that the demand for an object i that is not cached, may be split in different flows, each one sent to one of the external links that give access to that object. A binary variable $\Xi_i^{(l)}$ indicates if object i is reachable through the external link l. We denote with $f_i^{(l)}$ the fraction of demand directed to the external link l. It follows that the load on l is (using unit object size for the sake of simplicity in the formulation):

$$\lambda^{(l)} = \sum_{i\in\mathcal{O}} \lambda_i \cdot (1 - x_i) \cdot f_i^{(l)}. \tag{3.2}$$

Hence, unlike current literature that evaluates the cache vs. bandwidth tradeoff within ISP boundaries [156], we instead do not associate any cost to the traffic on the internal links, as in [126, 125], since we focus on the inter-domain traffic cost, assuming capacities of internal links are sufficient to carry the required traffic, as in [25, 125, 124, 126, 123]. Moreover, as [123, 126], we do not consider the cost of cache installation, because (i) it is a capital expenditure that is not related to the inter-domain traffic cost, which is the subject of our investigation, and (ii) we start from the assumption that a fixed amount of cache is already installed in the ISP network and we quantify the benefits achievable switching from classic cost-blind cache policies to our proposed cost-aware mechanism, with no difference in the cost of deployment of the cache infrastructure, thanks to the simplicity of our solution.

The 95% charging model is the most widely used among ISPs (see [157, 158, 159]): traffic volume on a provider link is sampled every period, e.g. every five minutes, and the 95th percentile of the samples, computed over a larger time span, e.g. one month, is charged. However, as usually assumed in the literature, our traffic model is stationary, i.e. its statistics do not change over periods, and thus the 95% charging model is equivalent to the proportional one, in which the cost incurred in retrieving objects from a certain link is directly proportional to the traffic volume flowing on that link. Therefore, we will use the proportional charging model, conforming to literature (see [157, 25, 126, 125, 124, 123, 133]). Ultimately, the cost of inter-domain traffic jointly depends on the traffic load $\lambda^{(l)}$ crossing any given link l and the link price $\pi^{(l)}$:

$$C \triangleq \sum_{l\in\mathcal{L}} \pi^{(l)} \cdot \lambda^{(l)} = \sum_{l\in\mathcal{L}} \pi^{(l)} \sum_{i\in\mathcal{O}} \lambda_i (1-x_i)\cdot f_i^{(l)}. \tag{3.3}$$

We argue that an interesting objective for ISPs is to minimize the above overall cost (3.3), considering not only the popularity λ_i but also the link prices $\pi^{(l)}$, as opposed to maximizing the overall hit ratio in a cost-blind fashion – that we show to be contrasting objectives in Sec. 3.4.4.

3.2 Optimization objectives and constraints

An ISP may design the cache system in order to minimize the cost C or to maximize the hit-rate H. We model these two conflicting goals with two different multi-objective Mixed Integer Linear Programs (MILPs), which we call respectively $MaxHit$ and $MinCost$:

$$\textsc{MinCost}: \ \min\,[C, 1-H] \tag{3.4}$$

$$\textsc{MaxHit}: \ \min\,[1-H, C] \tag{3.5}$$

Both are subject to the following constraints:

$$\sum_{i\in\mathcal{O}} x_i \le K \tag{3.6}$$

$$f_i^{(l)} \le \Xi_i^{(l)} \qquad \forall i\in\mathcal{O}, l\in\mathcal{L} \tag{3.7}$$

$$f_i^{(l)} \le 1 - x_i \qquad \forall i\in\mathcal{O}, l\in\mathcal{L} \tag{3.8}$$

$$\sum_{l\in\mathcal{L}} f_i^{(l)} + x_i = 1 \qquad \forall i\in\mathcal{O} \tag{3.9}$$

$$f_i^{(l)} \in [0,1] \qquad \forall i\in\mathcal{O}, l\in\mathcal{L} \tag{3.10}$$

$$x_i \in \{0,1\} \qquad \forall i\in\mathcal{O} \tag{3.11}$$

where the input parameter K is the cache storage, i.e. the number of objects that can be cached inside the ISP. The decision variables are x_i and $f_i^{(l)}$. Equation (3.6) represents the limitedness of the cache space, (3.7) imposes to retrieve objects only from links that give access to them, (3.8) allows only demand for non-cached object to be sent to an external link, while (3.9) forces all the demand for a non-cached object to be satisfied, either by the cache or through external links. The last two constraints specify the type of the decision variables.

Note that the *order* of the functions we minimize in the two problems is important: in MinCost the primary goal is the minimization of the cost while the secondary goal is the maximization of the hit-ratio (that we express as the minimization of the miss-ratio, $1 - H$). On the contrary, in MaxHit the sequence of objective functions is inverted. Note also that it is not relevant for our problem to look at how the ISP cache system is deployed. We are only interested in knowing whether an object is cached or not. It makes no difference to us whether the cache system is composed by one cache server only or a network of caches and, in the latter case, the placement of the replica has no impact on the model.

3.3 Greedy Algorithm

Before providing the greedy algorithm that solves the optimization problem, which is the main goal of this section, we will characterize the set of optimal solutions.

We define the price of an object i as:

$$\pi_i \triangleq \min\{\pi^{(l)} | \Xi_i^{(l)} = 1\}, \forall i \in \mathcal{O} \tag{3.12}$$

i.e. price of the cheapest among the links that give access to that object.

Proposition 1. *The optimal solution of* MinCost *sends all the request miss stream of an object to the cheapest links among the ones that give access to that object. In other words, at optimum*

$$f_i^{(l)} > 0 \iff \pi^{(l)} = \pi_i, \forall i \in \mathcal{O}, l \in \mathcal{L} \tag{3.13}$$

Moreover, there always exists a solution of MaxHit *that satisfies (3.13).*

Proof: First, we observe from (3.1) that the hit ratio does not depend on $f_i^{(l)}$. Let us consider a solution of MinCost . From (3.3), the overall cost can be expressed as:

$$C = \sum_{i \in \mathcal{O}} \lambda_i (1 - x_i) \cdot \sum_{l \in \mathcal{L}} \pi^{(l)} f_i^{(l)}. \tag{3.14}$$

Let us suppose, by contradiction, that there exists a link l' such that $\pi^{(l')} > \pi_i$ and $f_i^{(l')} > 0$. It is straightforward to note that moving the fraction of demand

$f_i^{(l')}$ from l' to one of the cheapest links giving access to i, which have price π_i, decreases the overall cost by leaving the hit ratio unchanged and still satisfying the constraints (3.6)-(3.11). Therefore, the solution considered at the beginning of this proof cannot be the optimal. The same arguments apply on MAXHIT . ■

By simple calculation, we obtain the following corollary.

Corollary 2. *Both in* MINCOST *and* MAXHIT *, at optimum, the cost is:*

$$C = \sum_{i \in \mathcal{O}} (1 - x_i) \cdot \lambda_i \pi_i. \tag{3.15}$$

Proof: Thanks to the previous proposition, we know that we can compute the cost at optimum in both MINCOST and MAXHIT , assuming that (3.13) holds. Therefore, from (3.15), the cost becomes

$$C = \sum_{i \in \mathcal{O}} \lambda_i (1 - x_i) \cdot \pi_i \cdot \sum_{l \in \mathcal{L}} f_i^{(l)}. \tag{3.16}$$

Applying the constraint (3.9), we obtain the result. ■

The previous corollary means that, as long as we send all the requests that are not satisfied by the cache to links among the cheapest ones that give access to it, the value of $f_i^{(l)}$ makes no difference. This allows us to remove this decision variable from our formulation. The optimization problems can thus be simplified as 0-1 Integer Linear Programs (ILPs):

$$\text{MINCOST}: \ \min\,[C, 1 - H] \tag{3.17}$$

$$\text{MAXHIT}: \ \min\,[1 - H, C] \tag{3.18}$$

Both are subject to the following constraints:

$$C = \sum_{i \in \mathcal{O}} (1 - x_i) \cdot \lambda_i \pi_i \tag{3.19}$$

$$H = 1 - \frac{1}{\sum_{i \in \mathcal{O}} \lambda_i} \sum_{i \in \mathcal{O}} \lambda_i \cdot x_i \tag{3.20}$$

$$\sum_{i \in \mathcal{O}} x_i \leq K \tag{3.21}$$

$$x_i \in \{0, 1\} \qquad \forall i \in \mathcal{O} \tag{3.22}$$

where the only decision variable is x_i, which represents caching.

The solution of these problems is easily found by the simple greedy algorithms described in the following propositions.

Proposition 3. *A solution of* MINCOST *is found by caching the K objects with the highest product $\lambda_i \pi_i$. If several objects have the same product, we break the ties by caching the ones with the highest λ_i.*

Proposition 4. *A solution of* MaxHit *is found by caching the K objects with the highest request rate λ_i. If several objects have the same request rate, we break the ties by caching the ones with the highest $\lambda_i \pi_i$.*

The time complexity to select those K objects is $O(|\mathcal{O}|)$ in both cases, where $|\mathcal{O}|$ is the total number of objects. Considering the MinCost problem, for instance, [1] it suffices first to select the object whose product $\pi_i \lambda_i$ is the K-th largest. To this aim, we can use the algorithm of Sec. 9.3 of [160], whose complexity is $O(|\mathcal{O}|)$. Then, for each object, we insert it into the cache only if its product $\pi_i \lambda_i$ is larger than or equal to the K-th object's. This requires also a time $O(|\mathcal{O}|)$.

3.4 Numerical results

We now provide and discuss numerical results, obtained through a GNU Octave [161] implementation of our greedy algorithm, aimed at both (i) quantitatively assessing the economic savings that an ISP can potentially get by targeting MinCost instead of classically targeting MaxHit and (ii) understanding structural differences in terms of cached content.

3.4.1 Content Popularity

We identify the objects with the natural numbers $i = 1, 2, \dots |\mathcal{O}|$. As in most literature, we leverage the *Independent Reference Model (IRM)*, first introduced by [162]. It can be expressed (see Sec. 2.1 of [163]) by saying that the arrival process of the requests of a certain object $i \in \mathcal{O}$ is a Poisson process with a rate λ_i. We emphasize that, under this model, each request is independent from the previous. Content popularity obeys to the Zipf distribution [164, 165, 166]:

$$\frac{\lambda_i}{\sum_{j \in \mathcal{O}} \lambda_j} = \frac{1}{H^{(\alpha)}_{|\mathcal{O}|}} \cdot \frac{1}{i^\alpha}, \forall i \in \mathcal{O} \tag{3.23}$$

where α is a positive real parameter, called *exponent*, and $H^{(\alpha)}_{|\mathcal{O}|}$ is called generalized harmonic number, i.e. $H^{(\alpha)}_{|\mathcal{O}|} = \sum_{j=1}^{|\mathcal{O}|} \frac{1}{j^\alpha}$, whose role in the equation above is to make the sum of (3.23) over all objects equal to 1.

3.4.2 Scenario

While our framework is general and can be applied to every type of ISP, in this section we focus on a Local ISP with three external links: a peering link l_{free}, a "cheap link" l_{cheap} and an "expensive link" l_{exp} with respective prices $\pi^{(l_{free})} = 0$, $\pi^{(l_{cheap})} = 1$ and $\pi^{(l_{exp})} = \pi$, where $\pi \in \{1, 2, \dots, 10\}$ denotes the

[1] We borrow the idea from `http://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/`

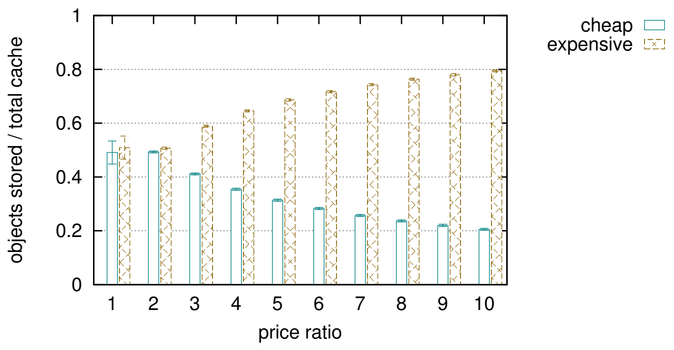


Figure 3.2: Relative size allocated by MinCost to cheap and expensive objects. The 95% confidence intervals are shown.

price ratio between the cheap and the expensive link. [2] Note that the scenario is also equivalent to having more than three links, whose prices fall in the set $\{0, 1, \pi\}$, since the problem formulation (3.17)-(3.22) does not change in this case.

We consider a realistic Internet-scale catalog [110], consisting of $|\mathcal{O}| = 10^7$ objects whose popularity obeys to Sec. 3.4.1 with exponent values $\alpha = \{0.8, 1.2\}$ [108], while the total cache budget is $K = \{10^2, 10^3, 10^4\}$ objects. For each configuration we generate 40 scenarios. In each scenario, objects are assigned to each external link with 0.5 probability, so that an object can be reachable through more than one link. If there are unassigned objects, each of them is uniformly assigned to one of the links. We then compute the 95% confidence intervals shown in the plots. In each experiment, we calculate the cost-optimal configuration using the MinCost algorithm, obtaining an optimal total cost C_{cost} (and a corresponding hit-ratio H_{cost}). We then calculate the hit-ratio-optimal configuration using the MaxHit algorithm, obtaining an optimal hit-ratio H_{hit} (and a different cost C_{hit}). We finally define the *hit-ratio loss* as

$$(H_{hit} - H_{cost})/H_{hit} \tag{3.24}$$

and the *cost saving* as

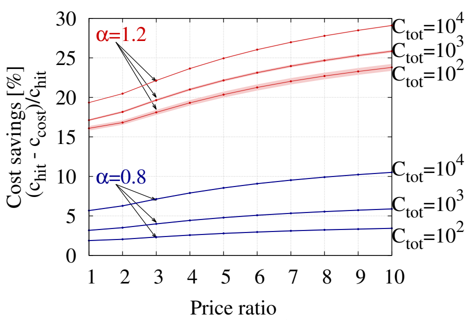
$$(C_{hit} - C_{cost})/C_{hit} \tag{3.25}$$

respectively. The former expresses how much the hit-ratio degrades in the MinCost with respect to the MaxHit configuration. The latter, instead, gauges the cost savings of MinCost that are lost in the MaxHit solution.

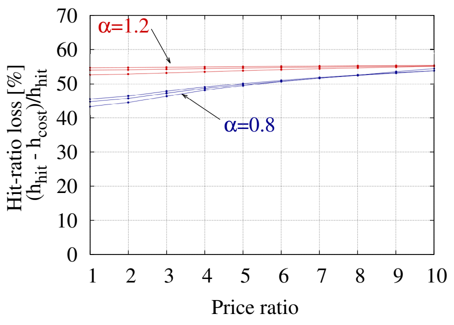
3.4.3 Price Breackdown

We analyze how the content cached by MinCost varies depending on the price ratio π. As expected, MinCost does not cache objects retrievable through the

[2]The range of π is in accordance to Sec. 3.3 of [167], Sec. 4 of [157] and Sec. 2 of [127].



(a) Relative cost saving of MIN-COST vs MAX-HIT. The (narrow) 95% confidence band for each curve is plotted.



(b) Relative hit-ratio loss of MIN-COST vs MAX-HIT. For each α, a band including the 95% confidence interval of all curves is plotted.

Figure 3.3: Numerical results: (a) cost saving, (b) hit-ratio loss and (c) cache size as a function of the price ratio π

free peering link l_{free}, while it caches more the objects that are more expensive, i.e. the ones that lie behind more expensive links. Fig. 3.2 shows the percentage of cache budget used to store cheap objects (retrievable through l_{cheap} but not l_{free}) and expensive objects (only retrievable through l_{exp}). The overall results depend on a combination of orthogonal factors such as prices, content popularity and content availability behind each link.

At first, observe that each object may in theory be reachable through both links. Proposition 1 ensures that, in this case, the copy on the most expensive link is never used. This implies that the cheap link l_{cheap} is more exploited since it is used to retrieve the objects that it is the only one to provide, as well as the objects that are provided by both links. On the contrary, l_{exp} is used only to retrieve objects that it is the only one to provide. As a consequence, more objects are retrieved through l_{cheap} rather than l_{exp}. For this reason, for small values of price ratio, meaning that object prices are close to be homogeneous, a non-negligible cache size is used for cheap objects, showing, as expected, that cache sizing is impacted by content availability more than by prices. On the contrary, when prices' heterogeneity grows, its influence prevails: the cache allocated to expensive objects noticeably stands out. In this case, the cache allocation is driven by prices more than by content availability. As for the content popularity impact, while Fig. 3.2 only shows the results for $\alpha = 1.2$, we verify that for $\alpha = 0.8$, the trend remains the same, but the difference between the cache space allocated to cheap objects and the expensive objects is more evident: the less the skewness in the popularity, the more the cache sizing is impacted by the prices.

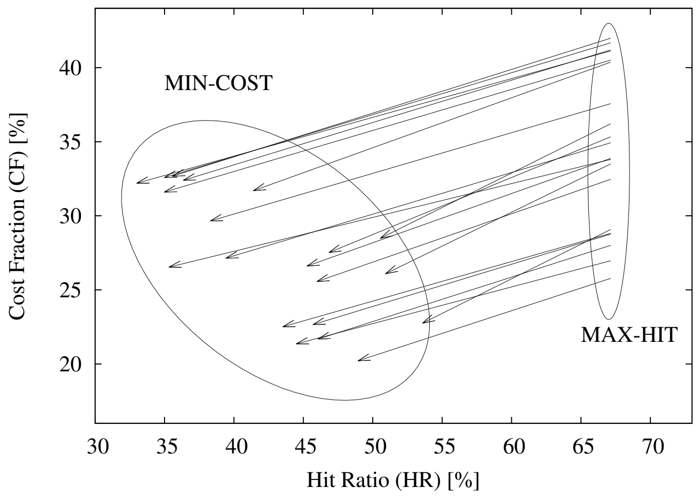


Figure 3.4: Hit ratio vs. cost trade-off. Values are numerically computed over 20 instances of a scenario with catalog size $|\mathcal{O}| = 10^5$, Zipf exponent $\alpha = 1$, objects uniformly distributed across the free, cheap and expensive links, price-ratio $\pi = 10$ and overall cache storage is $K = 10^3$. Each arrow is relative to a single instance and shows how cost fraction and hit ratio change when switching from MaxHit to MinCost .

3.4.4 Hit Ratio vs. Cost Tradeoff

In Fig. 3.3 we depict the cost saving and the hit ratio loss defined in (3.25) and (3.24). From Fig. 3.3-(a) we observe that, as the price ratio increases, optimizing the cost enables cost savings of up to 30%. As an expected side effect, this induces a loss of caching efficiency in terms of cost-blind metrics (i.e., the hit-ratio) up to 60%. Otherwise stated, if an ISP wants to get economical benefits from the use of caches, hit-ratio optimization must be a secondary objective: indeed, a loss of hit-ratio efficiency can translate into significant gains in terms of cost savings. Moreover, this holds especially when the external links are very heterogeneous in terms of prices (which we expect to be the common case), since the cost savings increase as the price ratio increases.

Note that both cost savings and hit-ratio loss consistently increase with the popularity skew (from $\alpha = 0.8$ to 1.2). In addition, the cost savings also increase consistently as the total cache size K increases from $K = 10^2$ to $K = 10^4$ (see the different curves in the plot). Intuitively, this means that when the cache budget is high, the operator has more freedom in its allocation, and the potential gains are larger; as a consequence, choosing the right cache strategy is crucial for attaining these potential gains. On the other hand, the hit-ratio loss is quite unaffected by K (Fig. 3.3-(b)).

To better illustrate the trade-off, we introduce the *cost fraction* CF of a caching strategy (either MinCost or MaxHit) as the ratio between the cost incurred by that strategy and the cost incurred by a cache-less system in the same scenario. Based on (3.15), we can easily compute:

$$CF \triangleq \frac{\sum_{i\in\mathcal{O}}(1-x_i)\cdot\lambda_i\pi_i}{\sum_{i\in\mathcal{O}}\lambda_i\pi_i} \tag{3.26}$$

The MaxHit vs. MinCost trade-off is illustrated in Fig. 3.4, for the scenario described in the caption. As previously observed, there is an inherent variability across instances of the same scenario, which is tied to the different breakdown of the objects among external links in each instance. Results clearly show that cost fraction and hit ratio are conflicting goals: specifically, the arrow implies that a cache fraction loss is necessary to achieve a cost reduction gain in the corresponding random instance of the considered scenario.

Chapter 4

An Online Distributed Policy for Cost Reduction

In the previous chapter we showed the cost vs. hit-ratio trade-off and the benefits of targeting cost-minimization instead of classic hit-ratio maximization. Nonetheless, the results were obtained through an optimization model under idealized conditions, first of all assuming that we know in advance the request rates λ_i of all objects i, and thus that it is possible to choose *offline* the objects to cache based on this knowledge. Unfortunately, this information is not available in reality and the greedy algorithm described in Sec. 3.3 is not applicable. For this reason, we propose in this chapter an on-line caching policy which approaches the optimal MinCost solution inferring the popularity of the objects online, with no a priori knowledge. Moreover, in the previous chapter we coarsely considered the caching system as a whole, without taking into account that it can be composed of different caches, deployed in different nodes and connected in a certain topology. In other words, we were only interested in establishing whether to cache or not an object in such caching system, without looking at where to store it. What we propose here is a distributed policy that is run by each node, autonomously and only leveraging local information, and we study the overall cost saving it brings. Our policy is essentially a metacaching policy (see Sec. 2.3.2) that we call *Cost-Aware* (`CoA`) policy.

The chapter is organized as follows. In Sec. 4.1, we describe our metacaching policy. In Sec. 4.2, we give its probabilistic model and evaluate its accuracy comparing it with simulation results. In Sec. 4.3 we simulate `CoA` to find the benefits achieved and comparing them with the theoretical optimal bounds found in the previous chapter. In Sec. 4.4 we show that performance is robust under different scenarios. Finally, in Sec. 4.5, we discuss the limitations that must be faced in a real implementation and evaluate their impact in the performance. Tab. 2.3 summarizes the notation used throughout this chapter.

The notation used in this chapter is included in Tab. 2.3.

4.1 The Metacaching Algorithm

We propose a novel Cost-Aware (`CoA`) design to achieve significant cost reduction, which we illustrate with the help of Fig. 4.1. As explained in Sec. 2.3.2, any new object arriving at a caching node is either cached or discarded, according to a *metacaching* policy (Sec. 2.3.2); in the first case, a *replacement* policy is triggered to select a previously cached object to be evicted. We inject cost-awareness in the metacaching policy. The motivations behind this choice will be clearer after having described our design and will be discussed later in this section.

4.1.1 Algorithm Parameters

Intuitively, to reduce costs, a cache has to not only store the most popular objects (which results in *caching efficiency*) but also and especially those that are obtained through the most expensive links (which results in *cost reduction*). Otherwise stated, the aim of cost-aware caching is to bias the caching process toward more expensive objects. However, it is not to be forgotten that, beyond the price of individual links, content popularity still plays a paramount role. Indeed, popularity and cost factors are independent and may even conflict: e.g., caching expensive but unpopular objects may not bring effective cost reductions while, on the other hand, caching cheap but very popular objects may be worthwhile. This is due to the fact that the cost depends on the product of price and popularity $\lambda_i \pi_i$ (3.15). Therefore, our goal is to consider price differences, but still differentiate between popular and unpopular objects. We suppose in this chapter that each object is retrievable through one and only one external link and that the price of the object corresponds to the price of that link. [1] The probability that, considering a generic object, it lies behind link l is $s^{(l)}$.

For this purpose, we design a modular metacaching policy, which is the composition of a popularity-based module and a price-based one, represented by the functions $\psi(\cdot)$ and $\beta(\cdot)$, respectively. The composition of the two modules is achieved via product of the two functions, i.e. a new object is accepted with probability $\psi(\cdot)\beta(\cdot)$. This composition permits to jointly weight popularity and price. $\psi(\cdot)$ can be any of the classic metacaching policies in the literature. We consider a constant

$$\psi(i) = \bar{\psi}, \forall i \in \mathcal{O}. \tag{4.1}$$

We design the function $\beta(\cdot)$ whose specific role is to weight price, biasing the acceptance toward expensive objects, as follows:

$$\beta(i) = \frac{M}{\sum_{l \in \mathcal{L}} [\pi^{(l)}]^\kappa} \cdot \pi_i^\kappa \tag{4.2}$$

The parameters M and κ have the following meaning:

[1]In case an object is potentially retrievable through more than one external link, we will consider only the cheapest one, ignoring the others, according to what we have found in Proposition 1.

- The constant M is set such that the average value

$$\bar{\beta} \triangleq \frac{\sum_{i\in\mathcal{O}} \beta(i)}{|\mathcal{O}|} \tag{4.3}$$

is 1. This guarantees that the average acceptance ratio is

$$\overline{\psi\beta} \triangleq \frac{\sum_{i\in\mathcal{O}} \beta(i) \cdot \psi(i)}{|\mathcal{O}|} = \bar{\psi} \cdot \bar{\beta} = \bar{\psi} \tag{4.4}$$

This means that the average accetance ratio is not modified by function $\beta(\cdot)$. Otherwise stated, the cache accepts, on average, the same fraction of objects as without the cost-aware module $\beta(\cdot)$, with the only difference that it preferentially stores the expensive ones. Additionally, this ensures that convergence rates of `Prob` and `CoA` are the same. In the single cache case, the normalization factor can be computed as:

$$M = \frac{\sum_{l\in\mathcal{L}} [\pi^{(l)}]^\kappa}{\sum_{l\in\mathcal{L}} s^{(l)} \cdot [\pi^{(l)}]^\kappa}. \tag{4.5}$$

- The exponent $\kappa > 0$ is used to tune the relative importance of popularity vs. price in the decision: indeed, the larger κ, the larger the skew toward expensive objects, while for $\kappa < 1$ the importance of price in the decision diminishes.

We observe here that classic metacaching policies approximate MaxHit (3.18), trying to infer λ_i by means of function $\psi(\cdot)$ in order to cache the (locally) most popular objects. `CoA` approximates MinCost (3.17), trying to infer $\lambda_i\pi_i$ by means of composition $\psi(\cdot)\beta(\cdot)$ in order to cache the objects that would generate the highest expenditure. For simplicity of notation, by plugging (4.5) into (4.2) we can rewrite the `CoA` function as

$$\psi(i)\beta(i) = W\pi_i^\kappa \tag{4.6}$$

where

$$W \triangleq \frac{\bar{\psi}}{\sum_{l\in\mathcal{L}} s_i \cdot [\pi^{(l)}]^\kappa} \tag{4.7}$$

is a constant that depends on both uniform probabilistic decisions (numerator) as well as on object cost (denominator).

At network set-up time, we estimate the constant W. Then, every time a new object i crosses a cache, we consider its price π_i and we cache it with the probability computed in (4.6).

4.1.2 Design Choices and Properties

We now motivate our design choices and pinpoint the peculiarities of our metacaching policy.

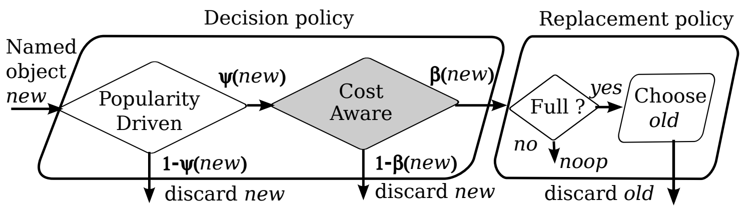


Figure 4.1: Cost-aware caching design, plugged within the metacaching policy of the caching component.

Metacaching vs. Replacement policies. We motivate now why we introduce cost-awareness in the metacaching policy rather than in the replacement policy. First, a properly tuned metacaching policy avoids the proliferation of irrelevant content along multiple caches, which would happen in case any new content were systematically accepted in the cache, like with Leave a Copy Everywhere (`LCE`), and which would lead to an excessive number of repeated evictions. Therefore, deterministic [81, 88] or probabilistic [87, 88, 168, 109] metacaching policies, different from `LCE`, are preferable. By extension, it is better to bias the acceptance toward more expensive objects in the cache, than to bias the *replacement* process toward cheaper objects a posteriori: in the latter case, each cache should keep additional state information of the cached objects (i.e., the price metadata), which it would need to manage at line speed (e.g., perform complex computations that take into account the price of all the cached objects, to select the cheapest one to evict). On the contrary, a cost-aware *metacaching* strategy, like the one we propose, is simpler to implement since it is lightweight and stateless (as price-related information can be inserted in some packet header field once it enters the ISP boundary and exploited independently by any cache along the path), allowing the rest of per-object operation to remain simple (e.g., Least Recently Used or random eviction policies).

Implicit distributed coordination. In order to efficiently exploit the total cache budget contained in a network of caches, a form of coordination is required, to prevent, for example, a node to store objects already stored by some other neighbors. Contrarily to mechanisms realizing explicit coordination by message exchange over the control plane [115, 114], which has the downside of complexity and communication overhead, our mechanism achieves distributed coordination with implicit coordination. In other words, in our approach no information is exchanged over the control plane, but rather a minimum amount of information –i.e., a price indication– is carried via packet headers directly in the data plane.

In practice, only *border routers*, i.e. the ones that are in front of the external links, know the link through which objects enter the ISP domain, and can thus tag the packet with a price indication. All the other caches along the path then take independent caching decisions based on the price information tagged by border routers. This price indication represents a negligible overhead, since it is marked only once and it travels together with the object, requiring the modification of only few bits of the header, as we will show in Sec. 4.5.

No additional cost. From the simplicity of our design, it follows that deploying `CoA` does not imply higher installation and operation costs than a classic caching policy (e.g., `LCE`+LRU). Therefore, the whole saving in the operational costs comes for free, i.e., it does not require an increase in the capital expenditure, as is often the case.

4.2 Che's Approximation Model

Abstract models of caches and network of caches have been present in the literature since many years [169, 170, 171, 172, 173, 174, 175], under the IRM hypothesis (Sec. 3.4.1). In recent years, a theoretical cache model, called *Che's Approximation* [176, 163] has imposed over the others thanks to its simplicity. Initially proposed to describe a simple `LCE`+LRU (Sec. 2.3.2) cache, it has been extended to the case of `Prob` and other cases by Leonardi et Al. [88]. We extend their work to model our Cost-Aware Caching.

We first restrict our attention to the subset $\mathcal{O}' \subseteq \mathcal{O}$ of objects having a chance to be cached, i.e. the objects whose price is non-zero, ignoring thus all the objects retrievable through a free link. By definition, the probability that `CoA` accepts an incoming object $i \in \mathcal{O}'$ in the cache is $\psi(i)\beta(i) = W\pi_i^\kappa$. Considering a single `CoA` cache of capacity c and whose incoming requests respect the Independence Reference Model (IRM), the hit probability for object an $i \in \mathcal{O}'$ is:[2]

$$\begin{aligned} \mathbb{E}(h_i) &= \frac{W\pi_i^\kappa \cdot (1-e^{-\lambda_i T_c})}{e^{-\lambda_i T_c} + W\pi_i^\kappa \cdot (1-e^{-\lambda_i T_c})} \\ &= \frac{1-e^{-\lambda_i T_c}}{1-e^{-\lambda_i T_c}\left(1-\frac{1}{W\pi_i^\kappa}\right)} \end{aligned} \tag{4.8}$$

where the characteristic time T_c is computed as in [176] by imposing that $\sum_{i\in\mathcal{O}'} \mathbb{E}(h_i) = c$. Notice that (4.8) degenerates into the Che's approximation of `Prob` (formula (5) of [88])) for $\kappa = 0$, i.e., when cost information is ignored. Clearly, for objects $i \in \mathcal{O} \setminus \mathcal{O}'$ retrievable through a free link, we instead have $\mathbb{E}(h_i) = 0$. The overall hit-probability, i.e. the expected hit ratio, can then be obtained over the whole catalog as:

$$\mathbb{E}(H) = \frac{1}{\sum_{j\in\mathcal{O}} \lambda_j} \cdot \sum_{i\in\mathcal{O}} \lambda_i \cdot \mathbb{E}(h_i) \tag{4.9}$$

whose numerical solution is depicted in Fig. 4.2 alongside that of `LCE` [176] and `Prob` [88] models. As for `LCE` and `Prob`, comparison against simulation exhibit an excellent match (for `CoA`, we additionally remark for both model and simulation the variability tied to the catalog split early noted in Fig. 3.4).

[2]The formula can be easily obtained like (5) of [88]

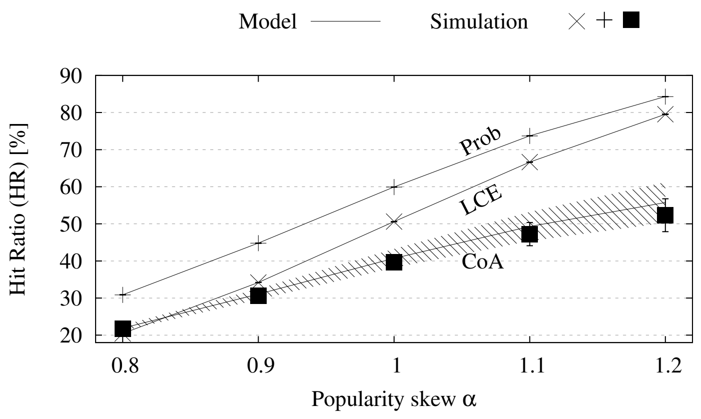


Figure 4.2: Model vs. simulation. Average hit ratio values over 20 instances of the default scenario (see Tab. 4.1) and 95% confidence intervals are depicted. The average acceptance ratio for both `Prob` and `CoA` is $\overline{\psi\beta} = \bar{\psi} = 1\%$ and $\kappa = 1$.

4.3 Simulation Results

We now assess the benefits of our proposed cost-aware design against cost-blind and cost-optimum caching strategies. On the one hand, comparison with cost-blind caching schemes can be viewed as a direct measure of the return of investment following caching deployment, and more precisely sizes the additional gain that can be attained by a cost-aware architecture. On the other hand, comparison with the optimal cost strategies allows us to gauge the extent of possible improvements in our design.

In this section we define the classic strategies that we contrast with `CoA` (Sec. 4.3.1), the simulation set up and the evaluation metrics (Sec. 4.3.2). We start our evaluation by considering the default scenario to cross compare, at a glance, all the above strategies (Sec. 4.3.3). We next expose the deficiencies of cost-blind strategies (Sec. 4.3.4) and finally verify that the `CoA` saving is consistent over real ISP topologies as well as in synthetic topologies generated with the Watts-Strogatz model (Sec. 4.3.5).

4.3.1 Terms of comparison

We contrast our design against several terms of comparison, which represent (i) cache-less systems, (ii) traditional caching schemes where price heterogeneity is not taken into account, (iii) ideal distributed metacaching policies with perfect knowledge of object popularity and (iv) MinCost achieving provably minimum cost, as in the previous chapter. As the following table illustrates, these different designs provide an exhaustive coverage.

	Cache-less	`LCE`	`Prob`	Ideal-Blind	`CoA`	Ideal-`CoA`	MIN-COST
Cost-aware				✓	✓	✓	✓
Implementable	✓	✓	✓		✓		

Cache-less system. As naive benchmark, we consider costs incurred by systems that do not employ any kind of caching. We point out that, other than providing an upper-bound of the costs incurred by the system, considering a common reference significantly simplifies the assessment of the relative improvement between more sophisticated strategies.

Cost-blind caching. Following our design, a natural term of comparison for cost-blind caching consists in considering state-of-the-art metacaching policies that ignore the cost of inter-domain traffic (i.e., equivalent to setting $\beta(\cdot) = 1$). The popularity-driven decision component could use `LCE`, equivalent to setting $\psi(\cdot) = 1$ or `Prob`, where $\psi(\cdot) = \psi_0$, ψ_0 being a fixed probability. To avoid cluttering the pictures, we do not include in our comparison other metacaching policies available in the literature, like Leave a Copy Down (LCD) [81] or policies based on distance [168], graph properties [109], etc. Indeed, `Prob` is a reasonable term of comparison, representative of state-of-the-art cost-blind decisions, since, as already explained in Sec. 2.3.2, it tends already toward the optimal hit-ratio-maximizing allocation, under opportune conditions. Comparison of `CoA` and cost-blind `Prob` strategies can be done on a fair ground, i.e., on the same number of acceptance decisions as stated in Sec. 4.1.1.

Ideal strategies. We additionally consider two strategies that have perfect knowledge of global object popularity, and that thus constitute an ideal term of comparison on the single-cache scenario. Yet, we point out that since this knowledge is not available in real situations, these policies cannot be implemented and are introduced here only as benchmarks.

Specifically, the decision whether to cache or not a new object is assisted by considering the eviction candidate, i.e. the object that would be removed from cache to make room for the new one. We assume the replacement policy is Least Recently Used (LRU), and thus the eviction candidate corresponds to the object that was requested least recently. The new object is accepted only if it is more "valuable" than the eviction candidate. This is expected to increase the value of the overall cache content over time. We implement two notions of value, depending on whether they limitedly consider object popularity, or jointly consider popularity and link price.

The ideal cost-blind strategy (*Ideal-Blind*) strives to keep only the most popular objects, deterministically admitting a new object i only if its arrival rate λ_i is greater than the one of the LRU eviction candidate.

The ideal cost-aware strategy (*Ideal*-`CoA`), instead, jointly considers the arrival rate and the price of the link through which the object has to be fetched. The aim is clearly to cache only the objects that are expected to provide the largest saving, which happens by admitting only objects whose $\lambda_i \pi_i$ is larger than that of the eviction candidate.

Optimal. We finally consider the minimal cost incurred by the ISP, obtained via the MinCost strategy. As opposed to the ideal strategies mentioned earlier, which take decisions on each packet arrival and are obtained via discrete event simulation, the MinCost strategy is obtained via a centralized optimal solution. A further difference between *Ideal-CoA* and MinCost is that MinCost basically pre-fills caches, so that it provides a lower bound to the ISP expenditures.

4.3.2 Scenario Settings and Metrics

To gauge the advantages introduced by `CoA`, we introduce two metrics beyond the cost fraction CF (3.26). Specifically, denoting with C^X the cost obtained with policy X, computed as in (3.15), we denote with *Potential Saving* (PS) the room for improvement of our proposal, i.e., the percentage of additional saving that could be leveraged by switching to an *Ideal*-`CoA` policy:

$$PS = \frac{C^{\texttt{CoA}} - C^{Ideal-\texttt{CoA}}}{C^{\texttt{CoA}}} \tag{4.10}$$

We further denote with *Achieved Saving* (AS) the percentage of expenditure which an ISP, currently running the state-of-the-art `Prob` policy, could save by switching to `CoA`:

$$AS = \frac{C^{\texttt{Prob}} - C^{\texttt{CoA}}}{C^{\texttt{Prob}}} \tag{4.11}$$

To perform a conservative evaluation, we need therefore to set the probability $\bar{\psi}$ in `Prob`, to avoid overestimating the achieved saving. We perform a preliminary calibration and identify in $\bar{\psi} = 1/100$ a value that is favorable to `Prob` in our scenarios, which we fix for the reminder of this work.

We point out that, since the average value of the price component is $\bar{\beta} = 1$ by design (see Sec. 4.1.1), the average acceptance ratio is $\overline{\psi\beta} = \bar{\psi} = \bar{\psi} = 1/100$ in both `Prob` and `CoA`. This ensures a *fair comparison*: indeed, the differences in performance cannot be ascribed to a different average cache admission probability, but are only due to cost-awareness, which is the main object of our investigation. Additionally, as the number of cache acceptance decisions taken by `Prob` and `CoA` is the same, their convergence speed is the same, despite the attained saving is different.

As in Sec. 3.4.2, we consider a Local ISP with three external links: a peering link l_{free}, a "cheap link" l_{cheap} and an "expensive link" l_{exp} with respective prices $\pi^{(l_{free})} = 0$, $\pi^{(l_{cheap})} = 1$ and $\pi^{(l_{exp})} = \pi$, where $\pi \in \{1, 2, 5, 10, 100\}$ denotes the price ratio between the cheap and the expensive link. With slight abuse of notation, we will refer to free, cheap and expensive objects, depending on the link behind which they lie. Consequently, we partition the catalog $\mathcal{O}$ in

Table 4.1: Parameters of the scenario. Bold values represent the default scenario used throughout the chapter.

Parameter	**#**	**Values**
Zipf skew α	3	0.8, **1**, 1.2
Price ratio π	5	1, 2, 5, **10**, 100
Catalog split **s**	13	$s^{(l)} \in \{\mathbf{1/3}, h/4 \vert h \in \{0,1,2,3,4\}\}$ $\sum_l s^{(l)} = 1$
System scale $K/\vert\mathcal{O}\vert$	5	$10^2/10^4, \mathbf{10^3/10^5}, 10^4/10^6,$ $10^5/10^7, 10^6/10^8$
Cache/catalog ratio $K/\vert\mathcal{O}\vert$	5	$\mathbf{10^3/10^5}, 10^3/10^6,$ $10^3/10^7, 10^3/10^8$

three subsets $\mathcal{O}_{free}, \mathcal{O}_{cheap}, \mathcal{O}_{exp}$ the subset of objects lying behind the cheap and the expensive link, respectively

To denote the breakdown of the objects across the three external links, we introduce the *split vector* $\mathbf{s} \triangleq (s_{free}, s_{cheap}, s_{exp})$, where $s_{free}, s_{cheap}, s_{exp}$ are the probabilities that, considering a generic object, it lies behind the free, the cheap and the expensive link, respectively. An important point is worth stressing: clearly, even in case that two partitions contain the same number of objects, e.g. $s_{cheap} = s_{exp}$), their aggregate request rates may differ, as objects have skewed popularity (i.e., $\sum_{i \in \mathcal{O}_{cheap}} \lambda_i \neq \sum_{i \in \mathcal{O}_{exp}} \lambda_i$). We cope with this imbalance of the aggregate link load resulting from a catalog split vector **s** by averaging results over multiple runs.

The parameters we will consider in the simulation campaign are listed in Tab. 4.1.

4.3.3 Comparison at a glance

In this section, we refer to the default single cache scenario (detailed values are highlighted with boldface in Tab. 4.1), setting $\kappa = 1$ and $\bar{\psi} = 1/100$. With the exception of the MinCost solution, which we compute numerically, all strategies are implemented in ccnSim. Results can be reproduced by running the scripts available at the ccnSim website [11].

In the following we report the average results with 95% confidence intervals gathered from 20 instances of scenario for each setting; the duration of each run is sized to have statistically relevant results, and statistics are computed only after the initial transient period needed for the cache hit metric to reach a steady state. We underline that, even under the same scenario, one instance of the scenario differs from the other, since each instance is generated using a certain seed and, at each seed, we have a different distribution of objects across the external links and a different request sequence (even if with the same statistical properties). To evaluate the cost-effectiveness achieved by each caching strategy, we consider the hit ratio and the cost fraction. Note that we already computed

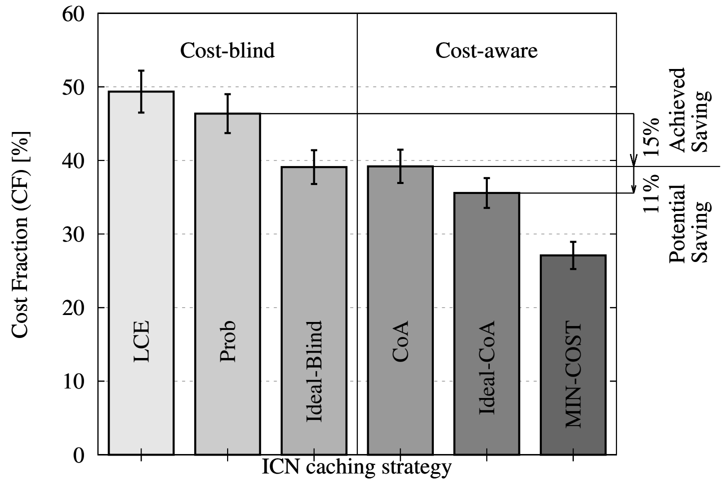


Figure 4.3: Benefits of cost-aware design. The cost fraction obtained by each strategy is reported. Achieved and potential saving (expressions (4.11) and (4.10), respectively) are annotated on the right y axis.

such quantities in (3.1) and (3.26) in previous chapter, directly using the values λ_i. On the contrary, referring to `LCE`, `Prob`, Ideal-Blind, `CoA` and Ideal-`CoA`, we compute those quantities in a different manner, without using λ_i directly, but counting the discrete events, i.e. the number of hits and misses, experienced during the simulation. In particular:

- The *hit ratio* H is the fraction of requests that are satisfied by the cache system.
- The *cost fraction* CF is the cost achieved by the strategy in question divided by the cost we would obtain without any cache in exactly the same instance of the scenario.

Fig. 4.3 shows, at a glance, the cost fraction for cost-blind (left bars) and cost-aware (right bars) strategies. Our strategy (`CoA`) brings sizable benefits over state-of-the-art cost-blind metacaching (about 15% of achieved saving over `Prob`), matching the performance of the Ideal-Blind strategy. This means that, exploiting information already at hand, and that changes over relatively long timescales (i.e., the prices negotiated with different ISPs), can bring benefits that are at least as important as those relative to information that is highly volatile and harder to infer (e.g., object popularity).

To interpret the practical relevance of the `CoA` benefits, consider the case of an ISP in which a state-of-the-art caching system is already deployed, which is tuned in a cost-blind fashion to maximize hit-ratio. If the ISP decides to switch to `CoA` tuning, it will save about 15% of the inter-domain traffic cost,

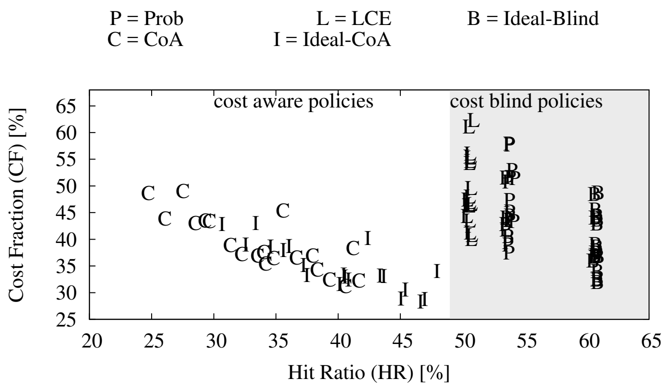


Figure 4.4: Comparison of cost-aware vs. cost-blind policies: Scatter plot of hit ratio versus cost fraction, confirming that higher cache hit ratio does not necessarily imply lower cost under a wider range of policies.

without facing any additional expense. Indeed, while the installation of the caching infrastructure implies a capital expenditure (CAPEX), our `CoA` mechanism consists in a simple tuning and does not require additional capex. Yet, `CoA` offers the ISP a consistent saving in the operational expenditure (OPEX), which becomes sizable as it accumulates over the years.

At the same time, considering the distance from Ideal-`CoA` to MinCost , we see that there is still additional room for improvement (11% of potential saving), which is however hard to obtain, as it would require knowledge of object popularity.

4.3.4 Root cause of cost saving

To understand the root cause of the performance gap, we extend the previous representation of the MaxHit vs. MinCost tradeoff depicted in Fig. 3.4, to include the `LCE` (L), `Prob` (P), `CoA` (C), Ideal-Blind (B) and Ideal-`CoA` (I) policies, which we represent with a capital letter in the scatter of Fig. 4.4. We generate 20 instances of the default scenario and run the different strategies on each instance. We observe that, despite the low hit ratio, cost-aware policies result in a lower cost fraction: this confirms that cost reduction does not only come from cache hit maximization, but is mainly due to price discrimination. This shows also that the tradeoff discussed when considering optimal strategies (Sec. 3.4.4) also holds in practical implementations. Similarly to Fig. 3.4, the dispersion in Fig. 4.4 is caused by the object-to-link mapping randomly generated for each instance of the scenario.

To further assess the impact of cost-aware caching on the network, in Fig. 4.5 we report the *normalized traffic load* of the free, cheap and expensive links, i.e. the number of requests flowing on the link in question, divided by the total number of requests coming from users. `CoA` and Ideal-`CoA` achieve structurally

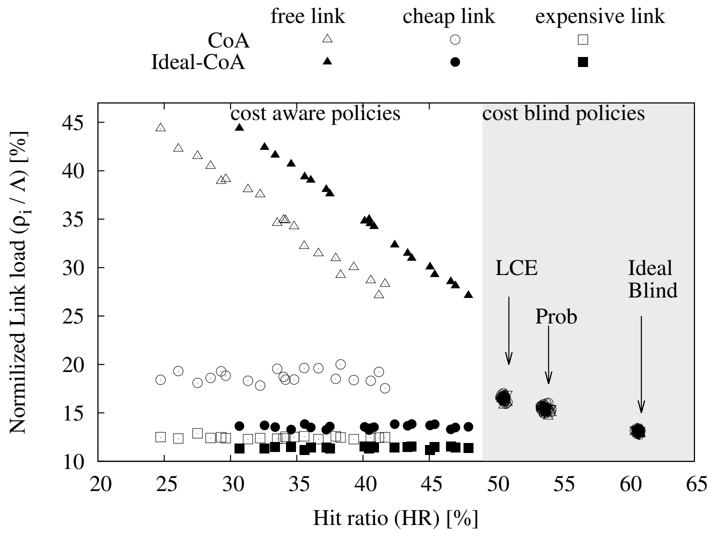


Figure 4.5: Scatter plot of hit ratio vs. load (normalized over the aggregate request arrival rate) on free, cheap and expensive links. Note that cost-aware policies differentiate loads on links with heterogeneous prices.

similar configurations; specifically, they reduce the load on expensive and cheap links (circles and squares in the figure), while increasing the load on the free link (triangles), since they both never cache free objects. Note that as the hit ratio decreases, the load on the free link increases while the loads on the cheap and expensive links are almost constant: this means that all the additional miss stream drains into the free link. Finally, Ideal-`CoA` exhibits better performance than `CoA`, in terms of both hit ratio and cost fraction, due to the perfect knowledge of object popularity.

While cost-aware policies differentiate link load based on link prices, cost-blind policies uniformly distribute the load, resulting in overlapping points in the scatter plot. Note that, while reasonable, this result is not straightforward and is due to the cache filtering effect: in other words, despite the load in a cache-less scenario would not be uniform due to the variability of the aggregated demand in each sub-catalog, the cache equalizes the miss-stream over these links. This is intuitive, since in a uniform scenario, links with higher demand (before caching) are those behind which the most popular objects are accessible, thus, they will be most benefited by load reduction thanks to caching.

To summarize, the price differentiation operated by cost-aware policies permits to cache only the objects that would result in a cost for the operator. This has two consequences: (i) it reduces cache efficiency in terms of hit ratio but, on the other hand, (ii) it limits ISP costs thanks to the diminished utilization of the costly links.

4.3.5 Performance on realistic network topologies

So far, we have analyzed the performance of a single cache operating with `CoA`. In this section, we show that cost reduction is consistent even in a distributed environment, consisting of a network of caches, each operating autonomously with `CoA`. We conduct a simulation campaign on both realistic (as in [108, 118]) and synthetic (as in [112]) network topologies (described in Fig. 4.7 and an example of which is depicted in Fig. 4.6) where, at each run, we attach the free, cheap and expensive links to randomly selected nodes. We allocate the total cache space uniformly among all routers (as in [82, 108]) and use the default values for the other parameters (bold values of Tab. 4.1). We consider two forwarding strategies:

- *Shortest Path Routing (SPR)*: the path traversed by a request is the shortest between the request source and the egress router. The egress router is the one attached to the external link which gives access to the requested object.

- *ideal Nearest Replica Routing (iNRR)* [108]: if there exists a cache that is storing the requested object, and it is closer than the egress router, the request is sent to that cache.

With SPR, an interest can be matched only with the copies cached in one of the nodes along the shortest path. Therefore, a content may be downloaded through an external link even if a copy is present inside the network, which happens whenever the cached copies lay off the shortest path between the requestor and the repository. Due to the increased redundancy, and reduced efficiency in using a fixed cache space budget, we expect cost reduction in the SPR case to be smaller than that estimated in the previous section on a single case scenario.

This limitation is overcome by iNRR [108], which is able to exploit all the copies stored in the network. Even though iNRR is ideal, since it would require the knowledge of the objects cached in all the nodes, it can be easily approximated [82] in caching and is thus worth considering. Opposite to the SPR case, we expect the iNRR cost reduction to be in line with the one estimated in the single cache case.

In complex topologies, interesting mutual effects among nodes arise, whereas they are not observable when considering a single node. In particular, in the distributed case there is a mismatch between the *global popularity* of any object vs. its *local popularity* which accounts only for the requests received for that object by a specific node. In particular, the local popularity of an object observed by a node depends on (i) the routing policy, since not all the requests pass through that node and (ii) the cache filtering effect, due to cache hit at neighboring nodes. It follows that Ideal-Blind and Ideal-`CoA` policies are not effective in these scenarios, as they base their decisions on global object popularity.

We therefore exclude Ideal policies from the analysis of this section, and limit our attention to comparing policies (namely, probabilistic cost-blind vs. cost-aware), under two routing schemes (namely, SPR vs. iNRR) on a range of

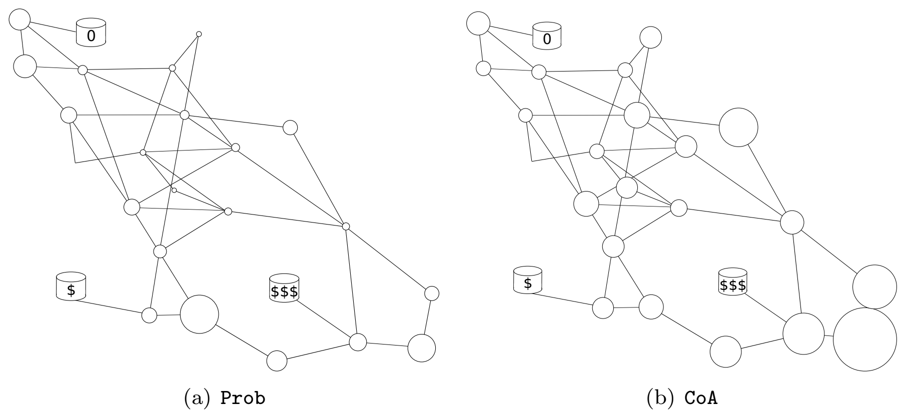
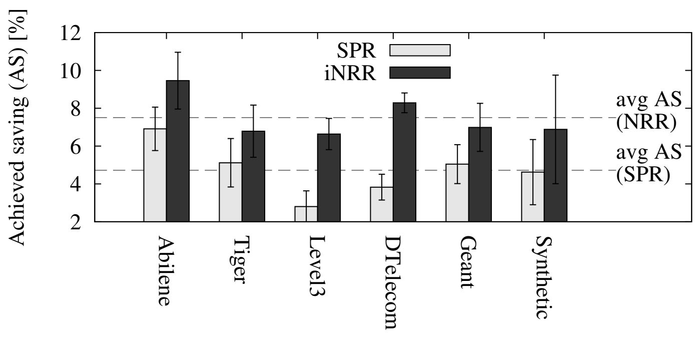


Figure 4.6: Geant topology. Node size is scaled based on their contribution to the overall saving when `Prob` (a) or `CoA` (b) is used. Objects are retrievable through the links connected to the nodes labeled with 0, $ and $$$, which are free, cheap and expensive, respectively.

both real and synthetic topologies. Fig. 4.7 illustrates the saving obtained in five real topologies and synthetic topologies, generated with Watts-Strogatz model, matching Geant's characteristics. The achieved saving amount to 7.5% (4.7%) on average with iNRR (SPR). These results suggest that the achieved reduction is consistent even in realistic networks of caches, the size of the reduction depending on the topology. The advantages of `CoA` over `Prob` also depend on the forwarding strategy and are more evident with iNRR, as expected. Another interpretation of the SPR vs. iNRR performance gap can be given anticipating that our sensitivity analysis will show gain to grow with the cache space (see Fig. 4.12 in Sec. 4.4.5): under this light, the gap follows from the fact that requests can leverage all the cache space under iNRR, while only the fraction of cache space included in the shortest path is exploitable with SPR.

Finally, in order to quantify the contribution of each node to the overall saving under the `Prob` vs `CoA` policies, we define the value of a cache node ν as $v_\nu = \sum_{i \in \mathcal{C}_\nu} \lambda_i \cdot \pi_i$, where $\mathcal{C}_\nu$ is the set of objects stored by node ν, which is clearly the cost absorbed by that node. Fig. 4.6 depicts the (rescaled) cache value averaged over 20 simulation runs. Note that, when `Prob` is used, the majority of nodes has a small cache value: on the contrary, `CoA` tends to equalize cache values, allowing each node to give a substantial contribution to the overall saving (despite topological constraints, e.g., nodes being closer to more valuable content, still have a clear impact).



#nodes	11	22	46	68	22	22
Avg node degree	2.5	3.6	11.7	10.4	3.4	4.0
Coeff. of variation of degree	0.2	0.2	0.9	1.3	0.4	0.4
Avg link propagation delay[ms]	11.3	0.1	8.9	17.2	2.6	2.6
Diameter	8	5	4	3	4	4

Figure 4.7: Achieved saving on different topologies. 95% confidence intervals are reported. The table reports the characteristics of the topologies.

4.4 Sensitivity analysis of cost-aware design

The previous sections have delved into benefits of cost-awareness into a sensible yet specific scenario. We now extend the reach of the above findings by showing that `CoA` benefits are robust and consistent in a wide range of conditions – Overall, we performed over 4000 simulation runs, accounting for $O(10^{10})$ requests.

Specifically, we perform a sensitivity analysis of scenario parameters that are external and, in Sec. 4.4.6, we show benefits to be smoothly varying with respect to internal `CoA` knobs, such as the κ parameter. We anticipate that our proposed Cost-Aware scheme provides a consistent and robust saving in all the considered network scenarios.

For what concerns evaluation scenarios, there are many factors that are unknown at best, which will likely change in unpredictable manner, and that are not under the control of either the manufacturers or the ISPs. We therefore perform a thorough sensitivity analysis of the `CoA` performance on scenarios other than the default one investigated earlier. Tab. 4.1 reports the parameter values we consider in this section. For the sake of simplicity, since `CoA` performance under state-of-the-art iNRR routing is consistent with that of the single cache scenario, in this section we limitedly consider the latter.

Clearly, each parameter concurs in determining the `CoA` performance: e.g., we expect the achieved saving to be marginal for very low skew values (α),

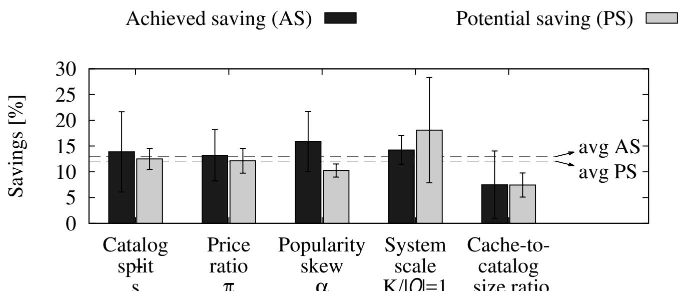


Figure 4.8: Robustness against external factors such as catalog split, price ratio, popularity skew, system scale, cache-to-catalog size ratio. Bars represent average and standard deviation of the achieved and potential saving over the full parameter space reported in Tab. 4.1.

or when most of the catalog is accessible only through the most costly link, or when the cache is too small, etc. The impact of these parameters is summarized in Fig. 4.8, which represents the mean value of the achieved vs. the potential saving (Sec. 4.3.2) and their standard deviation obtained by making the single parameters vary within their respective domains. We see that the gains resulting from biasing the cache decision policy along the cost dimension are consistent over all the parameter variations: on average, the achieved saving over `Prob` is 13%.

Hereafter, we investigate how each single parameter of the scenario impacts the `CoA` performance. If not otherwise stated, each configuration is obtained starting from the default one (see bold values in Tab. 4.1) and varying only the parameter under analysis. Each configuration is evaluated providing the mean value of saving and the 95th percentile over 20 runs.

4.4.1 Impact of catalog split

At each simulation run, we place each object behind one of the three external links (free, cheap or expensive), on a probabilistic basis with $s^{(l)}$ the probability that an object is assigned to link l. As a consequence, the catalog is split into free, cheap and expensive objects: we distinguish *pessimistic* scenarios in which at least half of the catalog is behind the expensive link, *optimistic* scenarios in which at least half of the objects are free, and *intermediate* scenarios.

Fig. 4.9 represents the cost fraction in 10 different scenarios, each characterized by a catalog split vector $\mathbf{s}$. The cost saving achieved by `CoA` over `Prob` is reported besides the arrows. As expected, cost-blind policies are insensitive to the catalog split, since they treat objects as they all had the same value. This is why in Fig. 4.9 their cost fraction is constant (with the exception of a slightly higher cost fraction in the pessimistic scenarios, which is due only to the fact that they are less favorable). On the contrary, the impact on cost-aware policies is evident. `CoA` performs better when a considerable part of the catalog is free

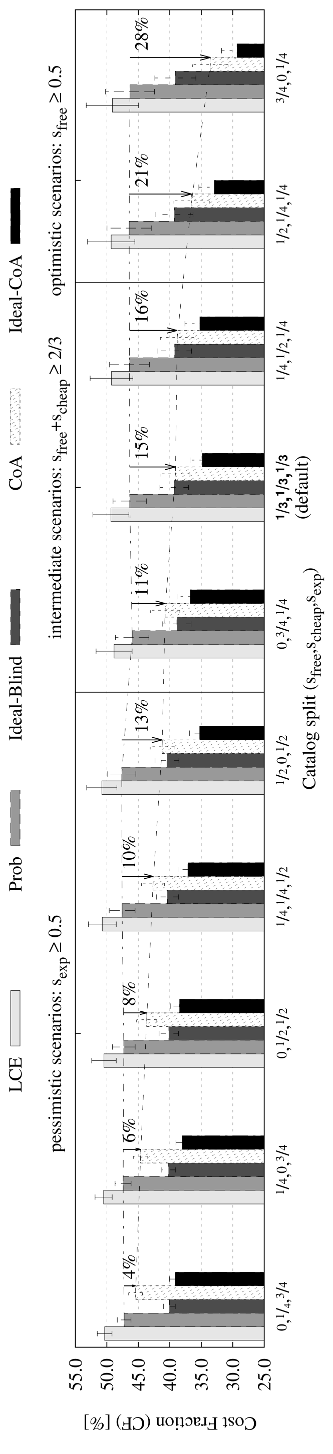


Figure 4.9: Impact of the catalog split: $(s_{free}, s_{cheap}, s_{exp})$ represent the probability that a content is retrievable through the free, cheap or expensive link, respectively. Arrows indicate the saving achieved by `CoA` over `Prob`.

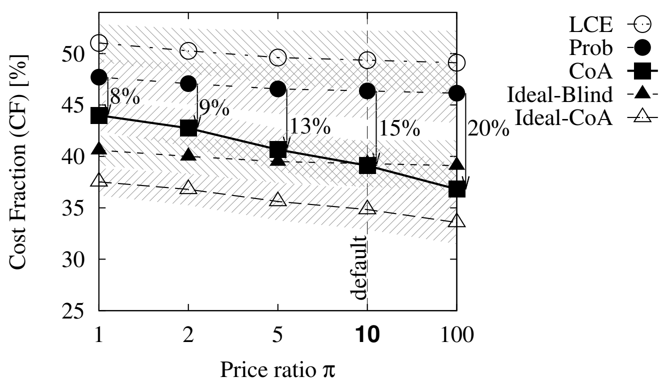


Figure 4.10: Impact of price heterogeneity on cost fraction. Arrows indicate the saving achieved by `CoA` over `Prob`.

or cheap: in this case, the achieved saving goes from 8% up to 27%. When half of the catalog is behind the expensive link, cost reduction is more modest, and this is due to the fact that there are inherently no gains to be exploited.

4.4.2 Impact of price heterogeneity

The price ratio $\pi = \pi_{exp}/\pi_{cheap}$ is the ratio of the expensive over the cheap link prices: the larger π, the higher the heterogeneity of the external link prices. We consider values of price ratios ranging from 1 to 10, in line with values reported by both [157] and [167], who gathered information from publicly available data and from interviews with operators, respectively. We plot the cost fraction in Fig. 4.10, where the arrows report the achieved saving of `CoA` over `Prob`. For $\pi = 1$, cheap and expensive links have the same price: therefore, cost-aware policies achieve cost reduction only by avoiding to cache free objects, while the other policies tend to blindly cache them. The cost reduction of cost-aware policies becomes more evident as price heterogeneity increases: while cost-blind policies are insensitive to price ratio, cost-aware ones leverage it. Contrarily to [111], we argue that to reduce the ISP costs it does not suffice to blindly reduce inter-ISP traffic across external links, since price heterogeneity plays an important role and must be exploited. In order to depict an asymptotic behavior of the cost saving, we include in our analysis a price ratio of 100, showing that, already for practical $\pi = 10$ values, our `CoA` proposal gets most of the asymptotic benefits.

In addition, we observe that, for high price ratios, `CoA` equals or outperforms Ideal-Blind. It is interesting to underline that this holds even if `CoA` requires only the knowledge of the objects price (which changes very slowly in time and is easily traced by ISPs, and thus of practical use) as opposed to Ideal-Blind

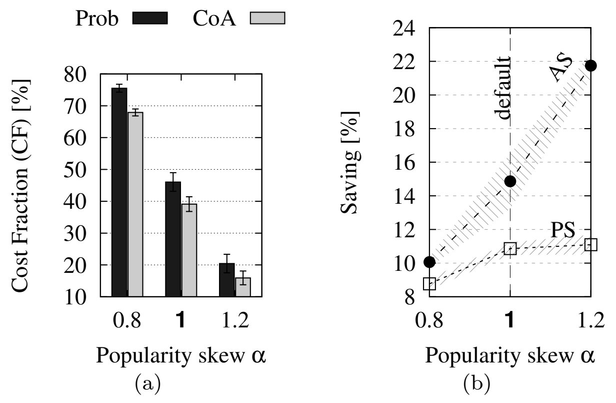


Figure 4.11: Impact of popularity skew: (a) Cost fraction of `Prob` and `CoA` and (b) achieved (AS) and potential (PS) saving, as defined in (4.11) and (4.10).

that requires a perfect and a priori knowledge of the popularity (which changes rapidly and is very difficult to infer properly, and thus impractical to exploit).

4.4.3 Impact of popularity skew

We study how cost reduction is impacted by the popularity skew of the catalog. We let the Zipf exponent α vary along the range of values that are reported in recent work employing measurement from either a global CDN [108] or a local PoP of an ISP [177]. As expected, increasing the popularity skew plays in favor of caching, i.e. both `Prob` and `CoA` reduce their cost fraction, which can be seen in Fig. 4.11-(a).

Nonetheless, `CoA` consistently outperforms `Prob`. Indeed, even if the cost fraction of `Prob` decreases for an increasing skew, the `CoA` saving over `Prob` increase further: this clearly emerges from Fig. 4.11-(b), and is due to the fact that the denominator of the achieved saving (4.11) becomes smaller. Additionally, we see that the potential saving saturates after $\alpha > 1$, meaning that `CoA` is able to efficiently take advantage of the favorable conditions to caching represented by the high popularity skew.

4.4.4 Impact of cache-to-catalog size ratio

We next verify gain dependency on the relative scales of the cache vs. catalog sizes. We fix the cache size $K = 10^3$ and make the catalog size vary in $\{10^8, 10^7, 10^6, 10^5\}$, thus getting, respectively, cache-to-catalog size ratios varying in the 0.01% to 1% range, in line with [108, 88, 82].

Results reported in Fig. 4.12-(a) show that the reduction achieved by `CoA` increases with the cache-to-catalog size ratio: this means that the larger the

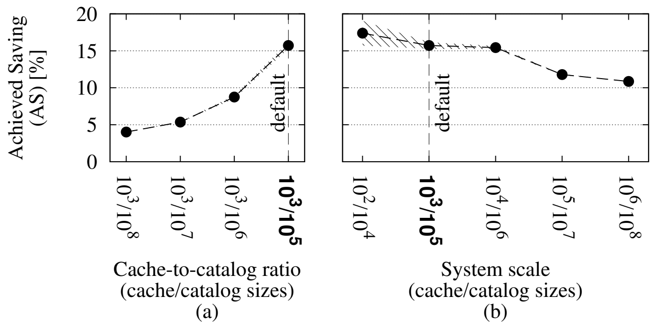


Figure 4.12: Impact of system scale and cache-to-catalog size ratio: Achieved saving of `CoA` over `Prob`.

cache budget available for the ISP, the more attention is worth paying to its management, as the attainable cost saving is larger. The iNRR gain over SPR routing policies is partly explained by the same observation.

4.4.5 Impact of system scale

Finally, maintaining the cache-to-catalog size ratio size fixed to the default value $K/|\mathcal{O}| = 10^{-2}$, we change the scale of the simulation by varying simultaneously the cache size and the catalog size. The considered catalog sizes are representative of content providers of different dimension, as Video on Demand services or Youtube, and are based on previous work in literature [178, 179]. Results summarized in Fig. 4.12-(b) show that the achieved saving diminishes as the scale increases: yet, from the smallest to the largest scale ($10^2/10^4$ to $10^6/10^8$), gains remain consistent (17% to 11%).

4.4.6 Impact of `CoA` Settings

As discussed earlier, for an efficient cost reduction the item worth should jointly weight popularity and price: the `CoA` parameter κ permits to tune this tradeoff giving more weight to popularity (low κ) or to price (high κ). It is thus important to perform a sensitivity analysis of κ, to assess to what extent its tuning is crucial for the correctness of `CoA` operation and to achieve the gain shown so far.

Fig. 4.13 illustrates the impact of κ on the achieved and potential saving, for different scenarios and price ratios. In particular, Fig. 4.13-(a) evaluates the impact in an optimistic scenario characterized by the prevalence of free objects $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$, a uniform scenario $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and a pessimistic scenario in which most of the objects are behind the expensive link $(\frac{1}{4}, \frac{1}{4}, \frac{1}{2})$. In Fig. 4.13-(b) the impact is measured varying the price heterogeneity, by letting π vary in $\{2, 10, 100\}$. Briefly, we observe that the value $\kappa = 1$ guarantees a good performance in all different conditions: indeed, (i) the achieved saving over `Prob` is close to the maximum value and (ii) the potential saving over Ideal-`CoA` is close to the minimum. In more details, from Fig. 4.13-(a) we observe that, even for small values of κ, price discrimination brings sizable gains over completely blind strategies. Second, the parameter κ effectively tunes between three regimes (a *mostly popularity-driven* regime, a *balanced* one and a *mostly cost-driven* regime). As expected, gains are larger in the balanced regime (highlighted in gray in the picture), as it is the one that better jointly weights popularity and price, better inferring $\lambda_i \pi_i$. Finally, while largest gains are achieved by $\kappa \approx 1$, we also gather that performance smoothly varies with κ, so that its setting is not critical. Similar considerations hold by fixing the catalog split $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and varying the price ratio in Fig. 4.13-(b).

4.5 Implementation Constraints

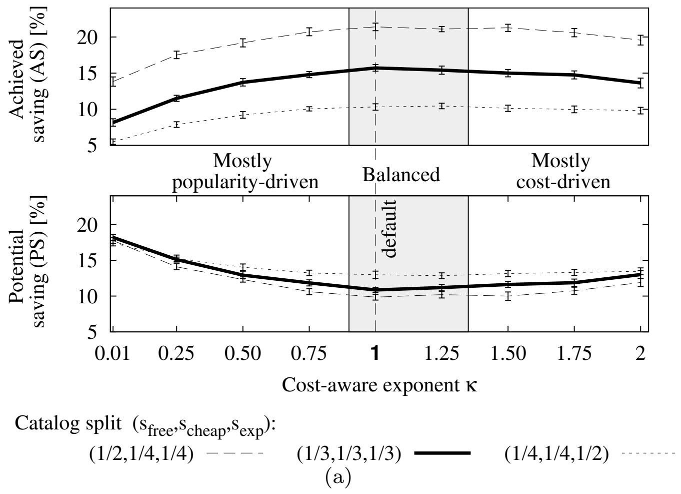
Our metacaching policy requires to tag incoming objects with the information about the price of the external link crossed by the object (Sec. 4.1.2). This tagging operation is performed by the border routers, i.e. the routers to which external links are attached.

We outline two possibilities to represent cost-related information in the data packets carrying objects: (i) to use a simple but rigid syntax, using a fixed-size field of a standard packet header format versus (ii) using a more complex but flexible syntax as Type Length Value (TLV) encoding.

Both implementations have pros and cons: experience with TCP/IP tells that while fixed-size fields are simpler (thus, faster) to handle, they also scale badly over time, and tend to become critical resources (e.g., IP TOS field). Moreover, while mechanisms to circumvent these limits exist (e.g., IP options), however they happen to be rarely used in practice. Conversely, flexibility (e.g., of TLV) comes at a price of increased complexity: historically, following the principle of pushing complexity to the edge, fixed framing has been preferred for lower layers of the protocol stack, which need to be treated within the network core, relegating syntactically more expressive formats to the application layer.

For our purpose, both solutions are in principle possible. For the sake of simplicity, so far we have implicitly assumed that border routers can tag packets with arbitrary information. However, this may not be true in practice, as the information bits available to express price differences may be limited. It follows that the architectural design should be stress-tested against such imposed limitations.

In order to do so, we set the link prices of the free, cheap and expensive link as $\pi_{free} = 0, \pi_{cheap} = \zeta, \pi_{exp} = 10$ and we make ζ vary in $1, 2, \ldots, 10$. Effects are expected to be non trivial: for instance, when a single quantization bit is used (binary decision), objects of the cheap link are not cached (as if they were attainable through the free link) when $\zeta < 5$, and are instead cached with the same probability of expensive objects when $\zeta \geq 5$. Additionally, the magnitude of the impact, and not only the frequency of errors in the decision process, also depends on ζ. We thus represent the average cost fraction loss (with standard deviation) in Fig. 4.14 for different amount of quantization bits and ζ values w.r.t. the case when no quantization is applied: it can be seen that the performance degradation is less than 1% (0.1%) with 2 (4) quantization bits, which is an encouraging result.



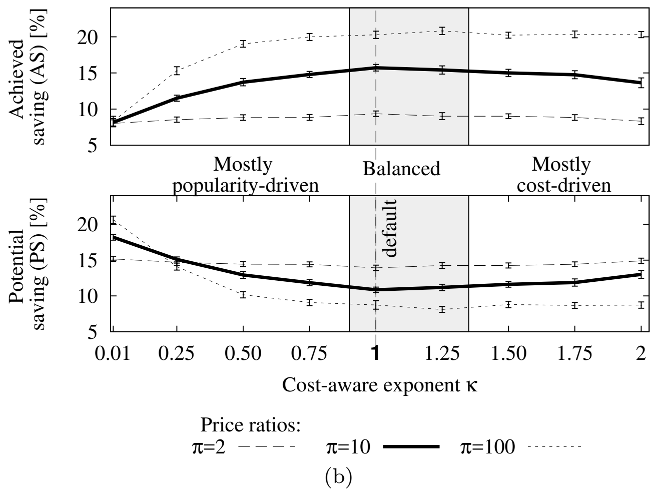


Figure 4.13: Impact of the κ exponent (that tunes the sensitivity to popularity or price), for different catalog splits (a) and different price ratios (b).

.

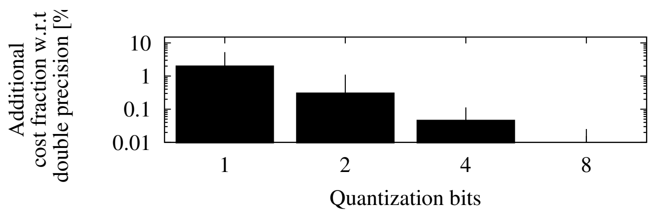


Figure 4.14: Robustness against implementation constraints: while price quantization affects accuracy of the decisions, the net effect is a negligible cost increase for the ISP.

Summary

In this part, we have tackled a fundamental question overlooked in current landscape of network caching research: namely, the reduction of operational costs as consequence of the reduced load on transit links due to caching. This sheds new light on the caching problem, which is classically devoted to the optimization of the hit ratio or other network related metrics. We show that this classic approach may not be effective in optimizing more practical metrics, the inter-domain traffic cost, in our case. Indeed, there is an inherent trade-off between hit-ratio maximization and cost minimization and thus classic caching cannot bring all the potential cost-saving.

We therefore design cost-aware mechanisms and we show sizable gains over traditional cost-blind mechanisms under a large number of settings and network topologies. Our results show that introducing a caching bias toward more expensive objects is a simple, scalable and robust solution, providing a significant cost saving at practically no additional complexity.

The caching mechanisms proposed in this part of the thesis are particularly tailored for CDNs and ICNs. As for the applicability to CDNs, we have to consider that our proposal requires object replication based on the ISP's goal of inter-domain traffic cost reduction. This is a reasonable assumption when considering CDNs operated by ISPs [59] or collaborating with them [155], so that our proposal can also fit this purpose. As concerns ICN, requirements for line of speed operation pose additional constraints, creating the need for simple yet effective solutions – challenge that we believe to have successfully tackled in this work.

While this part advocates considering economic implications of caching as first class network caching citizen, it does so by simplifying the reality: indeed, we are well aware that other aspects behind the monetary cost of inter-domain traffic could be taken into account. These aspects include for instance the latency incurred by users (QoE), the traffic flowing inside the ISP network or to the repository (QoS), which are the very same aspects we decided to ignore in the first place. Yet, while these issues are commonly studied in the literature [114, 115, 116, 59, 118, 108, 112, 109, 111, 82], the economic aspects addressed in this chapter are less common. As such, this part aims primarily at raising interest on this so far neglected aspect, and designing a viable scheme that achieve cost reduction. At the same time, our proposal is explicitly designed to be modular, so that refinement of the policies can take into account a more holistic view, combining more classical QoS/QoE aspects with the notion of cost.

Part II

Improving Video Delivery through Caching

Introduction to Video Delivery Caching

The large majority of the Internet traffic currently consists of video delivery. The related traffic is expected to explode due to increasing demand on the one hand and in reason of the increasing quality expectations of users. Indeed, at the Consumer Electronics Show in Las Vegas, “Beyond 4K Ultra HD” technologies were shown that increase pixel density by 167% [180] over the previous year – a much faster growth rate with respect to worldwide user population.

Caching video content, with either current Content Distribution Network (CDN) technologies and their interconnection[181] or more futuristic and pervasive Information Centric Network (ICN) architectures, may help containing this traffic deluge. However, the caching literature has, with few exceptions, focused on network-centric metrics like hit-ratio, hit-distance, server offload, etc., overlooking more important aspects related to the quality of user experience.

More importantly, except for some recent effort, video streaming and caching have been mostly studied as orthogonal problems, often in different research communities. Rephrasing the title of [135], caching and video are still not friends: classic video streaming mechanisms assume that a client downloads a video from a single source, which is not true in presence of caching, misleading control loops. Moreover, caching techniques are designed with generic content in mind, whereas we show in this part that video traffic has peculiarities that demand for caching mechanisms specifically tailored for it. The most important of these peculiarities is a different request-to-object mapping assumption: previous studies assume that a user request can be mapped to a single object, while a request for a video can be served by providing one of the different representations of the same video, corresponding to different quality levels, and ultimately different levels of user satisfaction.

As a first consequence, it is no longer sufficient to choose which *object* to cache, but also which of its available *representations*. Therefore, we add a new dimension to caching techniques: in addition to the classic *object placement* problem, i.e., which object to cache and where, we also consider the *representation selection* problem, i.e. which quality representation to cache. As a second consequence, the bandwidth required to satisfy a certain request is no more univocally determined by the object identifier, but depends on the quality at which we decide to serve that request. ISPs can leverage the possibility of serving the same request by using different bandwidth amounts to efficiently exploit their links and adapt to the dynamics of traffic, maximizing user satisfaction at the same time.

We consider a scenario in which Autonomous Systems (AS) peer together forming a coalition to collaboratively share their cache resources. We do not investigate the coalition formation problem, and rather focus on providing a strategy for the AS coalition to maximize the quality perceived by their users. Our key contributions can be summarized as follows:

- In Chapter 5 we propose a novel representation-aware Mixed Integer Linear Program (MILP), which determines the object placement, quality representation selection and routing, taking into account video quality in order to maximize users' experience, in a capacity and cache size-constrained network scenario.

- The knowledge gained by studying the structure of the optimal solution inspires the design of a distributed caching strategy that we present in Chapter 6. We implement it in an event-driven simulator to scale up the analysis to network sizes of hundred nodes and catalog of hundred million objects.

Our key finding is that, despite the cache deployment considerably helps in improving user quality of experience, utility maximization can be achieved by (i) minimizing the number of representations stored per object (to increase the cache efficiency), and (ii) selecting the most useful representation for each object (which is at the heart of the representation selection problem). We thus devise a simple yet effective distributed strategy that: (i) maintains a single representation per object, and (ii) incrementally improves the quality of cached objects at each new request, so that the average quality in steady state is inversely related to the object popularity.

Most of the content of this part is extracted from [15].

Chapter 5

Optimal Video Caching

In this chapter we formalize the *representation selection* problem that consists in selecting, whenever we cache an object, one of the available representations. Then, we study the optimal solution in order to understand which choices must be made in order to increase the user perception.

In more detail, we describe the system model in Sec. 5.1 and provide the representation-aware MILP model in Sec. 5.2. We point out and justify the assumptions behind our model in Sec. 5.3. In Sec. 5.4 we model some policies which are simpler to realize with respect to the optimal solution. We model them by simply adding appropriate constraints to the MILP. Finally, in Sec. 5.5 we numerically evaluate the optimal solution, we compare its performance with the policies described before and we study the structural properties of the optimal solution.

5.1 System model

We illustrate the system model considered in our work, with the help of an example scenario depicted in Fig. 5.1. We consider a set $\mathcal{V} = \{1, 2, \ldots, \mathcal{V}\}$ of Autonomous Systems (ASes), whose interconnection is represented by a graph, composed of nodes and capacitated arcs. Nodes in the graph (ASes) can act as *content producers* (when they are directly connected to some repositories), *transit* ASes that merely participate in the content caching and diffusion, or *consumer* ASes that additionally generate video requests. Repositories and caches distributed in the ASes store objects (in particular, multimedia content and videos), of which different representations (quality levels) exist, belonging to a discrete set $\mathcal{Q}$. Each of these quality levels is associated to a rate $\varsigma^{[q]}$ necessary to support and transmit the object at the given quality q, as well as to a storage space $s^{[q]}$ that is necessary to cache it. AS users issue requests for videos without specifying the quality representation, given that the model will find the optimal one.

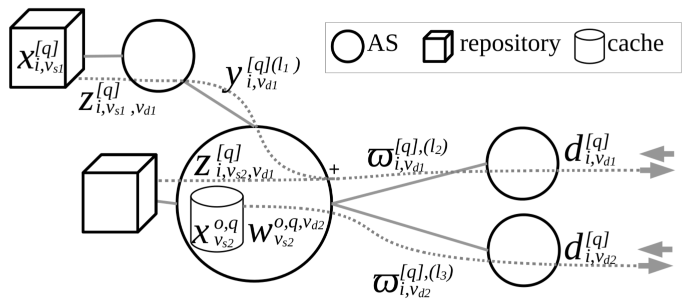

AS
repository
cache
$x_{i,\nu_{s1}}^{[q]}$
$z_{i,\nu_{s1},\nu_{d1}}^{[q]}$
$y_{i,\nu_{d1}}^{[q](l_1)}$
$z_{i,\nu_{s2},\nu_{d1}}^{[q]}$
$x_{\nu_{s2}}^{o,q}$
$w_{\nu_{s2}}^{o,q,\nu_{d2}}$
$\varpi_{i,\nu_{d1}}^{[q],(l_2)}$
$\varpi_{i,\nu_{d2}}^{[q],(l_3)}$
$d_{i,\nu_{d1}}^{[q]}$
$d_{i,\nu_{d2}}^{[q]}$
+

Figure 5.1: Example scenario indicating the main variables employed in the MILP model.

Each AS has *upstream links* through which data is retrieved from other nodes and *downstream links* through which data is sent to users. ASes are endowed with caching capabilities, and can store objects as well as route object requests/data towards neighbor routers, the repository or clients. To be as general as possible, we do not specify the details of the technology that provides caching capabilities (ICN, CDN, Web proxy, etc.). Each object can be served at different qualities, which may depend on the network characteristics (link capacities, bottlenecks) and the clients position, and produce a utility that is experienced by users. The aim of our work is to determine (i) optimal allocations of objects to AS caches, (ii) optimal quality level(s) to store for each cached object and to map to each request, as well as (iii) optimal routing strategies, which collectively contribute in maximizing the overall utility perceived by network users.

5.2 Representation-Aware MILP

We provide now the Representation-Aware model. We summarize the variables we use in Tab. 5.1. The model is formalized as follows:

$$\max \sum_{i\in\mathcal{O}} \sum_{q\in\mathcal{Q}} \sum_{\nu\in\mathcal{V}} n_{i,\nu}^{[q]} U^{[q]} \tag{5.1}$$

subject to:

$$\sum_{q\in\mathcal{Q}} n_{i,\nu}^{[q]} = n_{i,\nu}, \qquad \forall i \in \mathcal{O}, \nu \in \mathcal{V} \tag{5.2}$$

$$d_{i,\nu_d}^{[q]} = n_{i,\nu_d}^{[q]} \cdot \varsigma^{[q]}, \qquad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu_d \in \mathcal{V} \tag{5.3}$$

$$d_{i,\nu_d}^{[q]} = z_{i,\nu_d,\nu_d}^{[q]} + w_{i,\nu_d,\nu_d}^{[q]} + \sum_{e\in BS(\nu_d)} \varpi_{i,\nu}^{[q],(l)} - \sum_{e\in FS(\nu_d)} \varpi_{i,\nu}^{[q],(l)}, \quad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu_d \in \mathcal{V} \tag{5.4}$$

Parameters of the Model	
$\mathcal{A}$	Set of arcs
$\mathcal{V}$	Set of Nodes (Autonomous Systems, ASs)
$\mathcal{O}$	Set of objects
$\mathcal{Q}$	Set of qualities
$FS(\nu)$	Set of forward arcs $(\nu, \nu') \in \mathcal{A}$ for node $\nu \in \mathcal{V}$
$BS(\nu)$	Set of backward arcs $(\nu', \nu) \in \mathcal{A}$ for node $\nu \in \mathcal{V}$
$b^{(l)}$	Capacity of the arc $l \in \mathcal{A}$
$n_{i,\nu}$	Number of requests for object i, in AS $\nu \in \mathcal{V}$
$\varsigma^{[q]}$	Rate required to retrieve an object at quality $q \in \mathcal{Q}$
$s^{[q]}$	Storage space required to cache an object at quality $q \in \mathcal{Q}$
$U^{[q]}$	Utility gained to serve one request for an object at quality q
$\Xi_{i,\nu}$	0-1 Producers reachability matrix $\Xi_{i,\nu} = 1$ if AS ν has a producer for object $i \in \mathcal{O}$ (it can serve whatever quality of object o)
c_ν	Max caching storage that can be installed at AS ν
K	Max caching storage that can be installed in the network
be_ν	Max egress capacity for AS $\nu \in \mathcal{V}$, $be_\nu = \max\left(\sum_{l \in FS(\nu)} b^{(l)}; \sum_{i \in \mathcal{O}} n_{i,\nu} \cdot \max_{q \in \mathcal{Q}} \varsigma^{[q]}\right)$

Decision Variables of the Model	
$n_{i,\nu}^{[q]}$	Number of requests for object i at quality q satisfied at AS ν
$x_{i,\nu_s}^{[q]}$	0-1 Caching variable, if the source AS $\nu_s \in \mathcal{V}$ caches i at quality q
$\varpi_{i,\nu_d}^{[q],(l)}$	Flow on arc $e \in \mathcal{A}$ for object $i \in \mathcal{O}$, at quality q sent to the destination AS $\nu_d \in \mathcal{V}$
$d_{i,\nu_d}^{[q]}$	Rate requested at AS $\nu_d \in \mathcal{V}$, for object i at quality q
$z_{i,\nu_s,\nu_d}^{[q]}$	Rate provided by the source AS $\nu_s \in \mathcal{V}$, for object i, at quality q for the destination $\nu_d \in \mathcal{V}$, when ν_s behaves as a producer ($\Xi_{i,\nu_s} = 1$)
$w_{i,\nu_s,\nu_d}^{[q]}$	Rate provided by the source AS $\nu_s \in \mathcal{V}$, for object i, at quality q for the destination $\nu_d \in \mathcal{V}$, when ν_s behaves as a cache ($x_{i,\nu_s}^{[q]} = 1$)

Table 5.1: Summary of the notation used in this chapter.

$$z_{i,\nu_s,\nu_d}^{[q]} + w_{i,\nu_s,\nu_d}^{[q]} + \sum_{e \in BS(\nu_s)} \varpi_{i,\nu_d}^{[q],(l)} = \sum_{e \in FS(\nu_s)} \varpi_{i,\nu_d}^{[q],(l)}, \quad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu_s \in \mathcal{V}, \nu_d \in \mathcal{V}, \nu_s \neq \nu_d \tag{5.5}$$

$$\sum_{i \in \mathcal{O}} \sum_{q \in \mathcal{Q}} \sum_{\nu_d \in \mathcal{V}} \varpi_{i,\nu}^{[q],(l)} \leq b^{(l)}, \qquad \forall e \in \mathcal{A} \tag{5.6}$$

$$\sum_{q \in \mathcal{Q}} \sum_{\nu_d \in \mathcal{V}} z_{i,\nu_s,\nu_d}^{[q]} \leq \Xi_{i,\nu_s} \cdot be_{\nu_s}, \qquad \forall i \in \mathcal{O}, \nu_s \in \mathcal{V} \tag{5.7}$$

$$\sum_{\nu_d \in \mathcal{V}} w_{i,\nu_s,\nu_d}^{[q]} \leq x_{i,\nu_s}^{[q]} \cdot be_{\nu_s}, \qquad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu_s \in \mathcal{V} \tag{5.8}$$

$$\sum_{i \in \mathcal{O}} \sum_{q \in \mathcal{Q}} x_{i,\nu_s}^{[q]} \cdot s^{[q]} \leq c_{\nu_s}, \qquad \forall \nu_s \in \mathcal{V} \tag{5.9}$$

$$\sum_{i \in \mathcal{O}} \sum_{q \in \mathcal{Q}} \sum_{\nu_s \in \mathcal{V}} x_{i,\nu_s}^{[q]} \cdot s^{[q]} \leq K \tag{5.10}$$

$$x_{i,\nu}^{[q]} \in \{0,1\}, \qquad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu \in \mathcal{V} \tag{5.11}$$

$$n_{i,\nu}^{[q]} \in \mathbb{Z}^+, \qquad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu \in \mathcal{V} \tag{5.12}$$

$$\varpi_{i,\nu}^{[q],(l)} \in \mathbb{R}^+, \qquad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu_d \in \mathcal{V}, e \in \mathcal{A} \tag{5.13}$$

$$d_{i,\nu_d}^{[q]} \in \mathbb{R}^+, \qquad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu_d \in \mathcal{V} \tag{5.14}$$

$$z_{i,\nu_s,\nu_d}^{[q]}, w_{i,\nu_s,\nu_d}^{[q]} \in \mathbb{R}^+, \qquad \forall i \in \mathcal{O}, q \in \mathcal{Q}, \nu_d \in \mathcal{V}, \nu_s \in \mathcal{V}. \tag{5.15}$$

In particular, objective function (5.1) represents the overall utility experienced by network users, which is maximized by our model. The set of constraints (5.2) makes sure that all the requests are served at one (or more) quality level(s). In the problem instances we add a "special" quality level $q = 0$, which represents unserved traffic demands: when serving quality $q = 0$, no bandwidth is required ($\varsigma^{[q]} = 0$); moreover, no utility is generated, $U^{[0]} = 0$. Constraints (5.3) set the value of the rate requested at AS ν_d, for object i, at quality q. Such demand is satisfied in (5.4). In particular, it can be satisfied because: (i) the AS is a producer for that object (i.e.: $z_{i,\nu_d,\nu_d}^{[q]} = d_{i,\nu_d}^{[q]}$), (ii) the AS caches the object (i.e.: $w_{i,\nu_d,\nu_d}^{[q]} = d_{i,\nu_d}^{[q]}$), or (iii) the AS retrieves the object (i.e.: the sum of flows on incoming links).

Flow balance constraints are imposed in (5.5) and we bound the arc capacity in (5.6). Similarly, in (5.7) and (5.8), we limit the maximum emitted flows the AS sends when it behaves as a producer and a cache, respectively. The overall caching storage that can be deployed by an AS is bounded in (5.9), and we extend the same limit to the entire topology in (5.10). Finally, integrality and non-negativity constraints are imposed in (5.11)-(5.15).

5.3 Discussion

The optimal solution of the MILP can be seen as an offline policy: we take an optimal decision based on some a-priori knowledge (the input of the MILP) and we push it on all the nodes of the network. As such, our offline policy has the pros and cons already discussed in Sec. 2.3.1. Nonetheless, in this chapter our goal is not to provide a ready-to-use policy and evaluate it in realistic scenarios. Our aim here is to give an understanding of the representation selection problem. What we learn from this chapter will be the base of the next chapter, in which we devise and online policy and evaluate it in realistic scenarios.

We now discuss the goal of our MILP. While related work usually aims to minimize delay in order to improve user perception, we focus instead on maximizing the provided quality for two reasons: i) we want our contribution to be complementary to this related work, ii) the packet delay can be absorbed by playout buffers and be invisible to the user. The only exception to this is when this delay is excessively high or variable, causing high startup times or rebuffering episodes. This happens in case of congestion. For these reasons, rather than looking at the delay, we focus on caching content at the right quality, such that it can be transmitted using the available bandwidth on the path, thus avoiding congestion.

Another aspect worth underlining is that in today's video delivery plugins in the user Web browser select the quality representation to request, while we assume that ISPs choose the best possible quality to serve its users. We discuss this assumption in Sec. 2.6.5.

Additionally, we remark that, while a vast literature on video streaming has focused on congestion control algorithms to make the client choose the right representation to download, our study focuses on caching and aims at finding the performance bounds from a more abstract viewpoint. The findings we provide here should be considered what an optimal caching strategy can theoretically achieve, supposing a perfect congestion control mechanism at the bottom. For this reason, we can adopt the snapshot approach, as in other notable works on video delivery [131].

It is worth noticing that our model can be easily extended to have a fine-grained representation, considering heterogeneity of video type and user device. As for the former, it is known that videos with different subjects (sport, movies, TV shows), even if encoded at the same bit-rate and resolution, are perceived in a different way[131]. As for the latter, a user watching a video on a smartphone may be perfectly satisfied with a resolution and a bit-rate lower than the one demanded by a user using a ultra-HDTV 4K screen. However, this level of detail is beyond our scope and such directions can be incorporated at a later step in the model.

5.4 Model of offline AS policies

Jointly deciding the optimal representations that each node should cache and serve to users is a hard task to be performed by a distributed online strategy, in which each node makes local decisions without having knowledge of the status of the rest of the network and the overall set of requests. Nonetheless, our aim, which we will finalize in the next chapter, is to give a feasible solution that can

Table 5.2: Model variants implementing natural AS policies: additional constraints for the MILP model (5.1)-(5.15)

Caching Policy	Additional constraint in MILP model
NoCache	$x_{i,\nu}^{[q]} = 0$
CacheLQ	$x_{i,\nu}^{[q]} = 0, \forall q \neq LQ$
CacheHQ	$x_{i,\nu}^{[q]} = 0, \forall q \neq HQ$
AllQ	$x_{i,\nu}^{[q_h]} = x_{i,\nu}^{[q_k]}, \forall q_h, q_k \in \mathcal{Q}$
Partitioned	$\sum\limits_{i \in \mathcal{O}} x_{i,\nu}^{[q]} \cdot s^{[q]} \leq c_\nu \cdot \frac{s^{[q]}}{\sum_{q' \in \mathcal{Q}} s^{[q']}}$

be deployed in a real network, providing a good performance at least close to the optimal one.

We thus constrain our model to give solutions with a simpler structure and we verify how far they are from the optimum. The constrained variants of the model, detailed hereafter, are easier to approximate in distributed, online algorithms and, as our numerical results will show, some of them exhibit indeed very good performance, close to optimality in several situations. Thanks to the flexibility of our MILP model, modeling AS policies is as simple as adding a single constraint for each strategy.

Such constraints, specified in Tab. II, include: a *No Cache* strategy, which never caches videos; *CacheLQ* and *CacheHQ*, which exclusively cache the lowest (highest) quality representation available, indicated with quality level LQ (HQ), respectively; *AllQ*, which caches all quality representations for any cached object; finally, *Partitioned* stores the same *number* of objects for each quality representation (while their buffer occupancy depends on the quality of the corresponding representation). Note that the constraints only concern caching, and do not force to serve a request with a specified quality representation. For example, a HQ video can still be served, even when *CacheLQ* is employed; in such case, given that HQ videos cannot be cached, they must be retrieved directly by a repository and cross all links between this latter and the user. In this work, we assume that all ASes in the coalition use the same policy, chosen among the ones described above.

5.5 Numerical Results

This section evaluates the impact of caching on the overall video quality perceived by users, showing the validity of the caching strategies proposed so far. To this aim, we provide numerical solutions of the MILP model via the CPLEX 12.5 solver. After describing the scenario in Sec. 5.5.1, we investigate performance and properties of the proposed strategies in an incremental fashion.

Focusing on a single AS, we first illustrate the advantages of optimal video caching and the structure of the optimal solution in Sec. 5.5.1. In Sec. 5.5.3 we analyze which fraction of the provided videos are served by the cache and which fractions flows through the upstream link and we study the breakdown of the served quality in the two cases. In Sec. 5.5.4 we evaluate the performance of the policies introduced in Sec. 5.4 and their structural differences. Moving to a multi-AS scenario, we finally confirm MILP results to hold on a 10-node topology in Sec. 5.5.5

5.5.1 Scenario

We consider five quality levels [182] in the set $\mathcal{Q}$ as reported in Tab. 5.3. Each quality corresponds to a given resolution and bitrate, which both increase for increasing quality levels. We only report the bitrate as this is more pertinent to our optimization goal: video bitrate correlates to both cache storage space, as well as network bandwidth. Resolution, instead, does not come into play directly in the system model, apart from determining a different user perception, which is accounted for in the utility function.

The utility function must be an increasing function of the provided quality, since the higher the quality provided to a user is, the better the utility. Moreover, it must be concave to express the diminishing return in the experience of human vision when providing improved quality [183]. The exact shape of such an utility function is still subject to debate, and there is no unanimously accepted function. Indeed, gathering this function is a hard task that requires intensive experimentation with real users, which is far from the scope of this thesis. To gather results that are not tied to a specific function, we consider two shapes at the broad end of the spectrum of plausible utility functions, tabulated in Tab. 5.3. Specifically, we define $u_1(q)$ as a model with linear return with respect to the quality: the model likely underestimates contributions of low quality videos, and does not exhibit diminishing returns [183], so that it is biased toward high-quality content. We next define $u_2(q)$ as a power function with a higher concavity: this model does exhibit diminishing returns and sits at the other side of the spectrum as it possibly overestimates contributions of low-quality videos (notice indeed that $u_2(q=1) > 3u_1(q=1)$). In the following, we will report the *average system utility* as the average per-request utility, i.e. the total utility as in (5.1) divided by the total number of requests.

Unless otherwise stated, we consider the single AS scenario depicted in Fig. 5.4: at a logical level, in the cache-stream we include the flows of videos downloaded from the cache, while up-stream includes the flows retrieved through

Table 5.3: Quality levels and corresponding transmission rates, cache occupancy and perceived utility (linear/concave).

Quality	Rate (Kbps)	Utility $u_1(q)$	Utility $u_2(q)$
1	300	0.2	0.67
2	700	0.4	0.80
3	1500	0.6	0.88
4	2500	0.8	0.95
5	3500	1	1

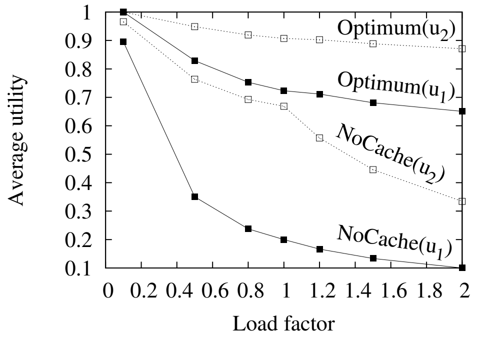


Figure 5.2: Benefits of optimal caching and impact of the utility functions.

other ASes. The cache represents the aggregate of several cache nodes within the AS, and similarly the up-stream resource represents a single logical link, aggregating all physical links where the request can be satisfied (i.e., all the links except the one where the request is coming from).

Also, unless otherwise stated, the catalog comprises $|\mathcal{O}| = 10^4$ objects whose popularity conforms to Sec. 3.4.1. The exponent of the Zipf distribution we consider here is $\alpha = 1$. The cache space at each AS is sufficient to store 1/100 of the catalog objects at the highest quality, *HQ*. Observe that the size of each object depends on its quality, i.e. an object at quality q is $s^{HQ}/s^{[q]}$ times smaller than an object at the highest quality, with s^{HQ}/s^{LQ} exceeding one order of magnitude as can be seen in Tab. 5.3.

All links have the same capacity b. We express the number of user requests as a *load factor* Υ, i.e. the factor by which we should multiply b in order to transmit all the requested objects at the lowest quality, *LQ*. Otherwise stated, if the load factor is $\Upsilon = 1$, even if no cache is deployed in the network, we can satisfy all requests at quality LQ, by fully utilizing the network capacity. Notice, however, that due to cache space, it makes sense to consider a normalized load larger than $\Upsilon > 1$, since part of the endogenous requests can be served from the cache without consuming upstream bandwidth.

5.5.2 Utility Gains and Structural Properties of the Optimum

We first start assessing the dependency of our results on the particular perceptual model in the single cache scenario. We contrast two extremes, namely the optimal solution against the case in which the system is not equipped with caches (so that this latter can sustain a load at most equal to $\Upsilon = 1$). The average utility is shown in Fig. 5.2 for both linear $u_1(q)$ and concave models $u_2(q)$: while quantitative results are of course affected by the peculiar function, qualitative results are instead independent of the utility function considered. In particular, the improvement of user experience provided by optimal caching

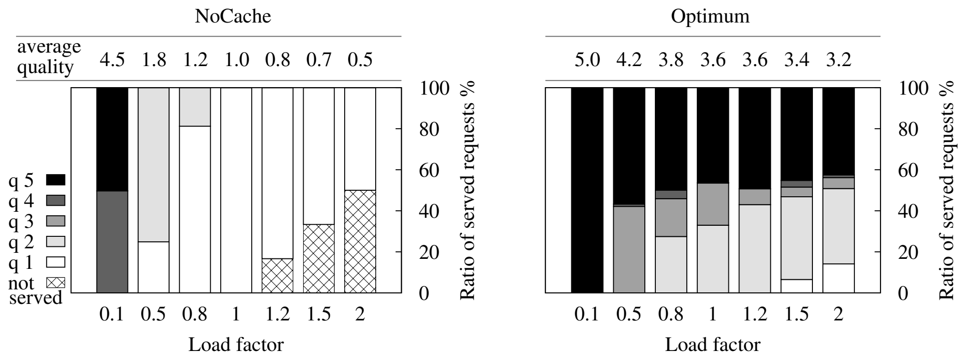


Figure 5.3: Percentage of requests served at different quality levels, with and without cache, for a varying load, in the single cache scenario.

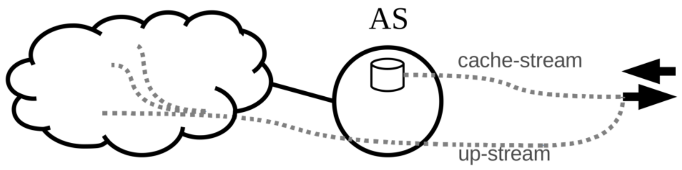


Figure 5.4: Single-AS scenario: Videos can be downloaded from the cache (cache-stream), while up-stream includes the flows retrieved through other ASes.

is notable at high load, where caches at the AS absorb a large fraction of the requests, alleviating the impact of the upstream bandwidth limitation. Since the qualitative results between $u_1(q)$ and $u_2(q)$ remain unchanged, and to avoid cluttering the pictures, we only consider the concave profile of $u_2(q)$ in what follows.

A breakdown of the quality levels served to users is reported in Fig. 5.3, which helps to better understand the structure of the optimal solution – thus, ultimately, where the utility gain comes from. Without any cache, all the delivered videos must cross the upstream link, and the bandwidth is hardly available to transmit them at high quality, unless the input load is particularly low ($\Upsilon = 1/10$). At high load, the bandwidth is not sufficient to serve all the requests, not even at the lowest quality, and a growing fraction remains unsatisfied. The situation is drastically improved by optimal caching, which stores a significant fraction of videos, and *especially the most popular ones, at high quality.* Since the requests for these videos account for a large part of the overall requests, the upstream link is relieved of a considerable amount of traffic. As a first consequence, we are able to satisfy all the requests coming from users.

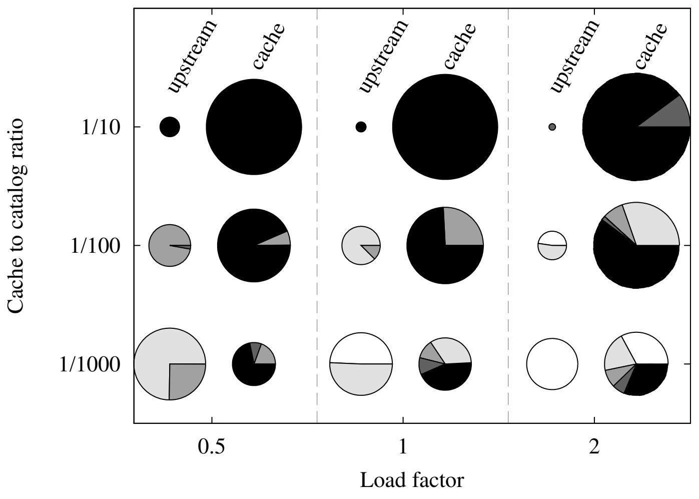


Figure 5.5: Single-AS scenario: Contributions of cache and upstream links. Circle size reflects relative cache vs upstream contribution, and the breakdown reports the qualities of both contributions.

Moreover, the most popular objects are served at high quality, which as net effect increases the average utility perceived by users.

5.5.3 Contributions of cache and upstream link

To better understand the relative contribution of storage vs bandwidth, we decompose the video flows arriving at users in *cache-stream* and *up-stream*, where the former is the stream of data retrieved from the cache, whereas the latter is the flow coming from the upstream links, as illustrated in Fig. 5.4. We represent the breakdown of the utility provided by content retrieved from cache vs content retrieved from upstream in Fig. 5.5, where the sizes of the circles represent the relative contribution of the two utility values. Circles additionally report the quality breakdown of the two contributions.

From Fig. 5.5 we first observe that, in the scenarios under consideration, the cache is responsible of the most part of the utility (storage circles are bigger than upstream ones), as it stores the most popular objects (thus intercepting a large fraction of traffic) at a furthermore high quality.

Second, an interesting specialization arises between the cache-stream and up-stream: the highest quality levels (darker colors) are served by the cache and only low representations cross the upstream link. Indeed, it would not be beneficial to serve high quality objects through the upstream link, since the high bandwidth cost should be paid repeatedly, at each request. On the contrary, placing them in the cache permits to pay only once the cost in terms of memory, and to still repeatedly gather utility at each request.

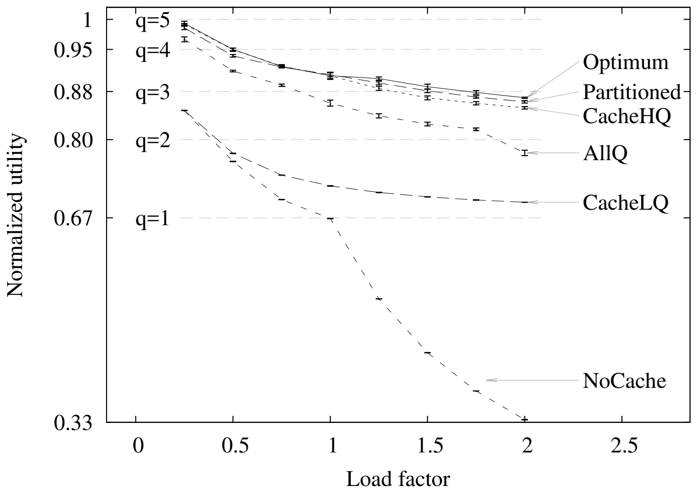


Figure 5.6: Single-AS scenario: comparison of MILP variants.

Third, the load has an evident impact on the breakdown. At high load, both streams must carry lower quality representations. Indeed, in this case, the average quality of the upstream must be low, to fit the link capacity. At the same time, we need to reduce the number of transmissions on the upstream by intercepting more requests with cached copies. To do so, we need to cache a larger number of different videos and, since the cache space is limited, we need to store smaller copies of them, i.e. lower quality representations. This explains why, at high load, the quality of the cache-stream decreases.

Fourth, we observe the impact of cache size on the breakdown: as expected, when the cache size increases, its relative contribution to the overall utility increases as well. Yet, more interestingly, also the breakdown of the stored video quality changes as well: in particular, the larger the cache, the higher the quality, which is intuitive.

Finally, observe that the cache size has a side effect on the breakdown of the upstream video quality: indeed, the average quality increases for increasing cache size, which can be explained with the fact that the larger the cache, the larger the fraction of absorbed traffic. As a consequence, at any given load the upstream link has to serve less requests and can afford to do it at higher quality.

5.5.4 Performance of Offline Policies

We next compare the performance of the five strategies discussed in Sec. 5.4 (viz., *NoCache, CacheLQ, CacheHQ, AllQ, Partitioned*), with the solution that maximizes the quality of experience perceived by network users (*Optimum*). Utility is reported in Fig. 5.6, whereas the structure of the solution is reported

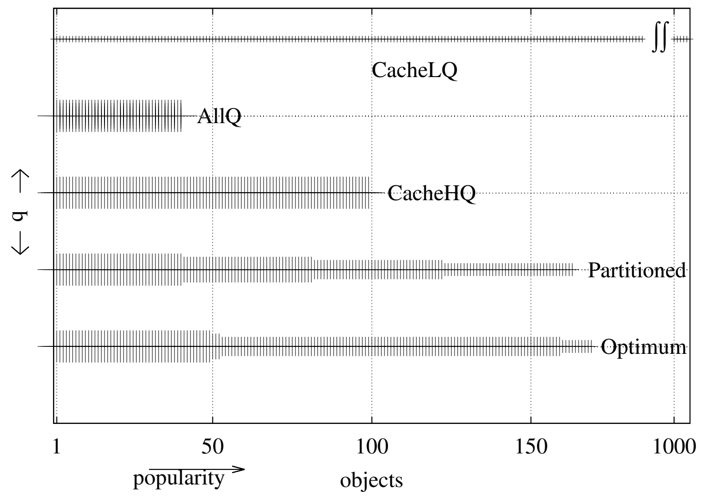


Figure 5.7: Quality levels cached by offline strategies (MILP variants) in the single-AS scenario.

in Fig. 5.7, which depicts the quality level of each stored object under the offline strategies.

Note that, when the network caches only low quality objects (*CacheLQ*), their small size permits to store a large number of them, intercepting a large fraction of the requests. This already provides robustness with respect to load, guaranteeing at least a minimum quality (the *CacheLQ* curve is above the $q = 1$ reference quality), which is not possible without cache. However, *CacheLQ* does not exhibit cache efficiency because higher quality objects, necessary to increase the average utility, can only be retrieved through the upstream link. Rigidly storing all quality representations (*AllQ*) further improves the performance but is still far from the optimum. Indeed, for each object we must waste cache space for all the representations, although only a subset of them will be actually served to users. This limits the number of different objects that can be actually cached. *CacheHQ* performance approaches to the *Optimum*, suggesting that storing few (due to their large size) popular objects at HQ already provides a notable payoff (due to the product of their popularity times their utility at high quality).

Yet, *Partitioned* performance is even closer to the *Optimum*: the root cause is that the quality representation selection is similar to the optimal one, as Fig. 5.7 shows. In particular, the optimal behavior in terms of overall utility is to store a number of objects at each quality, preferring to store more popular objects at higher quality, and *Partitioned* implements this behavior. This increases the overall cardinality of cached content and assigns to each object the "right" quality, i.e. the one such that the cost in terms of occupied memory is compensated by the pay-off in terms of utility provided to the set of requests for

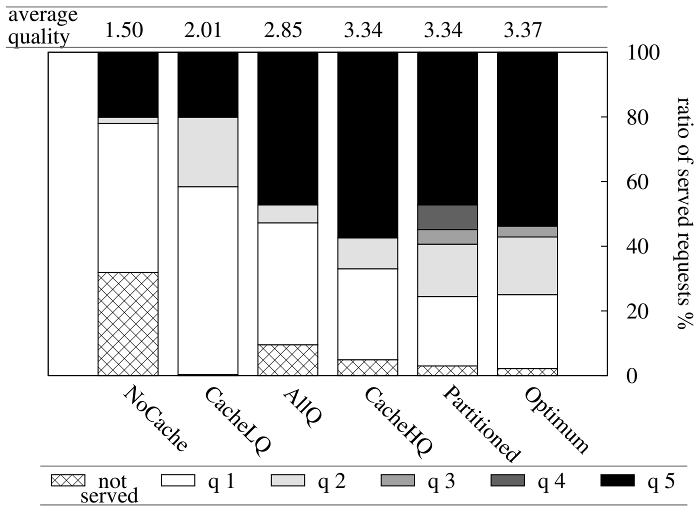


Figure 5.8: MILP variants (10-nodes network, 10^3 catalog, load $\Upsilon = 2$)

it. The difference between *Partitioned* and *Optimum* is in the number of objects stored at each quality. While *Partitioned* constrains this number to be the same for each quality, *Optimum* does not incur this constraint and prefers, in this scenario, roughly two quality levels. So doing, the *Optimum* strategy caches more objects than *CacheHQ* (but less than *Partitioned*), a significant fraction of which is at lower quality than *CacheHQ* (but higher than *Partitioned*).

From the above observations, we infer that the quality at which each object must be cached should increase with its popularity. This is the observation we will leverage in the design of our online policy, in the next chapter. Note that a fairness concern may arise, since popular content is served better than the rest. In any case, bandwidth is limited and it is impossible to serve all the content at high quality. Therefore, a network provider has two choices: i) being fair and lowering the quality of all the served videos or ii) differentiating based on popularity. While the former case is admissible, we have shown that the latter permits utility maximization, which is our target. On the other hand, a network provider may wish to provide always a quality above a certain threshold higher than LQ. We can easily model this by removing from the set $\mathcal{Q}$ of the admissible levels the lowest ones.

5.5.5 Multi-AS Scenario

We now consider a multi-AS environment, where each AS operates a cache system with a storage space sufficient to cache 1/100 of the catalog at the highest quality. We solve the MILP model for a coalition of 10 ASes and a 10^3 objects catalog and we show results in Fig. 5.8. The multi-AS graphs are generated in accordance to the Barabasi-Albert model [184], which is considered to approximate the AS interconnection in Internet [112].

Observe that results are coherent with the observations on the single-AS case. Specifically, we notice that while the average quality is very similar among *CacheHQ, Partitioned* and *Optimum*, however the fraction of content that is not served is largely different. In the case of *CacheHQ*, about 5% of the videos are not served, which is 2.3 times larger than the fraction of non-served videos in the *Optimum* case. In contrast, the *Partitioned* strategy limits to +30% the amount of additional videos not served with respect to *Optimum.*

While this fact does not appear in the perceptual model we used (where a non served content has a utility 0 and does not generate any penalty) nevertheless it can be argued that the impact of service denial can be much worse. Indeed, from loss aversion models commonly used in prospect theory [185], not receiving a video at expected quality q generates a negative utility, which could be accounted for in the model. Yet, repeatedly receiving denial of requests could lead users to change ISPs on a long timescale, which can have disastrous consequences on the ISP business, for which limiting the fraction of non-served content is primordial.

A second important observation is that gains are structurally equivalent to what early shown in the single-AS case.

Chapter 6

An Online Distributed Caching Strategy for Video Delivery

The strategies proposed in the previous chapter are offline policies and, as already explained in Sec. 2.3.1, are not realistically implementable. Nonetheless, by studying the results obtained via the MILP, we learned two important guidelines. First, to manage more efficiently the resources deployed in the network, i.e. link capacities and caches, we do not have to cache all the available representations of the objects, but just the "right" ones. Second, the right representation should obey to the following: the most popular objects should be cached at high quality and then the quality should decrease with the popularity. In this chapter we devise an online algorithm, which we call *QImpr*, that complies at regime with these guidelines.

In the previous chapter the only objective was to maximize user experience. While this allowed us to present the representation selection problem and gaining insights on caching strategies specifically tailored for video delivery, we are aware that user maximization is not the sole goal of ISPs. If it were the case, they would risk to incur too large cost, due to excessive bandwidth usage. In reality, an ISP seeks to guarantee good quality of experience to users while limiting its operational cost. To investigate this aspect, we also evaluate in this chapter the trade-off between user experience maximization and the bandwidth usage.

The chapter is organized as follows: we describe the algorithm in Sec. 6.1 and the simulation setup in Sec. 6.2. Then we show that the behavior of our online algorithm approaches the the optimum in Sec. 6.3. Finally, in Sec. 6.3 we present simulation results in large topologies and discuss the user experience vs. bandwidth usage trade-off, showing that QImpr efficaciously balances the two.

6.1 QImpr Algorithm

We propose a *Quality Improvement* (*QImpr*) strategy that reactively incrementally improves quality of stored replicas at each new request, and opportunely balances bandwidth and user utility. *QImpr* operates as follows. Each request $req(q)$ carries a value q specifying the minimum required quality, which is always set to $q = 1$ at the ingress of the network (either by the user browser or the ISP proxy) meaning that any quality is accepted (i.e., we assume that receiving a LQ representation is preferable to not receiving the video at all). When a new request $req(q)$ arrives at any cache, if a copy at quality $q_{\text{cached}} \geq q$ is found, the request is served with that copy. At the same time, if q_{cached} is less than the maximum quality, the AS node issues another request $req(q_{\text{cached}} + 1)$ for the same object. If no cached copy is found, $req(q)$ is normally forwarded.

Caches maintain objects in an ordered list. Whenever an object i at quality q arrives, if a better quality $q_{\text{cached}} \geq q$ of that object is already cached, the incoming object is discarded. Otherwise, the new object representation (i) is placed at the head of the list, (ii) any lower quality representation of i is evicted, (iii) if further space is needed to store i, this is obtained by evicting cached objects starting from the least recently used one, up toward the head, until a sufficient space to accommodate i at quality q is available. Shortly, expected benefits of this policy are that unpopular objects will only be cached at low quality, whereas popular objects will quickly escalate quality levels. On the downsides, popular objects will be requested at multiple quality levels, generating a slight overhead in the quality improvement process.

Note that in reason of size heterogeneity between representations at different levels, caching a new object causes the eviction of a variable number of least-recently-used objects sufficient to make room for the incoming higher quality representation. This is in contrast with what usually assumed in the caching literature that assumes all chunks having equal size.

6.2 Simulation Setup

We implement our algorithm in ccnSim, an Omnetpp-based simulator of network of caches, available and opensource in [11]. We consider the scenario already described in Sec. 5.5.1 and Sec. 5.5.5, with the difference that we first consider one only node in Sec. 6.3 and for the rest of the chapter we will consider much larger networks, with 100 nodes (an example of which is depicted in Fig. 6.3), and a much larger catalog, with 10^8 objects.

Results are collected in steady state over 10 runs for each scenario. The bandwidth utilization is computed considering that, every time an object at quality q crosses a link, it occupies a bandwidth $\varsigma^{[q]}$. The bandwidth that we represent in the plots is averaged over time and over all the links of the network.

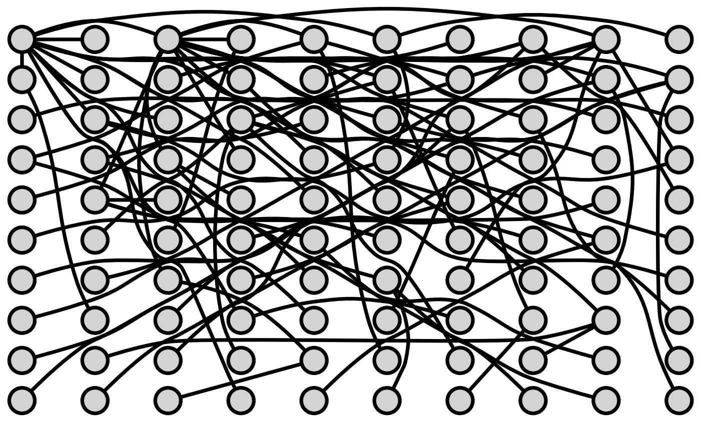


Figure 6.1: Example large-scale topology

6.3 Comparison with the Optimal Solution

We analyze what the breakdown of the quality of the content cached by QImpr is at regime and compare it with the optimal breakdown in Fig. 6.2.

Notice that while solving the optimization problem of the previous chapter returns exactly one object quality, in the simulation case the representation of an object stored in the cache varies over time, so that we report the average quality for an object sampled at 100 random times during the simulation. It can be seen that *QImpr* tends to store only the popular objects at high quality, thus approaching a solution that is structurally similar to *Partitioned* or *Optimal* strategies – which confirms the mechanism of improving the object quality at each new hit to pay off.

6.4 Distributed policy: Bandwidth-Utility Trade-Off

In the MILP of the previous chapter our only objective is to maximize the utility. To do so, we let the model use the links at their full capacity. In practice, ASes may tend to limit link utilization, in order to avoid the occurrence of congestion, to ensure low end-to-end latency and bound operational costs associated to traffic transmitted toward transit providers. Therefore, a trade-off arises between the utility provided to users and the bandwidth used to guarantee it, which we investigate via simulation.

Aiming at assessing the utility vs. bandwidth trade-off, we use two additional reference points where only low-quality ($OnlyLQ$), or high-quality objects are cached and transmitted in the system ($OnlyHQ$), and caches employ standard Least Recently Used (LRU) replacement. $OnlyLQ$ corresponds to a crude attempt to minimize the bandwidth, but, as a consequence, brings low user

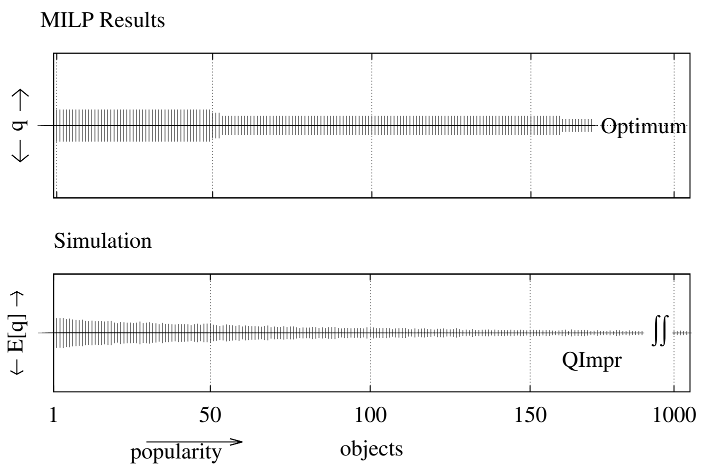


Figure 6.2: Breakdown of the quality cached at optimum and by the online policy QImpr.

utility. On the other hand, *OnlyHQ* maximizes the user utility but incurs in high bandwidth usage.

Note that the points in Fig. 6.3 are well clustered, meaning that the performance of *OnlyLQ*, *OnlyHQ* and *QImpr* is coherent and our findings do not vary with the scale of the problem. *QImpr* nicely fits halfway the extreme bounds represented by *OnlyLQ* and *OnlyHQ*, realizing a smooth tradeoff between bandwidth and quality.

The picture finally shades an area where the performance of interesting distributed algorithms lays: i.e., those that achieve a more convenient bandwidth-quality tradeoff. *QImpr* design can be ameliorated to move performance in the upper-left part of Fig. 6.3 by (i) reducing the overhead (i.e., move left) and (ii) improving the utility (i.e., move up). As far as (i) *overhead* is concerned, recall that whenever a request hits a cached copy at quality q, the cache immediately triggers a request to improve the content quality to $q + 1$. These cache-originated requests constitute an overhead, which could be limited by probabilistically reducing the rate at which they are issued – much as in probabilistic metacaching. As far as (ii) *utility* is concerned, recall that the *Optimal* solution implicitly quantized the quality levels to a subset of all the available ones, which should be easy to implement.

Notice however that overhead reduction and utility maximization are conflicting goals, since e.g., slowing down the rate at which quality of content is improved from q to $q + 1$ by a given factor also implies that the amount of requests served at quality q instead of $q + 1$ grows by the same factor. While this observation affects only the transient but vanishes in the steady state, it can however be argued that it has practical relevance in real scenarios where popularity is time-varying and there is no steady state. Additionally, the choice

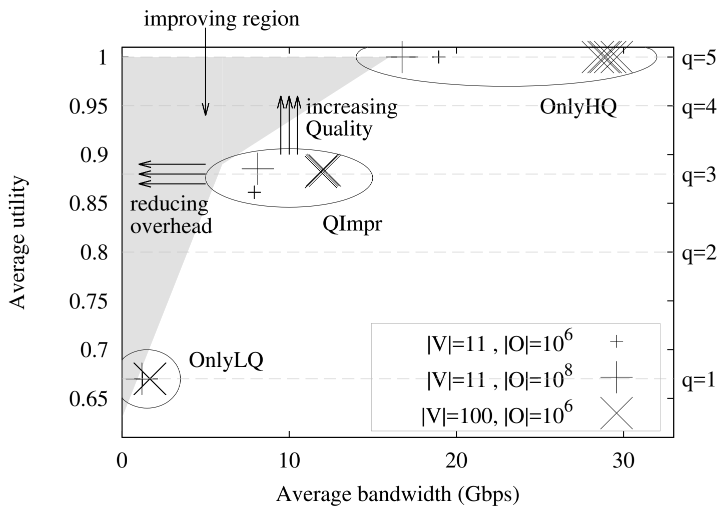


Figure 6.3: Simulation results showing the utility vs. bandwidth trade-off

of the best subset may depend on the utility, cache/upstream ratio, topology, request load, popularity skew, etc. which requires future work.

Summary

In this part we tackled the problem of optimal content distribution in cache-enabled networks, by explicitly taking into account multiple representations of the same video, each having a different utility perceived by users. Despite its relevance, this has not sufficiently been investigated so far. The need for caching techniques that, apart from the general ones, are optimized for video traffic is enforced by the prevalence of this traffic on the other types and its inherent cacheability. We find the optimal caching solution that maximizes user utility and we contrast it against several candidate strategies along the user experience angle. We study the fundamental properties of the solution to infer important guidelines to optimize object-level caching in video delivery. We leverage these guidelines in designing a distributed solution that we benchmark via event-driven simulation. Our key findings suggest that (i) the quality at which each object should be cached is inversely related to its popularity, (ii) a balance between user perceived utility and bandwidth usage is possible by means of intelligent caching distributed policy of which *QImpr*, the one proposed in Chapter 6, is an example.

Part III

Caching Encrypted Content

Introduction to Caching of Encrypted Content

It is widely known that content delivery over the Internet represents a sizeable and increasing fraction of the overall traffic demand. Furthermore most of the content, including video, is carried over HTTP connections: this evolution of the last decade was not among those forecasted for the IP hourglass model evolution [186], and is rather a choice of practical convenience. This evolution has a tremendous practical relevance, to the point that HTTP was overtly recognized [187] and proposed [188] as the new de facto "thin waist" of the TCP/IP protocol family.

In very recent times, we are on the verge of yet another shift of the thin waist: we indeed observe that the fraction of traffic delivered through HTTPS has already passed 50% [9], and it is expected to increase, as the IETF Internet Architecture Board (IAB) recommends "protocol designers, developers, and operators to make encryption the norm for Internet traffic" [10]. Besides the IAB recommendation, Content Providers (CP) are already heavily relying on encryption to both protect the privacy of their users, as well as sensitive information (related to user preferences) of their own business.

This evolution toward an all-encrypted Internet creates a tussle between security and efficiency. Today's Internet heavily relies on middleboxes such as NATs (to combat the scarcity of IPv4 addresses) and transparent or proxy caches [189] (to relieve traffic load). However, some of these middleboxes will simply fail to operate in tomorrow's Internet with end-to-end encryption: for example, end-to-end encryption renders caching useless, since multiple transfers of the same object generate different streams if the same object is encrypted with different keys. At times where the design of the new 5G architecture strives to reduce latency, increase the available bandwidth and better handle mobility, this tradeoff is especially unfortunate, as distributed caches represent a natural way to reduce latency, reduce bandwidth usage and to cope with mobility avoiding long detours to anchor points.

This architectural evolution calls for a redesign of the current operations involving both Internet Service Providers (ISP) and Content Providers (CP): by design, the solutions should preserve business-critical CP information (e.g., information about content popularity, user preferences) on the one hand, while allowing for a deeper integration of caches in the ISP architecture (e.g., in 5G femto-cells) on the other hand.

In this part we address this issue by proposing a content-oblivious algorithm that manages the storage space of an ISP cache that delivers *encrypted content*: the algorithm *dynamically partitions* the cache storage among various CPs so as to maximize the cache hit rate (hence, the bandwidth savings). The most important feature of the algorithm is that in order to protect business-critical information of the CPs, the ISP only needs to measure the *aggregated miss rates* of the individual CPs. In Chapter 7, we analytically prove that relying on the measurement of aggregated miss rates only, our algorithm converges close to the optimal allocation that maximizes the overall cache hit rate, and provide a bound on the gap to the optimal allocation. In Chapter 8, we show via simulations under realistic scenarios the feasibility and good performance of the proposed algorithm.

Most of the content of this part is extracted from [14].

Chapter 7

Analysis of Caching of Encrypted Content

This chapter covers the theoretical aspects of our content-oblivious mechanism. We define the model of the system we envisaged and the underlying assumptions in Sec. 7.1. In Sec. 7.2, we present our *Stochastic Dynamic Cache Partitioning* (SDCP) algorithm, whose goal is to iteratively adjust the allocation of the cache space among CPs, based on the measurement of the miss rate, in order to improve the overall hit ratio. The rest of the chapter is devoted to prove the convergence of SDCP. We first present some mathematical background Sec. 7.3, selecting from the literature on Convex Optimization, Stochastic Optimization, Control Theory and Operation Research the results that we need for our proof. Then we show in Sec. 7.4 that the algorithm makes all the variables vary in a correct way, i.e. during all the iterations each variable assumes values belonging to the respective admissible set, which is important to assure the correctness of the algorithm. The main convergence result is presented in Sec. 7.5, while we bound the distance between the allocation to which the algorithm converges and the optimal one in Sec. 7.6.

7.1 System Model

We consider a cache with a storage size of K slots (e.g., in units of MB) maintained by an operator and shared by P content providers (CPs). The operator is not aware of what content the individual slots store and only decides how to partition the slots among CPs.

We denote by $\theta_p \in \mathbb{Z}_{\geq 0}$ the number of cache slots allocated to CP p, which it can use for caching its most popular contents. We define the set of feasible cache allocation vectors

$$\Theta \triangleq \{\boldsymbol{\theta} \in \mathbb{Z}^P_{\geq 0} | \sum_{p=1}^{P} \theta_p \leq K\} \subset \mathbb{Z}^P_{\geq 0}. \tag{7.1}$$

We consider that the arrival of requests for content can be modeled by a stationary process, and the number of arrivals over a time interval of length T can be *bounded* by some positive constant $A(T)$. This assumption is reasonable as requests are generated by a finite customer population, and each customer can generate requests at a bounded rate in practice. Upon reception of a request for a content of the CPs that share the cache, the request can either generate a cache hit (for content stored in the CP partition at time of the request) or a cache miss (otherwise). Formally, we denote the *expected* cache miss rate (i.e., expected number of misses per time unit) of CP p when allocated θ_p slots of storage by $L_p(\theta_p)$. We make the reasonable assumption that L_p is decreasing and strictly convex on $[0 \dots K]$. This assumption corresponds to that having more storage decreases the miss intensity (in expectation) with a decreasing marginal gain, and each CP would in principle have enough content to fill the entire storage. For convenience we define the cache miss intensity vector

$$\vec{\mathbf{L}}(\boldsymbol{\theta}) \triangleq (L_1(\theta_1), \dots, L_P(\theta_P))^T \tag{7.2}$$

Finally, we define the overall expected cache miss intensity

$$L(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{P} L_p(\theta_p). \tag{7.3}$$

Motivated by the increasing prevalence of encrypted content delivery, we assume that the operator cannot observe what content an individual request is for, but it can observe the number of content requests received by a CP and the corresponding number of cache misses.

Given a static cache partitioning $\boldsymbol{\theta} \in \Theta$, the observed number of content requests and the number of cache misses would form a stationary sequence when measured over subsequent time intervals. The objective of the operator is to find the optimal allocation $\boldsymbol{\theta}^{OPT}$ that minimizes the overall *expected cache miss intensity*, i.e.,

$$\boldsymbol{\theta}^{OPT} \in \arg\min_{\boldsymbol{\theta} \in \Theta} L(\boldsymbol{\theta}) \tag{7.4}$$

based on the measured cache miss intensity. In what follows we propose the Stochastic Dynamic Cache Partitioning (SDCP) algorithm that iteratively approximates the optimal allocation.

7.2 Partitioning Algorithm

The proposed SDCP algorithm is an iterative algorithm that is executed over time slots of fixed duration T. The pseudo code of the algorithm is shown in Alg. 1. For simplicity we present the algorithm assuming that P is even, but the case of an odd number of CPs can be handled by introducing a fictitious CP with zero request rate.

Table 7.1: Frequently used notation (with place of definition)

P	Number of content providers (CP)
K	Available cache slots
K'	Allocated cache slots (7.6)
$\boldsymbol{\theta}$	Cache configuration
Θ	Set of feasible cache allocations (7.1)
$\mathcal{C}$	Set of allowed virtual cache allocations (7.5)
$L(\boldsymbol{\theta})$	Expected cache miss intensity (7.3)
$\vec{\mathbf{L}}$	Miss intensity vector (7.2)
$\bar{L}(\boldsymbol{\theta})$	Interpolant of the miss intensity (Lemma 12)
$\boldsymbol{\theta}^*$	Unique minimizer of $\bar{L}$ (Lemma 12)
$\mathbf{D}^{(k)}$	Perturbation vector (7.8)
T	Time slot length
$\boldsymbol{\varphi}$	Euclidean projection (7.10)
$\hat{\mathbf{g}}^{(k)}$	Stochastic subgradient (line 1 of Alg. 1)
$\bar{\mathbf{g}}(\boldsymbol{\theta})$	Subgradient of $\bar{L}$ (7.21)

At time slot k the algorithm maintains a virtual cache allocation $\boldsymbol{\theta}^{(k)}$. The virtual allocation is an allocation of K' storage slots among the CPs, i.e.,

$$\boldsymbol{\theta}^{(k)} \in \mathcal{C} \triangleq \left\{ \boldsymbol{\theta} \in \mathbb{R}^p | \mathbf{1}_P^T \cdot \boldsymbol{\theta} \triangleq K' \right\} \tag{7.5}$$

where

$$K' = K - P/2 \tag{7.6}$$

We will justify the introduction of K' and of $\mathcal{C}$ in the proof of Lemma 10. Observe that, while a feasible allocation is a discrete vector, a virtual allocation may be real. This is due to the computation we will perform to obtain it. Therefore, the virtual allocation needs to be discretized before being used in our algorithm, as we will explain later.

In order to obtain from $\boldsymbol{\theta}^{(k)}$ an integral allocation that can be implemented in the cache, we define the center-point function $\boldsymbol{\Gamma} : \mathbb{R}^P \to \mathbb{R}^P$, which assigns to a point in Euclidean space the center of the hypercube containing it, i.e.,

$$\begin{array}{rl} \gamma(x) \triangleq \lfloor x \rfloor + 1/2 & \forall x \in \mathbb{R} \\ \boldsymbol{\Gamma}(\boldsymbol{\theta}) \triangleq (\gamma(\theta_1), \dots, \gamma(\theta_P))^T , & \forall \boldsymbol{\theta} \in \mathcal{C} \end{array} \tag{7.7}$$

where we use $\lfloor \cdot \rfloor$ to denote the floor of a scalar or of a vector in the component-wise sense. Furthermore, we define the perturbation vector $\mathbf{D}^{(k)} = (D_1^{(k)}, \dots, D_P^{(k)})^T$ at time slot k, which is chosen independently and uniform at random from the set of -1,$+1$ valued zero-sum vectors

$$\mathbf{D}^{(k)} \in \mathcal{Z} \triangleq \left\{ \mathbf{z} \in \{-1, 1\}^P \left| \mathbf{z}^T \cdot \mathbf{1}_P = 0 \right. \right\}. \tag{7.8}$$

Algorithm 1: Stochastic Dynamic Cache Partitioning Algorithm

```
Choose an initial allocation $\boldsymbol{\theta}_0 \in \mathcal{C} \cap \mathbb{R}^P_{\geq 0}$
for $k = 0; ; k++$ do
    Generate $\mathbf{D}^{(k)}$
    ${}^{+}\boldsymbol{\theta}^{(k)} = \boldsymbol{\Gamma}(\boldsymbol{\theta}^{(k)}) + \frac{1}{2}\mathbf{D}^{(k)}$
    ${}^{-}\boldsymbol{\theta}^{(k)} = \boldsymbol{\Gamma}(\boldsymbol{\theta}^{(k)}) - \frac{1}{2}\mathbf{D}^{(k)}$
    Set the configuration to ${}^{+}\boldsymbol{\theta}^{(k)}$ for time $T/2$
    Measure ${}^{+}\vec{\mathbf{y}}^{(k)}$
    Set the configuration to ${}^{-}\boldsymbol{\theta}^{(k)}$ for a time $T/2$
    Measure ${}^{-}\vec{\mathbf{y}}^{(k)}$
    $\delta\vec{\mathbf{y}}^{(k)} =^{+} \vec{\mathbf{y}}^{(k)} -^{-} \vec{\mathbf{y}}^{(k)}$
    $\hat{\mathbf{g}}^{(k)} = \delta\vec{\mathbf{y}}^{(k)} \circ \mathbf{D}^{(k)} - \frac{1}{P} \cdot (\delta\vec{\mathbf{y}}^{(k)^T} \cdot \mathbf{D}^{(k)})\mathbf{1}_P$
    $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\varphi}(\boldsymbol{\theta}^{(k)} - a^{(k)}\hat{\mathbf{g}}^{(k)})$
end
```

Given $\boldsymbol{\Gamma}$ and $\mathbf{D}^{(k)}$ the algorithm computes two cache allocations to be implemented during time slot k,

$$\begin{aligned} {}^{+}\boldsymbol{\theta}^{(k)} &\triangleq \boldsymbol{\Gamma}(\boldsymbol{\theta}^{(k)}) + \tfrac{1}{2}\mathbf{D}^{(k)}, \\ {}^{-}\boldsymbol{\theta}^{(k)} &\triangleq \boldsymbol{\Gamma}(\boldsymbol{\theta}^{(k)}) - \tfrac{1}{2}\mathbf{D}^{(k)}. \end{aligned} \tag{7.9}$$

The algorithm first applies allocation ${}^{+}\boldsymbol{\theta}^{(k)}$ for $T/2$ amount of time and measures the cache miss rate ${}^{+}y_p^{(k)}$ for each provider $p = 1, \ldots, P$. It then applies allocation ${}^{-}\boldsymbol{\theta}^{(k)}$ during the remaining $T/2$ amount of time in slot k and measures the cache miss rates ${}^{-}y_p^{(k)}$. The vectors of measured cache misses ${}^{-}\vec{\mathbf{y}}^{(k)} \triangleq ({}^{-}y_1^k, \ldots, {}^{-}y_P^{(k)})^T$ and ${}^{+}\vec{\mathbf{y}}^{(k)} \triangleq ({}^{+}y_1^k, \ldots, {}^{+}y_P^{(k)})^T$ are used to compute the impact $\delta y_p^{(k)} \triangleq^{+} y_p^{(k)} -^{-} y_p^{(k)}$ of the perturbation vector on the cache miss intensity of CP p, or using the vector notation $\delta\vec{\mathbf{y}}^{(k)} \triangleq^{+} \vec{\mathbf{y}}^{(k)} -^{-} \vec{\mathbf{y}}^{(k)}$.

Based on the measured miss rates, the algorithm then computes the allocation vector $\boldsymbol{\theta}^{(k+1)}$ for the $(k+1)$-th step. Specifically, it first computes (line 1, where $\circ$ denotes the Hadamard product) the update vector $\hat{\mathbf{g}}^{(k)}$, which we show in Corollary 16 to match in expectation a subgradient of the miss-stream interpolant $\bar{L}$, defined in the statement of Lemma 12. The $(k+1)$-th allocation moves from the k-th allocation in the direction of the update vector $\hat{\mathbf{g}}^{(k)}$, opportunely scaled by a *step size* $a^{(k)} > 0$. Additionally, denoting with $\mathbb{R}_{\geq 0}$ the set of non-negative numbers, $\boldsymbol{\theta}^{(k+1)}$ is computed using the Euclidean projection $\boldsymbol{\varphi} : \mathcal{C} \to \mathcal{C} \cap \mathbb{R}^P_{\geq 0}$, defined as

$$\boldsymbol{\varphi}(\boldsymbol{\theta}) \triangleq \arg \min_{\boldsymbol{\theta}' \in \mathcal{C} \cap \mathbb{R}^P_{\geq 0}} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \tag{7.10}$$

Several remarks are worth making. First, we will show in Lemma 9 that the equation above admits a unique solution and thus the definition is consistent. Second, we will show in Lemma 10 that $\hat{\mathbf{g}}$ computed as in line 1 guarantees that the update $\boldsymbol{\theta}^{(k)} - a^{(k)}\hat{\mathbf{g}}^{(k)}$ at line 1 lies inside $\mathcal{C}$. Nonetheless, this update may have some negative components and we need to project it into $\mathcal{C} \cap \mathbb{R}_{\geq 0}$ by applying $\boldsymbol{\varphi}$, to ensure that the subsequent virtual allocation $\boldsymbol{\theta}^{(k+1)}$ is valid. Third, the *step size* $a^{(k)}$ must be chosen to satisfy $\sum_{k=1}^{\infty} a^{(k)} = \infty$ and $\sum_{k=1}^{\infty}(a^{(k)})^2 < \infty$ in order to guarantee convergence (see Theor. 17). Fourth, although the convergence of the proposed algorithm is guaranteed, for stationary content popularity, irrespectively of the choice of $a^{(k)}$ satisfying the above conditions, we point out that the step size plays an important role in determining the convergence speed, which we will numerically investigate in Chapter 8.

7.3 Preliminaries

Let us start by introducing the forward difference defined for functions on discrete sets.

Definition 5. For a function $\mathbf{F} : \mathbb{Z}^{q_1} \to \mathbb{R}^{q_2}$, $q_1, q_2 \geq 1$ the forward difference is

$$\Delta_n \mathbf{F}(\mathbf{x}) \triangleq \mathbf{F}(\mathbf{x} + n \cdot \mathbf{1}_{q_1}) - F(\mathbf{x}), \forall \mathbf{x} \in \mathbb{Z}^{q_1}, n \in \mathbb{Z} \setminus \{0\}$$

By abuse of notation, we will simply use $\Delta\mathbf{F}(\mathbf{x})$ to denote $\Delta_1\mathbf{F}(\mathbf{x})$.

The forward difference is convenient for characterizing convexity using the following definition [190].

Definition 6. A discrete function $F : \mathbb{Z} \to \mathbb{R}$ is strictly convex iff $x \to \Delta F(x)$ is increasing.

Furthermore, for a class of functions of interest we can establish the following.

Lemma 7. *Let $F : \mathbb{Z} \to \mathbb{R}$ decreasing and strictly convex, $x \in \mathbb{Z}$ and $n \in \mathbb{Z} \setminus \{0\}$ we have*

$$\Delta_n F(x) > n\Delta F(x), \tag{7.11}$$

Proof:

We first show that $\forall x, y \in \mathbb{Z}$ such that $y > x$, the following holds

$$\Delta_n F(y) > \Delta_n F(x) \quad \text{if } n > 0 \tag{7.12}$$

$$\Delta_n F(y) < \Delta_n F(x) \quad \text{if } n < 0 \tag{7.13}$$

For $n > 0$ we can use Def. 6 to obtain

$$\Delta_n F(y) = \sum_{i=0}^{n-1} [F(y+i+1) - F(y+i)] = \sum_{i=0}^{n-1} \Delta F(y+i)$$

$$> \sum_{i=0}^{n-1} \Delta F(x+i) = \Delta_n F(x). \tag{7.14}$$

which proves (7.12).

For $n < 0$ algebraic manipulation of the definition of the forward difference and (7.14) gives

$$\Delta_n F(y) = -\Delta_{|n|} F(y - |n|) < -\Delta_{|n|} F(x - |n|) = \Delta_n F(x),$$

which proves (7.13). To prove (7.11) for $n > 0$, observe that, thanks to Def. 6, each of the n terms of the last summation in (7.12) is lower bounded by $\Delta F(x)$. For $n < 0$ via algebraic manipulation we obtain

$$\Delta_n F(x) = -\sum_{i=1}^{|n|} \Delta F(x - i) > -\sum_{i=1}^{|n|} \Delta F(x) = -|n| \cdot \Delta F(x),$$

which proves (7.11) as $|n| = -n$. ■

Since SDCP generates virtual configurations whose components are not necessarily integer, we have to extend the discrete functions L_p to real numbers. Thanks to Theor. 2.2 of [191], we have the following existence result.

Lemma 8. *Given a discrete decreasing and strictly convex function $F : \mathbb{Z} \to \mathbb{R}$, there exists a continuous and strictly convex function $\bar{F} : \mathbb{R} \to \mathbb{R}$ that extends F, i.e., $F(x) = \bar{F}(x), \forall x \in \mathbb{Z}$. We call $\bar{F}$ the* interpolant *of F.*

Finally, we formulate an important property of the Euclidean projection $\boldsymbol{\varphi}$.

Lemma 9. *There is a unique function $\boldsymbol{\varphi}$ satisfying (7.10). Furthermore, $\boldsymbol{\varphi}$ satisfies*

$$\|\boldsymbol{\varphi}(\boldsymbol{\theta}) - \boldsymbol{\theta}'\| \leq \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|, \forall \boldsymbol{\theta} \in \mathcal{C}, \boldsymbol{\theta}' \in \mathcal{C} \cap \mathbb{R}^P_{\geq 0}, \tag{7.15}$$

i.e., $\boldsymbol{\varphi}(\boldsymbol{\theta})$ is no farther from any allocation vector than $\boldsymbol{\theta}$.

Proof: Observe that $\mathcal{C} \cap \mathbb{R}^P_{\geq 0}$ is a simplex, and thus closed and convex. Hence, the Euclidean projection $\boldsymbol{\varphi}$ is the unique solution of (7.10) [192]. Furthermore, the Euclidean projection is non-expansive (see, e.g., Fact 1.5 in [193]), i.e., for $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{C}$ it satisfies $\|\boldsymbol{\varphi}(\boldsymbol{\theta}) - \boldsymbol{\varphi}(\boldsymbol{\theta}')\| \leq \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$. Observing that if $\boldsymbol{\theta}' \in \mathcal{C} \cap \mathbb{R}^P_{\geq 0}$ then $\boldsymbol{\varphi}(\boldsymbol{\theta}') = \boldsymbol{\theta}'$ proves the result. ■

7.4 Consistency

We first have to prove that during each time slot the configurations ${}^{-}\boldsymbol{\theta}^{(k)}, {}^{+}\boldsymbol{\theta}^{(k)}$ that SDCP imposes on the cache are feasible. This is non-trivial, as the operators used in computing the allocations are defined on proper subsets of $\mathbb{R}^p$. The following lemma establishes that the allocations computed by SDCP always fall into these subsets.

Lemma 10. *The allocations $\boldsymbol{\theta}^{(k)}$ are consistent in every time slot, as they satisfy*

1. $\boldsymbol{\theta}^{(k)} - a^{(k)} \hat{\mathbf{g}}^{(k)} \in \mathcal{C}$,
2. $\boldsymbol{\theta}^{(k+1)} \in \mathcal{C} \cap \mathbb{R}^P_{\geq 0}$,

3. $^+\theta^{(k)}, ^-\theta^{(k)} \in \Theta$.

Proof: Recall that $\theta_0 \in \mathcal{C} \cap \mathbb{R}^P_{\geq 0}$. To show (a) observe that

$$\hat{\mathbf{g}}^{(k)} \cdot \mathbf{1}_P = \sum_{j=1}^{P} \delta y_p^{(k)} \cdot D_p^{(k)} - \sum_{j=1}^{P} \delta y_p^{(k)} \cdot D_p^{(k)} = 0 \tag{7.16}$$

and thus if $\theta^{(k)} \in \mathcal{C}$, then $\theta^{(k)} - a^{(k)}\hat{\mathbf{g}}^{(k)} \in \mathcal{C}$. The definition of the Euclidean projection (7.10) and (a) together imply (b). Finally, observe that

$$\mathbf{1} \cdot^+ \theta^{(k)} = \mathbf{1} \cdot \lfloor \theta \rfloor + \frac{P}{2} \leq \mathbf{1} \cdot \theta + \frac{P}{2} \leq K' + \frac{P}{2} = K, \tag{7.17}$$

which proves (c). Note that the above motivates the choice of K' in the definition of the set of virtual allocations $\mathcal{C}$, as if $K' > K - \frac{P}{2}$ then $^+\theta^{(k)}, ^-\theta^{(k)} \in \Theta$ may be violated due to the use of the mapping γ and $\mathbf{D}^{(k)}$ in (7.9). ■

7.5 Convergence

To prove convergence of SDCP, we first consider the relationship between the measured miss rates $^+y_p^{(k)}$ and $^-y_p^{(k)}$ and the expected miss intensities $L_p(^+\theta_p^{(k)})$ and $L_p(^-\theta_p^{(k)})$, respectively. We define the measurement noise

$$\begin{aligned} ^+\varepsilon^{(k)} &\triangleq^+ \vec{\mathbf{y}}^{(k)} - \vec{\mathbf{L}}(^+\theta^{(k)}) \\ ^-\varepsilon^{(k)} &\triangleq^- \vec{\mathbf{y}}^{(k)} - \vec{\mathbf{L}}(^-\theta^{(k)}), \end{aligned} \tag{7.18}$$

and the corresponding differences

$$\begin{aligned} \delta\varepsilon^{(k)} &\triangleq \quad ^+\varepsilon^{(k)} -^- \varepsilon^{(k)} \\ \delta\vec{\mathbf{L}}^{(k)} &\triangleq \vec{\mathbf{L}}(^+\theta^{(k)}) - \vec{\mathbf{L}}(^-\theta^{(k)}). \end{aligned} \tag{7.19}$$

Observe that $\mathbf{D}^{(k)}$, $^+\vec{\mathbf{y}}^{(k)}$ and $^-\vec{\mathbf{y}}^{(k)}$ are random variables and form a stochastic process. Using these definitions we can formulate the following statement about the measured miss rates.

Lemma 11. *The conditional expectation of the measurement noise and its difference satisfy*

$$\mathbb{E}[\delta\varepsilon^{(k)} | \theta^{(k)}] = \mathbf{0}_p \tag{7.20}$$

Proof: Observe that due to the stationarity of the request arrival processes we have $\mathbb{E}[^+\varepsilon^{(k)}|\theta^{(k)}] = 0$ and $\mathbb{E}[^-\varepsilon^{(k)}|\theta^{(k)}] = 0$, which due to the additive law of expectation yields the result. ■

Intuitively, this is equivalent to saying that the sample averages provide an unbiased estimator of the miss rates. In what follows we establish an analogous result for the update vector $\hat{\mathbf{g}}^{(k)}$ with respect to a subgradient of the interpolant $\bar{L}$ of the expected miss intensity L, which itself is a discrete function. We define and characterize $\bar{L}$ in the following lemma, which recalls known results from convex optimization.

Lemma 12. *Given the interpolants $\bar{L}_p$ of the expected miss intensities L_p of the CPs and defining the interpolant of L as $\bar{L}(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{P} \bar{L}_p(\theta_p), \forall \boldsymbol{\theta} \in \mathbb{R}^P_{\geq 0}$, $\bar{L}$ is strictly convex and admits a unique minimizer $\boldsymbol{\theta}^*$ in $\mathcal{C} \cap \mathbb{R}^P_{\geq 0}$.*

Proof: Recall that each interpolant $\bar{L}_p$ of L_p is strictly convex as shown in Lemma 8. The strict convexity of $\bar{L}$ can then be obtained applying Theor. 1.17 of [194]. Then, we observe that $\boldsymbol{\theta}^*$ is the solution to a convex optimization problem with a strictly convex objective function, which is unique (Sec. 4.2.1 of [195]. ■

For completeness, let us recall the definition of a subgradient of a function from (see, e.g., [196]).

Definition 13. Given a function $\bar{L} : \mathbb{R}^p \to \mathbb{R}$, a function $\bar{\mathbf{g}} : \mathcal{C} \subseteq \mathbb{R}^P \to \mathbb{R}^P$ is a subgradient of $\bar{L}$ over $\mathcal{C}$ iff

$$\bar{L}(\boldsymbol{\theta}') - \bar{L}(\boldsymbol{\theta}) \geq \bar{\mathbf{g}}(\boldsymbol{\theta})^T \cdot (\boldsymbol{\theta}' - \boldsymbol{\theta}), \forall \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{C}$$

We are now ready to introduce a subgradient $\bar{\mathbf{g}}(\boldsymbol{\theta})$ for the interpolant of the expected cache miss intensity $\bar{L}$.

Lemma 14. *The function*

$$\bar{\mathbf{g}}(\boldsymbol{\theta}) \triangleq \Delta\vec{\mathbf{L}}^{(k)}(\lfloor \boldsymbol{\theta} \rfloor) - \frac{1}{P} \cdot \Delta L(\lfloor \boldsymbol{\theta} \rfloor) \cdot \mathbf{1}_P. \tag{7.21}$$

is a subgradient of $\bar{L}$ over $\mathcal{C} \cap \mathbb{R}^P_{\geq 0}$.

Proof: Observe that for $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{C}$

$$\bar{\mathbf{g}}(\boldsymbol{\theta})^T \cdot (\boldsymbol{\theta}' - \boldsymbol{\theta}) = \Delta\vec{\mathbf{L}}^{(k)}(\lfloor \boldsymbol{\theta} \rfloor)^T \cdot (\boldsymbol{\theta}' - \boldsymbol{\theta})$$

$$-\frac{1}{P} \cdot \Delta L(\lfloor \boldsymbol{\theta} \rfloor) \cdot \left[\mathbf{1}_P^T \cdot (\boldsymbol{\theta}' - \boldsymbol{\theta})\right].$$

At the same time, for $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{C}$ we have

$$\mathbf{1}_P^T \cdot (\boldsymbol{\theta}' - \boldsymbol{\theta}) = (\mathbf{1}_P^T \cdot \boldsymbol{\theta}' - \mathbf{1}_P^T \cdot \boldsymbol{\theta}) = K' - K' = 0.$$

Therefore, for any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{C}$

$$\bar{\mathbf{g}}(\boldsymbol{\theta})^T \cdot (\boldsymbol{\theta}' - \boldsymbol{\theta}) = \Delta\vec{\mathbf{L}}^{(k)}(\lfloor \boldsymbol{\theta} \rfloor) \cdot (\boldsymbol{\theta}' - \boldsymbol{\theta}) \tag{7.22}$$

Thus, according to Def. 13, in order to show that $\bar{\mathbf{g}}$ is a subgradient of $\bar{L}$ it suffices to show that

$$\sum_{p=1}^{P} \left[\bar{L}_p(\theta'_p) - \bar{L}_p(\theta_p)\right] \geq \sum_{p=1}^{P} \Delta L_p(\lfloor \theta_j \rfloor) \cdot (\theta'_p - \theta_p). \tag{7.23}$$

We now show that this holds component-wise. If $\lfloor\theta'_p\rfloor - \lfloor\theta_p\rfloor = 0$, then the above clearly holds. Otherwise, if $n = \lfloor\theta'_p\rfloor - \lfloor\theta_p\rfloor \neq 0$ we apply a well known property of convex functions (Theor. 1.3.1 of [197]) to obtain:

$$\begin{aligned}\frac{\bar{L}_p(\lfloor\theta'_p\rfloor) - \bar{L}_p(\lfloor\theta_p\rfloor)}{(\lfloor\theta'_p\rfloor - \lfloor\theta_p\rfloor)} &\leq \frac{\bar{L}_p(\theta'_p) - \bar{L}_p(\theta_p)}{(\theta'_p - \theta_p)} \\ &\leq \frac{\bar{L}_p(\lfloor\theta'_p\rfloor + 1) - \bar{L}_p(\lfloor\theta_p\rfloor + 1)}{(\lfloor\theta'_p\rfloor + 1 - (\lfloor\theta_p\rfloor + 1))}\end{aligned}$$

which, by Def. 5, can be rewritten as:

$$\frac{\Delta_n L_j(\lfloor\theta_j\rfloor)}{n} \leq \frac{\bar{L}_p(\theta'_p) - \bar{L}_p(\theta_p)}{\theta'_p - \theta_p} \leq \frac{\Delta_n L_j(\lfloor\theta_j + 1\rfloor)}{n} \tag{7.24}$$

For $n > 0$ we can use the first inequality of (7.24) and Lemma 7 to obtain

$$\bar{L}_p(\theta'_p) - \bar{L}_p(\theta_p) \geq \Delta L_j(\lfloor\theta_j\rfloor) \cdot (\theta'_p - \theta_p). \tag{7.25}$$

For $n < 0$ we can use the second inequality of (7.24) and Lemma 7 to obtain

$$\frac{\bar{L}_p(\theta'_p) - \bar{L}_p(\theta_p)}{\theta'_p - \theta_p} \leq \Delta L_j(\lfloor\theta_j + 1\rfloor) \leq \Delta L_j(\lfloor\theta_j\rfloor), \tag{7.26}$$

and by multiplying the first and the second term of (7.26) by $\theta'_p - \theta_p$ (which is negative since $n = \lfloor\theta'_p\rfloor - \lfloor\theta_p\rfloor$ is negative), we obtain the result.

■

The subgradient $\bar{\mathbf{g}}$ will be central to proving the convergence of SDCP, but it cannot be measured directly. The next proposition establishes a link between the update vector $\hat{\mathbf{g}}^{(k)}$, which we compute in every time slot, and the subgradient $\bar{\mathbf{g}}$.

Proposition 15. *The update vector $\hat{\mathbf{g}}^{(k)}$ is composed of the subgradient $\bar{\mathbf{g}}$ plus a component due to the noise,*

$$\hat{\mathbf{g}}^{(k)} = \bar{\mathbf{g}}(\boldsymbol{\theta}^{(k)}) + \delta\boldsymbol{\varepsilon}^{(k)} \circ \mathbf{D}^{(k)} - \frac{1}{P} \cdot \left[\delta\boldsymbol{\varepsilon}^{(k)^T} \cdot \mathbf{D}^{(k)}\right] \mathbf{1}_P.$$

Proof: We first apply (7.19) to obtain

$$\begin{aligned}\hat{\mathbf{g}}^{(k)} &= \delta\vec{\mathbf{L}}^{(k)} \circ \mathbf{D}^{(k)} - \frac{1}{P} \cdot (\delta\vec{\mathbf{L}}^{(k)^T} \cdot \mathbf{D}^{(k)})\mathbf{1}_P \\ &+ \delta\boldsymbol{\varepsilon}^{(k)} \circ \mathbf{D}^{(k)} - \frac{1}{P} \cdot (\delta\boldsymbol{\varepsilon}^{(k)^T} \cdot \mathbf{D}^{(k)})\mathbf{1}_P\end{aligned} \tag{7.27}$$

Consider now a particular realization of the random variable $\mathbf{D}^{(k)}$. We can express component p of $\delta\vec{\mathbf{L}}^{(k)} \circ \mathbf{D}^{(k)} = \left[\vec{\mathbf{L}}({}^{+}\boldsymbol{\theta}^{(k)}) - \vec{\mathbf{L}}({}^{-}\boldsymbol{\theta}^{(k)})\right] \circ \mathbf{D}^{(k)}$ as

$$\left[L_p\left(\Pi\left(\theta_p^{(k)}\right) + \frac{1}{2}D_p^{(k)}\right) - L_p\left(\Pi\left(\theta_p^{(k)}\right) - \frac{1}{2}D_p^{(k)}\right)\right]$$

$$\cdot D_p^{(k)}$$

$$= \left[L_p\left(\Pi\left(\theta_p^{(k)}\right) + \frac{1}{2}\right) - L_p\left(\Pi\left(\theta_p^{(k)}\right) - \frac{1}{2}\right)\right]$$

$$= \left[L_p\left(\lfloor\theta_p^{(k)}\rfloor + 1\right) - L_p\left(\lfloor\theta_p^{(k)}\rfloor\right)\right],$$

where the first equality can be easily verified assuming that $D_p^{(k)} = -1$ and then assuming that it is $D_p^{(k)} = 1$. We thus obtain

$$\delta\vec{\mathbf{L}}^{(k)} \circ \mathbf{D}^{(k)} = \Delta\vec{\mathbf{L}}(\lfloor\boldsymbol{\theta}^{(k)}\rfloor)$$

and in scalar form

$$\delta\vec{\mathbf{L}}^{(k)^T} \cdot \mathbf{D}^{(k)} = \Delta L(\lfloor\boldsymbol{\theta}^{(k)}\rfloor).$$

By substituting these in (7.27)

$$\hat{\mathbf{g}}^{(k)} = \Delta\vec{\mathbf{L}}(\lfloor\boldsymbol{\theta}^{(k)}\rfloor) - \frac{1}{P} \cdot \Delta L(\lfloor\boldsymbol{\theta}^{(k)}\rfloor) \cdot \mathbf{1}_P$$

$$+\delta\boldsymbol{\varepsilon}^{(k)} \circ \mathbf{D}^{(k)} - \frac{1}{P} \cdot (\delta\boldsymbol{\varepsilon}^{(k)^T} \cdot \mathbf{D}^{(k)})\mathbf{1}_P,$$

and using (7.21), we obtain the result. ■

Furthermore, thanks to Lemma 11, the second term of (7.27), which is due to the noise, is zero in expectation, which provides the link between the update vector $\hat{\mathbf{g}}^{(k)}$ and the subgradient $\bar{\mathbf{g}}(\boldsymbol{\theta}^{(k)})$.

Corollary 16. *The conditional expectation of* $\hat{\mathbf{g}}^{(k)}$ *is* $\mathbb{E}[\hat{\mathbf{g}}^{(k)}|\boldsymbol{\theta}^{(k)}] = \bar{\mathbf{g}}(\boldsymbol{\theta}^{(k)})$ *and thus* $\hat{\mathbf{g}}^{(k)}$ *is a stochastic subgradient of* $\bar{L}$*, i.e.* $\mathbb{E}[\hat{\mathbf{g}}^{(k)}] = \bar{\mathbf{g}}(\boldsymbol{\theta}^{(k)})$.

This leads us to the following theorem.

Theorem 17. *The sequence* $\boldsymbol{\theta}^{(k)}$ *generated by SDCP converges in probability to the unique minimizer* $\boldsymbol{\theta}^*$ *of* $\bar{L}$*, i.e., for arbitrary* $\delta > 0$

$$\lim_{k\to\infty} Pr\{\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^*\| > \delta\} = 0.$$

Proof:

The proof of convergence is based on supermartingales, similar to (Theor. 46 in [196]), with the difference that our proof holds for Euclidean projection-based stochastic subgradients. Let us compute

$$\begin{aligned}
\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^*\|^2 &= \|\boldsymbol{\varphi}(\boldsymbol{\theta}^{(k)} - a^{(k)}\hat{\mathbf{g}}^{(k)}) - \boldsymbol{\theta}^*\|^2 \\
&\leq \|\boldsymbol{\theta}^{(k)} - a^{(k)}\hat{\mathbf{g}}^{(k)} - \boldsymbol{\theta}^*\|^2 \\
&= \|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^*\|^2 - 2a^{(k)} \cdot (\hat{\mathbf{g}}^{(k)})^T \cdot (\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^*) \\
&\quad +(a^{(k)})^2 \cdot \|\hat{\mathbf{g}}^{(k)}\|^2, \qquad (7.28)
\end{aligned}$$

where the first inequality is due to Lemma 9. Thanks to cor 16 and Def. 13

$$\left(\mathbb{E}[\hat{\mathbf{g}}^{(k)}|\theta^{(k)}]\right)^T \cdot (\theta^{(k)} - \theta^*) = \left(\bar{\mathbf{g}}(\theta^{(k)})\right)^T \cdot (\theta^{(k)} - \theta^*) \geq 0.$$

Recall that the number of arriving requests per time slot $A(T)$ is bounded, and thus $\|\hat{\mathbf{g}}^{(k)}\|^2$ is bounded, i.e., $\|\hat{\mathbf{g}}^{(k)}\|^2 \leq c$ for some $0 < c < \infty$. Hence, applying the expectation to (7.28)

$$\mathbb{E}\left[\|\theta^{(k+1)} - \theta^*\|^2 \middle| \theta^{(k)}\right] \leq \|\theta^{(k)} - \theta^*\|^2 + c(a^{(k)})^2. \tag{7.29}$$

Defining the random variable

$$z_k \triangleq \|\theta^{(k)} - \theta^*\|^2 + c\textstyle\sum_{s=k}^{\infty}(a^{(s)})^2,$$

it can be easily verified that (7.29) is equivalent to the inequality $\mathbb{E}[z_{k+1}|z_k, \dots, z_1] \leq z_k$. Consequently, $\{z_k\}_{k=1}^{\infty}$ is a supermartingale and converges almost surely to a limit z^*. Recalling now one of the required properties of the step size sequence, i.e., $\lim_{k\to\infty} \sum_{s=k}^{\infty}(a^{(k)})^2 = 0$, we have that the sequence $\{\|\theta^{(k)} - \theta^*\|^2\}$ also converges to z^* with probability one.

We now show by contradiction that the limit z^* is equal to zero. If this were not true, then one could find $\epsilon > 0$ and $\delta > 0$ such that, with probability $\delta > 0$, $\|\theta^{(k)} - \theta^*\| \geq \epsilon$ for all sufficiently large k, and thus

$$\sum_{k=0}^{\infty} a^{(k)} \cdot (\mathbb{E}[\hat{\mathbf{g}}^{(k)}|\theta^{(k)}])^T \cdot (\theta^{(k)} - \theta^*) = +\infty,$$

with probability δ, which would imply

$$\mathbb{E}\left[\sum_{k=0}^{\infty} a^{(k)} \cdot \left(\bar{\mathbf{g}}(\theta^{(k)})\right)^T \cdot (\theta^{(k)} - \theta^*)\right] = +\infty.$$

However, this would contradict the following relation (which is obtained by a recursion on (7.28 and then applying the expectation))

$$\begin{aligned}&\mathbb{E}[\|\theta^{(k+1)} - \theta^*\|^2] \leq \\ &\quad \|\theta^{(0)} - \theta^*\|^2 - 2\mathbb{E}\left[\sum_{s=0}^{k} a^{(s)} \cdot (\hat{\mathbf{g}}^{(s)})^T \cdot (\theta^{(s)} - \theta^*)\right] + \\ &\quad \mathbb{E}\left[\sum_{s=0}^{k} a^{(s)} \cdot \|\hat{\mathbf{g}}^{(s)})\|^2\right],\end{aligned}$$

as the left hand side cannot be negative. This proves the theorem.

■

7.6 Optimality Gap

It is worthwhile to note that the minimizer θ^* of $\bar{L}$ over $\mathcal{C} \cap \mathbb{R}^P_{\geq 0}$ may not coincide with its minimizer θ^{OPT} over Θ for two reasons: i) $K' < K$ and ii) θ^{OPT} is forced to have integer components while θ^* is can be a real vector. In what follows we show that the optimality gap $\|\theta^{OPT} - \theta^*\|_\infty$ is bounded by a small number, compared to the number of cache slots available.

Lemma 18. *The gap between the optimal solution* θ^{OPT} *and the configuration* θ^* *to which SDCP converges is* $\|\theta^* - \theta^{OPT}\|_\infty \leq (3/2)P$

Proof: We observe that θ^* is the optimal solution of the continuous Simple Allocation Problem (SAP), expressed as $\max\left(-\sum_{p=1}^{P} L_p(\theta_p)\right)$, subject to $\sum_{p=1}^{P} \theta_p \leq K'$ with $\theta \in \mathbb{R}^P_{\geq 0}$. K' usually referred to as *volume* and we denote with $\text{SAP}_{\text{cont}}(K')$ the problem above. The integer version of the SAP, which we denote by $\text{SAP}_{\text{int}}(K')$, is obtained from the problem above with the additional constraint $\theta \in \mathbb{Z}^p$. According to Cor. 4.3 of [198] there exists a solution $\hat{\theta}$ of $\text{SAP}_{\text{int}}(K')$ such that $\|\theta^* - \hat{\theta}\|_\infty \leq P$. The solution of the integer SAP can be constructed via the greedy algorithm presented in Sec. 2 of [198]. In our case, it consists of iteratively adding storage slots, one by one, each time to the CP whose miss intensity is decreased the most by using this additional slot. Based on this, it is easy to verify that a solution θ^{OPT} can be obtained starting from $\hat{\theta}$ and adding the remaining $K - K'$ slots. Therefore, $\|\theta^{OPT} - \hat{\theta}\|_\infty \leq P/2$, which implies

$$\|\theta^{OPT} - \theta^*\|_\infty \leq \|\theta^* - \hat{\theta}\|_\infty + \|\theta^{OPT} - \hat{\theta}\|_\infty \leq (3/2)P$$

■

Chapter 8

Performance of Caching of Encrypted Content

In this chapter, we evaluate the performance of SDCP through simulations performed in Octave. We first describe the evaluation scenario (Sec. 8.1) and show how the convergence speed is impacted by the choice of the step size sequence (Sec. 8.2). We then evaluate the sensitivity of SDCP to various system parameters (Sec. 8.3). Finally, recognizing that content catalogs are rarely static in the real world, we investigate the expected performance in the case of changing content catalogs (Sec. 8.4).

8.1 Evaluation Scenario

We consider a content catalog of 10^8 objects, in line with the literature [199] and recent measurements [200]. We partition the catalog in disjoint sub-catalogs, one per each CP. We assume that the content popularity in each sub-catalog follows Zipf's law with exponent $\alpha = 0.8$, as usually done in the literature [201]. We use a cache size of $K \in \{10^4, 10^5, 10^6\}$ objects (which corresponds to cache/catalog ratios of $10^{-4}, 10^{-3}$ and 10^{-2} respectively). In practice, the request arrival rate may depend on several factors such as the cache placement in the network hierarchy, the level of aggregation, the time of day, etc. Without loss of generality, we set the request arrival rate to $\lambda = 10^2$req/s, according to recent measurements performed on ISP access networks [200]. We compare the performance of SDCP to that of the optimal allocation $\boldsymbol{\theta}^{OPT}$ (*Opt*), and to that of the naive solution in which the cache space K is equally divided among all the CPs and is unchanged throughout the simulation (*Unif*).

While we proved convergence of SDCP, the speed of convergence is crucial to let the algorithm also be of practical use: we thus consider three step size sequences, as follows. In the *Reciprocal* scheme, the step size is $a^{(k)} = a/k$, where $a = \frac{1}{\|\hat{\mathbf{g}}^{(1)}\|} \cdot \frac{K'}{p}$. Observe that, with this choice, the Euclidean norm of

Algorithm 2: Conditional Step Size Sequence Computation

```
$a = \frac{p}{\|\hat{\mathbf{g}}^{(1)}\|_1} \cdot \frac{K'}{p}$
$b = a/10$
if $k \le k_{BS}$ ;                                        // Bootstrap Phase
then
    $a^{(k)} = a$
else if $k \le M$ ;                                        // Adaptive Phase
then
    Compute the miss-ratio $m^{(k)}$ during the current iteration
    Compute $m_{5th}$, i.e. the 5th percentile of the previous miss ratios
    $m^{(1)}, \dots, m^{(k-1)}$
    $\hat{a}^{(k)} = a^{(k-1)}/2$
    $\tilde{a}^{(k)} = a^{(k-1)} - \frac{a^{(k-1)}-b}{M-k+1}$
    $a^{(k)} = \begin{cases} \left(\min(\hat{a}^{(k)}, \tilde{a}^{(k)}), b\right) & m^{(k)} \le m_{5th} \\ \tilde{a}^{(k)} & \text{otherwise} \end{cases}$
else
    $a^{(k)} = a^{(k-1)} \cdot \left(1 - \frac{1}{(1+k)}\right)^{\frac{1}{2}+\epsilon}$ ;    // Moderate Phase
end
```

the first update $a^{(1)} \cdot \hat{\mathbf{g}}^{(1)}$ is $\frac{K'}{p}$. This step size allows to change this amount of slots in the allocation, thus obtaining a broad exploration of the cache allocation space at the very beginning.

In the *Moderate* scheme, step sizes decrease slowly, to avoid confining the exploration only to the beginning. We resort to guidelines of [202], and define the step size as $a^{(k)} = a^{(k-1)} \cdot \left(1 - \frac{1}{(1+M+k)}\right)^{\frac{1}{2}+\epsilon}$, where a is computed as above and M, ϵ are positive constants, which can be tuned to modify the decrease slope.

The third step size sequence, which we refer to as *Conditional*, is defined in Alg. 2. It consists of a *Bootstrap phase* (up to iteration k_{BS}) in which step sizes remain constant, thus allowing broad exploration. Then an *Adaptation* phase follows, up to iteration M, in which step sizes decrease, by default, linearly from an initial value a to a final value b. This decrease is steeper than linear when the miss ratio measured at the current iteration is smaller than the 5-th percentile of the miss ratio values observed so far. In this case the step size is halved, unless it already equals b. The intuition behind this phase is that we try to reduce the exploration extent every time we encounter a “good” allocation, i.e., an allocation that shows a small miss ratio compared to what we experienced so far. Note that we do not start immediately with the Adaptation phase, since we need to collect enough samples during the Bootstrap phase in order to correctly evaluate the quality of the current allocation. Finally, we continue with a *Moderate* phase, in which step sizes are updated as above and are asymptotically vanishing, thus guaranteeing convergence.

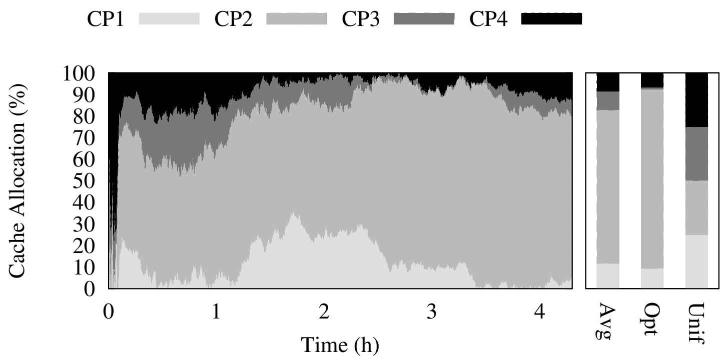


Figure 8.1: Evolution of the allocation of cache slots across CPs, with cache size $K = 10^5$ and *Conditional* step sizes. The three bars on the right represent the component-wise average of the allocated slots under SDCP allocation with Conditional steps (Avg), the Optimal (Opt) and Uniform (Unif) allocations.

After a preliminary evaluation, we set $\epsilon = 1/100$ as in [202] and $b = a/10$. We set k_{BS} and M, i.e., the duration of the bootstrap and adaptive phases, to the number of iterations in 6 minutes and 1 hour, respectively. While the duration of these phases is clearly tied to the arrival rate, and are expected to require tuning when ported to a different scenario, we point out that performance achieved with these choices remains satisfying under the different scenarios we consider.

8.2 Convergence

We first consider a cache size of $K = 10^5$ and 4 CPs, receiving 13%, 75%, 2% and 10% of requests, respectively. From Fig. 8.1, we can observe that, after a first exploration phase, the algorithm converges to a stable allocation. It is interesting to note that the average allocation (Avg), which is obtained by averaging each component of the allocation vector throughout the iterations, is very close to the optimal one, unlike the naïve uniform allocation policy.

Second, we consider a larger scenario with cache size $K = 10^6$ and $p = 10$ CPs, one of which we is a popular CP, to which 70% of requests are directed, followed by a second one receiving 24% of requests, other 6 CPs accounting for 1% each and the remaining two CPs receiving no requests. Fig. 8.2 shows the step sizes and the inaccuracy of the algorithm, i.e. the distance to the optimal allocation, measured as:

$$Error(\boldsymbol{\theta}) \triangleq \frac{\|\boldsymbol{\theta}^{OPT} - \boldsymbol{\theta}\|_\infty}{K} = \frac{\max_{j=1\ldots p} |\theta_p - \theta_p^{OPT}|}{K} \tag{8.1}$$

Observe that *Reciprocal* steps decrease too fast, which immediately limits the adaptation of the allocation, significantly slowing down convergence. Conversely, *Moderate* steps remain large for an overly long time, preventing the

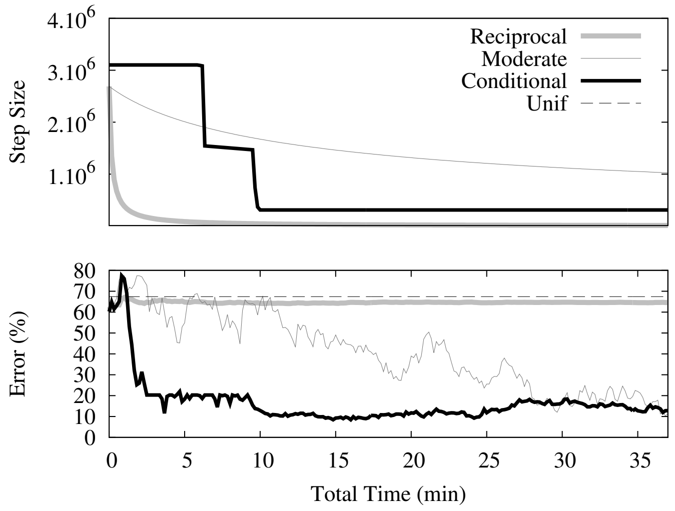


Figure 8.2: Error and step size sequence with cache size $K = 10^6$.

algorithm to keep the allocation in regions that guarantee good performance. *Conditional* steps show the best performance since in the Adaptation phase the step sizes are sharply decreased if the current allocation is providing a small miss ratio.

8.3 Sensitivity Analysis

We next study how the performance of SDCP is affected by the algorithm parameters and the scenario. We first focus on the time slot duration T. On the one hand, a small T implies that only few requests are observed in each time slot, which may result in a high noise ${}^{+}\boldsymbol{\varepsilon}^{(k)}, {}^{-}\boldsymbol{\varepsilon}^{(k)}$, and ultimately affects the accuracy of the update. On the other hand, a large T decreases the measurement noise, but allows updates to be made less frequently, which possibly slows down convergence.

To evaluate the impact of T, Fig. 8.3 shows the miss ratio measured over 1h for the default scenario. We consider SDCP with the three step size sequences, and compare it to the Uniform and to the Optimal allocations as benchmarks. The figure shows that SDCP with the Conditional step size sequence enhances the cache efficiency significantly. We also observe that an iteration duration of $T = 10s$ (corresponding to 100 samples on average per CP) represents a good compromise between a more accurate miss ratio estimation based on more samples (with large T) and a larger number of iterations at the cost of lower accuracy (with small T).

Fig. 8.4 shows the cache miss rate measured over 1h for a time slot length of $T = 10$s and for various cache sizes $K \in \{10^4, 10^5, 10^6\}$ and arrival rates

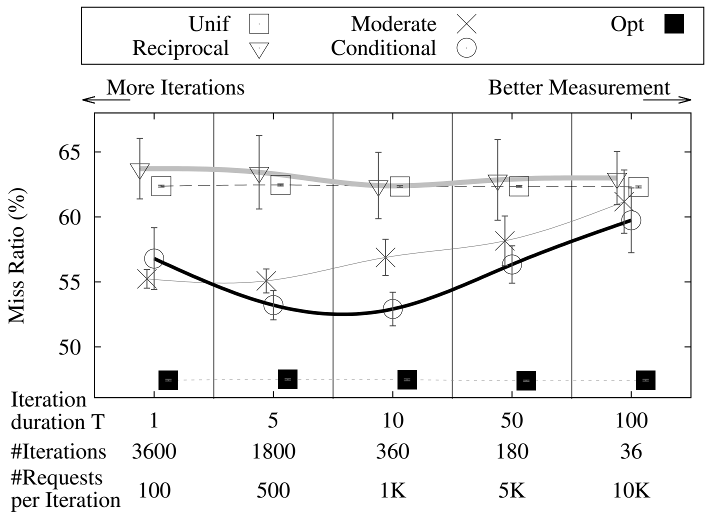


Figure 8.3: Impact of time-slot duration T on the average miss ratio (bars represent the 95% confidence interval over 20 runs).

$\lambda \in [1, 10^4]$. The figure confirms that the gains of SDCP hold for different cache sizes, and shows that the gain increases for large caches. To interpret the results for different arrival rates, recall that for any given time slot duration T, the average request rate affects the measurement noise. Fig. 8.4 confirms that the miss rate increases when the measurement noise is higher, i.e., for lower λ, but it also shows a very limited impact: the number of time slots in a relatively short time (in 1h, there are 360 time slots of duration $T = 10s$) allows SDCP to converge to a good cache configuration, in spite of the noise and the consequent estimation errors.

8.4 Changing Content Popularity

Recent studies [203] have observed that the catalog statistics vary over time. We show in this section that in order for SDCP to be robust to these variations, it suffices to periodically reinitialize the step sequence. To model changing content popularity, we adopt the model of [203], in which each object is characterized by a sequence of ON and OFF periods, with exponentially distributed duration T_{ON} and T_{OFF}, respectively. At each time instant, an object can be ON or OFF, and only ON objects attract requests. As in [203], we set the catalog size to $3.5 \cdot 10^6$ and the cache size to $K = 10^4$ objects. We set the average ON and OFF duration to 1 and 9 days, respectively. On average, we maintain the overall request rate of active objects equal to our default value $\lambda = 100 req/s$.

In Fig. 8.5 we compare *Unif* and *Conditional(τ)* that reinitialize the step sequence every τ amount of time. We consider $\tau \in \{3h, 1d, \infty\}$, i.e., 8 reini-

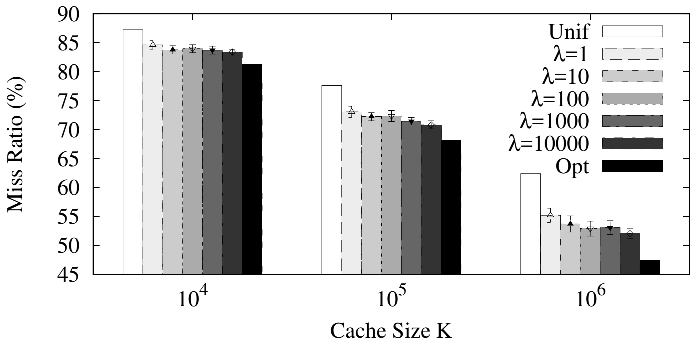


Figure 8.4: Miss rate measured over 1h for various average request rates λ and cache sizes K.

tializations per day, daily reinitialization, or no-reinitialization, respectively. As expected, reinitialization improves cache efficiency. Indeed, already after 3 hours of simulation, the evolution of the catalog misleads *Conditional* and *Conditional(1d)* (that overlap in this time interval) causing them to have performance worse than Unif. This is expected, since *Conditional(∞)* tries to converge to the optimal allocation, which is problematic in a non-stationary scenario. At the same time, it also shows that reinitializing step sequences as in *Conditional(3h)* is sufficient to respond to the catalog dynamics.

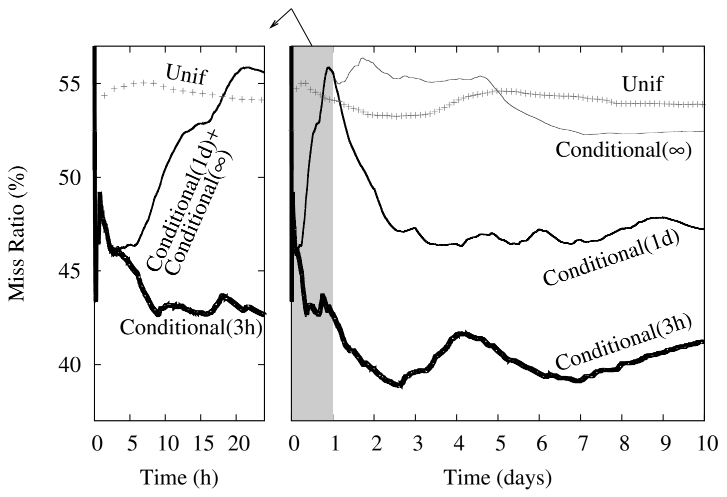


Figure 8.5: Miss ratio with varying content popularity. Step size sequences in *Conditional(τ)* are reset every τ interval. On the left the first day is zoomed, where *Conditional(∞)* and *Conditional($1d$)* correspond, since none of them have reinitialized the step sequence.

Summary

One of the main challenges of in-network caching nowadays is its incompatibility with encrypted content. Our work represents a first step in solving this challenge by proposing a simple and therefore appealing system design: Stochastic Dynamic Cache Partitioning requires solely the knowledge of aggregated cache miss-intensities, based on which it provably converges to an allocation with a small optimality gap. Simulation results show the benefits of the proposed algorithm under various scenarios, and results obtained under complex content catalog dynamics further confirm the algorithm to be applicable in scenarios of high practical relevance.

Part IV

Conclusion

Chapter 9

Summary of our Contribution

Network caching is one of the most promising techniques (i) to cope with the Internet traffic deluge, (ii) to meet the increasing expectation of the users in terms of quality of experience and (iii) to help ISPs and CPs contain their cost incurred to fulfill (i) and (ii). Nonetheless, most of the work in the current literature is about incremental improvements of proposals conceived during the 90s, when the Internet was very different from today's. Starting from this observation, we aim to provide new viewpoints on the subject in order to make caching i) efficient in enhancing the metrics that are the most relevant for the actors of the Internet, in particular ISPs and users, ii) suitable for the type of data that largely dominates the traffic today, i.e. video and iii) feasible and harmoniously integrated in the ecosystem.

In this thesis, we tackled three problems that are tightly coupled with the goals above, namely i) the reduction of the ISP operational cost, ii) caching decisions to properly handle the different quality representations available for videos and iii) caching of content encrypted by CPs, to maintain the secrecy of their business-sensitive information. Overall, we showed that network caching is a flexible tool whose benefits go beyond the classic hit ratio maximization, the hop reduction and the latency minimization, classically studied in the literature.

9.1 Cost Reduction

Part I is devoted to the problem of minimizing the inter-domain traffic cost for ISPs. When an ISP with caching capabilities receives a request for an object, it can serve it with a replica cached inside its network or, if no replica exists, it must retrieve it generating traffic on some inter-domain link. The ISP pays for the traffic generated there. Previous work has mainly aimed to maximize hit ratio and, as a consequence, minimize the inter-domain traffic. At first sight, cost minimization may appear as a direct consequence. We showed that it may not be true, due to the price heterogeneity among inter-domain links. Indeed, there are links regulated by settlement-free agreements, which can be used for free, and other links with different prices. Therefore, blindly reducing traffic across all of them do not bring all the potential saving. We showed that there is a trade-off between hit ratio maximization and cost reduction. In other words, if we manage to deploy a cache strategy that maximizes hit ratio, we are necessarily losing in terms of cost saving. Intuitively, to achieve cost minimization, we must preferentially store the expensive objects, i.e. the ones that we can only retrieve from expensive links. To show this trade-off we provide an optimization model and a greedy algorithm to solve it. While hit ratio maximization is achieved by storing the most popular object, cost minimization is obtained by storing the objects with the highest product of price and popularity. Doing this would require an omniscient offline policy that knows in advance the popularity of all the objects, which is unrealistic. For this reason, we also devised an online caching policy, which is distributed and stateless, each node taking decisions every time it is traversed by an object, independently from the other nodes and the previous objects observed. In more detail, we proposed a metacaching policy that accepts each objects with a probability proportional to its price. We gave a probabilistic model of our policy and compare it with simulation results to check its accuracy. We showed by means of large scale simulation that the online policy approximates the optimal solution and that the achieved cost saving is remarkable.

9.2 Caching of Video

In the second part we studied caching mechanisms specifically tailored for video delivery. Video represents most of the Internet traffic and is cacheable by nature. Therefore, caching is potentially particularly beneficial if applied to it. Nonetheless, video delivery is more complex than other types of transmission, in that we do not simply have to move video frames from the source to the users, but we have to move them in a "proper" way, with the appropriate quality level and the appropriate timeliness. These transmissions are regulated by intelligent control mechanisms, which are mostly conceived without having caches in mind. Therefore, the interaction between such mechanisms and caches is not easy to capture and to manage. More importantly, except some recent effort, caching and video delivery have not been jointly studied and have been investigated by different communities. As a consequence, caching mechanisms are not typically tailored for video. The main impairement lies in the basic assumptions: classic caching assumes that a user request can be satisfied by one and only one file. This assumption does not hold in the context of video delivery. Indeed, a video can have different representation, i.e. different files at different bit-rates and resolutions. Therefore, it is no more sufficient to decide whether to cache or not an object, since we must also select one of the available representations. To this aim, we introduced the *representation selection* problem as a new dimension to the cache design space. We first provide a MILP that jointly decides which objects to cache in a network of nodes, at which location, which representation to select and how to route objects toward users. The objective is to maximize the overall user utility, which depends on the quality served to users and is constrained by the limited available storage and link capacities. We obtained the optimal solution by running the MILP on a solver and we evaluated the performance, which represents the theoretical bound achievable. We also studied the structural properties of the optimum, learning two important and simple guidelines: i) we must not cache all the representations of the objects (that is what CDNs likely do), but just the "right" ones and ii) the right representation of a certain object depends on its popularity: cached quality must decrease with the popularity. As discussed above, the optimum obtained from the MILP can be seen as an offline policy, which is not implementable in reality. We thus devised an online caching policy that can be implemented in a real network of caches. To summarize its functioning, every time a request arrives for a video in the cache, the cache itself, apart from satisfying the request, issues another request for an improved representation of the same video. We checked, by means of simulation, that at regime the cache breakdown of this policy approximates the optimal one, i.e. it tends to cache at higher quality the objects that are more popular. We also explore by means of large scale simulation the trade-off between bandwidth and user utility: in order to provide the maximum utility, we risk to incur in a too large bandwidth usage, which may imply an excessive cost. We showed in our simulation that our online policy manages to find a balance balance between these metrics, guaranteeing a satisfying utility without having excessive bandwidth requirements.

9.3 Caching of Encrypted Content

Finally, we observed that the applicability range of classic caching techniques is becoming smaller and smaller due to the increasing adoption of HTTPS. Indeed, classic techniques require the visibility of the user requests and of the objects being sent. Given that most of the traffic is encrypted by the CPs and thus opaque to ISPs, the ISP cannot perform any cache operation. To solve this problem, we proposed *Content Oblivious* caching, whose contribution consist in the architecture and the algorithm to manage it. In more detail, in our vision caching is a service that ISP can provide to CPs. The cache space is partitioned in separated segments, each assigned to a CP. The CP can exclusively control its segment via a remote proxy process running on the cache server. All the objects are stored encrypted, with only the proxy of the respective CP being able to decrypt it. This allows at the same time CPs to maintain their exclusive control on their content and ISPs to enjoy the benefits of caching. This architecture let an interesting problem arise: how to allocate the cache space to the different CPs. Clearly, assigning the same space to all the CPs is inefficient, due to the difference in their catalog popularities. For this reason we devised a cache partitioning iterative algorithm, which, by only observing the miss rate, continuously adjusts the allocation. We analytically proved, leveraging stochastic subgradient-based techniques, that the algorithm converges boundedly close to the optimum, i.e. the allocation that guarantees the highest overall hit ratio. By means of simulation, we studied the performance of the algorithm, showing that it converges in an amount of time sufficiently small to permit to adapt to a time-changing catalog popularity. We also performed a study on the set up of the algorithm parameters, in order to get the best performance.

9.4 Final Remarks

Overall, we showed that, despite its more than twenty years of history, network caching still offers interesting research challenges, also very relevant for the business of the Internet companies.

Chapter 10

Discussion and Future Work

We point out that the goal of our research was not to provide ready-to-use solutions for production networks, but to reveal a potential of network caching that has not been sufficiently investigated so far. In other words, we are not aiming at enhancing techniques that already exist and are already mature. Rather, we provide new viewpoints on the caching problem and novel mechanisms (as repeatedly acknowledged during the process of peering review of our articles), which, as such, are not necessarily meant to be self-complete and represent the first step in their respective directions. In this chapter, we discuss and somehow "criticize" our proposals in order to reason about what the next steps should be toward their real deployment.

Before going into the details of the single proposal, we emphasize that we tackled three problems (each corresponding to a part of this thesis) separately and proposed independent respective solutions. Since we repeatedly claimed in this thesis that each of these problems are relevant, we would not support in a real network the adoption of the proposals in isolation. Rather, we would envisage to deploy them jointly. For this reason, an interesting future research direction would be to combine the three proposals together. For example, in Part I we only target cost minimization and we ignore the fact that objects, video in particular, have multiple representations. On the other hand, in Part II we aim to maximize user utility, ignoring the heterogeneity of inter-domain link prices. It would be interesting to investigate the trade-off that we expect to have between cost and user utility. Along this direction, we just examined the trade-off between the bandwidth and the user utility in Sec. 6.3 but considering bandwidth reduction, blindly across all the links and not considering their price, is not enough to include the cost into the picture, as already showed in Part I. Similarly, while in Part III we maximize hit ratio, we could embed there the user utility and the cost. These are all interesting future direction of our research that, furthermore, could also turn the research in a solution ready to be deployed on production networks. At the same moment, we do not regret the choice of having tackled the problems separately, as it was the only way to make them emerge clearly and to give strong, clean and simple claims about them.

It is unavoidable in research to start from simplifying assumptions, to make problems formalizable and tractable. As future research, we can incrementally remove them in order to obtain a more fine-grained description of the reality. In what follows, we individuate some details that could be added to our models. As an example, we assume all over this thesis that the objects have the same size, which is an assumption of most work on caching, but which does not hold in reality. We could study the impact of considering the size of the objects. We now highlight the possible improvement on each of the three proposals.

10.1 Cost Reduction

In Part I we suppose the cost of the inter-domain traffic on a certain link is proportional to the number of objects sent, which is an assumption of most related work. In reality, more complex price models rule payment between ISPs (like the 95% price model), which we could embed in our models.

In Chapter 3, we also look at the cache as a monolithic system, without considering how it is deployed, whether it is a unique server or a network of cache servers and, in the latter case, where these servers are placed and what are the bandwidth limits of the links connecting them. We could enrich our formulation by taking these other aspects into account.

Moreover, there are other techniques, orthogonal to caching, aimed at cost reduction, like the ones commented in Sec. 2.5.4. We could investigate if associating our caching techniques with them would further improve cost reduction capabilities or, on the contrary, some conflict among them would emerge. Another possible enhancement could be combining our metacaching policy with some cost-aware replacement policy.

In Part I, an intra-domain view is provided. All the nodes of the considered topologies are routers or caches inside an ISP boundary. We could zoom-out our study to observe what is the result of applying our intra-ISP policies to a network of ISPs. The analysis should also take into account the heterogeneity between ISPs. Indeed, as already pointed out in Sec. 2.1.2, only Local ISPs (and not Transit ISPs) have interest in caching. Moreover, we consider in this part a stationary popularity, while we could evaluate the performance in case of time-varying popularity, like we did in Sec. 8.4, when studying the problem of caching encrypted content.

We also point out that in this thesis we consider cache space to be a fixed value. We could instead consider the amount of storage as a variable. We could investigate how to choose the right amount, considering the cost of deployment and the advantages in terms of inter-domain traffic cost reduction and better service to users. We could also model the competition between ISPs, taking into account the possible abandonment of users toward other ISPs, in case of poor service, which is expected to represent an incentive to deploy caches.

10.2 Caching of Video

As for Part II, we assume that all the ISPs collaborate sharing their cache storage with no fees. It would be interesting to study the incentive of forming ISP coalitions or to revisit our model putting payment among ISPs into the picture. We could also investigate how cache content differentiates with respect to the node position in the network, due to the interaction and the filtering effect of neighbors. Moreover, while we assume that all the ISPs use the same cache policy, we could examine the case where policies are heterogeneous. As for the goal, we already discussed that fairness concerns may arise, due to the fact that, in order to increase the overall perceived utility, we must privilege the popular content, while serving the rest at lower quality. We could investigate more on this aspect especially considering that our MILP is suitable for this sort of investigation, being flexible enough to, for example, impose a minimum quality threshold to be met. As regards our online representation aware policy, QImpr, it forces the cache to issue a new request every time an object is requested and is not yet stored at the highest quality. Even a very unpopular object, requested just once, triggers this cache-generated request. We plan to consider to let the cache issue such requests not always, but with a certain probability. In this way, only the requests of popular objects have sufficient chances to trigger the cache-generated requests. This is expected to i) limit the overhead due to the transmission of such requests and the related objects retrieved and ii) reduce the number of cache replacements due to the arrival of the objects downloaded as a consequence of those cache-generated requests.

We also assume that the user perception is only related to the bit-rate of the video, independently from its category, while in the related literature it has been discovered that the perception changes depending on whether a video is about sport, or a documentary, an action movie or a drama. We also suppose that the devices from which users consume multimedia content are homogeneous, which is not true, given the widespread access from smartphones, tablets, desktop computers, HD televisions, etc. The utility perceived at a certain bit-rate when using a smartphone is different from when using a HD television. We could refine our model in order to take into account such heterogeneity.

Moreover, we could compare our results with other mechanisms based on Scalable Video Coding (SVC), in which each video chunk is decomposed in a base layer and several enhancement layers. If only the base layer is available, the video will be watched at low quality, and will improve if the additional layers are available. SVC is not widely used in current video distribution due to the overhead it imposes and its complexity. Nonetheless, it would be interesting to check if these negative aspects are compensated by a clear improvement in the user perceived utility.

Another contribution we could give to the scientific community is to refine the code of QImpr and make it available in a new version of the open source software ccnSim (like we did for the cost-aware caching), in order to help other researchers to investigate the caching of videos.

The most important point of improvement that we recognize regards the online policy, QImpr. Our goal in providing it was to show that the representation selection problem is addressable with implementable policies, but we are aware that our design needs to be enriched a lot before being deployable in real networks. The most relevant weakness is the lack of any control mechanism that takes into account the available bandwidth of the traversed links and the congestion condition. Therefore, the quality that is served to users is not dictated by bandwidth measures, but is "cache-driven", in that we serve to users the quality that is available on the cache, if any, otherwise we just serve the minimum. This may not always be the smartest choice. For example, suppose that links are completely unoccupied and that it would be possible to transmit across them all the requested objects at the highest quality. With QImpr, this is not immediately possible, as the first time an object is requested, it is served at the lowest quality, despite the availability of a larger bandwidth. Another related weakness is that, not taking into account the bandwidth available but only the cache content, QImpr may try to retrieve objects stored at high quality in a cache, even if the capacity available on the downstream link is not sufficient. All these problems could be addressed enriching QImpr with control mechanisms. We think that this could open other interesting problems related to the transmission of objects with different representations and available at different locations. We think the best way to evaluate such enriched QImpr would be the realization of a small testbed, with some real player downloading chunks and some real server serving them. The framework offered by TAPAS [204] is an excellent framework for such experimentation. By means of such testbed, we could also consider other metrics, like the probability of video reproduction stalls, which we have neglected in this thesis, although they are relevant in determining user perception.

10.3 Caching of Encrypted Content

In Part III, we suppose that each CP knows its most popular objects and it is able to place them in the assigned slots. This is an ideal assumption. We plan to extend our analysis removing it. We could,, for example, suppose that CPs have no information about the popularity, and they handle their allocated cache space with an online policy, like LCE and LRU (see Sec. 2.3.2). If we find that the expected miss-stream, computed via Che's approximation (see Sec. 4.2), respects the convexity assumptions we needed for the proof of the convergence of our SDCP algorithm (Chapter 7), we may be able to affirm that the algorithm still converges to the optimum.

Moreover, we can further improve the speed of convergence of the algorithm. Indeed, as it was presented in Part III, the algorithm has to learn the CP popularities from scratch. We are working on a revised version of our mechanism, in which the CPs declare the popularity of their catalog and the ISP can immediately compute the optimal allocation based on these declarations. The problem to solve is to convince CPs to communicate truthful declarations, which can be formalized in a game theoretical framework.

More generally, even if we tackle the specific problem of cache allocation across multiple CPs, the mechanism we proposed in Chapter 7 can be in general applied to the problem of minimizing a separable multi-variable convex functions. In the future we plan to analyze our algorithm under this more general light, showing that it performs better than other approaches that do not exploit separability (like [205]), extending for example the theoretical analysis of the convergence time of Chapter 5 of [202]. Our goal would be to make our algorithm a stochastic optimization tool to be used in automatic control problems, apart from the application that we presented here in Part III.

10.4 Beyond the Tackled Problems

It would be wrong to affirm that we covered in this thesis all the open problems on network caching. First, we expect some margin for improvement for the classic caching strategies. In particular, we would be interested in investigating the offline vs. online caching strategy duality (see Sec. 2.3). The former assumes that we know everything about content popularity, the latter assume we know nothing. We believe that the former is too optimistic, while the latter is too pessimistic. In reality, the a-priori knowledge lies somewhere in the middle. Content Providers cannot exactly predict the future requests but, at least, they have *some* information. For example Youtube can exploit its huge historical data to this aim. Therefore, instead of considering pure offline policies vs. pure online policies, we could consider a hybrid policy, which is able to exploit the *partial* information on content popularity. An architecture like the content oblivious one, proposed in Part III, is a good candidate to permit these kinds of policies, since it leaves the caching decisions to the CPs, thus allowing them to exploit their knowledge about the popularity of their content.

More importantly, caching may still have potential to achieve other goals, apart from the ones classically investigated in the ones examined in this thesis (cost reduction and user utility improvement). The most interesting contribution we expect from the research in network caching lies indeed, we believe, in proposing these new achievable goals. We expect these goals to naturally emerge due to the evolution of the Internet, like it was for the problems we decided to tackle. Summarizing, we expect network caching technique to continuously evolve, as long as the Internet evolves.

Acknowledgments

I would like to thank my supervisors Prof. Dario Rossi and Prof. Fabio Martignon, who continuously guided me. Their suggestions allowed me to improve not only my technical skills, but, more generally, my ability to organize my time, to prioritize tasks, to organize a research and a professional activity. During these three years we have always worked together in a frank, direct and effective way. They gave me an autonomy sufficient to develop my ideas and a personal imprint in the research, but, at the same time, they continuously adjusted my trajectory in order to achieve good results effectively. They paid attention and they carefully weighted my proposals and my observations but they firmly affirmed their viewpoint and took the responsibility to take the final decisions, whenever needed. For all the above reasons, they will represent a model of "a good boss" that will be a reference point in my future career.

I would like to express my gratitude to my mother and my father, who have always supported me, always believed in me, and loved me, from before the beginning of this adventure. My gratitude goes in particular to my mother, who accepted my departure far from her, knowing that my happiness and my fulfillment come first. She also became quite skilled in using all sort of telecommunication tools in order to constantly keep in contact, which constitutes one of the most relevant outcome of my thesis in the telecommunication field!

Finally, I would like to thank all my friends, thanks to whom I never felt alone, neither in Paris, nor in Sicily.

Bibliography

[1] "Cisco Visual Networking Index: Forecast and Methodology, 2014-2019," Cisco, Tech. Rep., 2015.

[2] "Cisco Visual Networking Index : Forecast and Methodology , 2013 – 2018," Tech. Rep., 2014.

[3] G. Xylomenos, C. N. Ververidis, V. a. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2013.

[4] Y. Jin, Y. Wen, and C. Westphal, "Towards Joint Resource Allocation and Routing to Optimize Video Distribution over Future Internet," in *IFIP Networking*, 2015.

[5] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *IEEE INFOCOM*, 2014.

[6] I. Sodagar and A. Vetro, "Industry and Standards The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[7] "Webrtc: Web browser with real-time communications." [Online]. Available: https://webrtc.org/

[8] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communication Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[9] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," in *ACM CoNEXT*, 2014.

[10] "IAB Statement on Internet Confidentiality," IETF, Tech. Rep., 2014.

[11] "ccnSim Webpage." [Online]. Available: http://www.enst.fr/~drossi/ccnSim

[12] A. Araldo, D. Rossi, and F. Martignon, "Cost-aware caching: Caching more (costly items) for less (ISPs operational expenditures)," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 27, no. 5, pp. 1316–1330, 2016.

[13] V. Catania, A. Araldo, and D. Patti, "Parameter Space Representation of Pareto Front to Explore Hardware-Software Dependencies," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 4, pp. 1–25, 2015.

[14] A. Araldo, G. Dan, and D. Rossi, "Stochastic Dynamic Cache Partitioning for Encrypted Content Delivery," in *ITC*, 2016.

[15] A. Araldo, F. Martignon, and D. Rossi, "Representation Selection Problem : Optimizing Video Delivery through Caching," in *IFIP Networking*, 2016.

[16] A. Araldo, D. Rossi, and F. Martignon, "Design and evaluation of cost-aware information centric routers," in *ACM SIGCOMM ICN*, 2014.

[17] A. Araldo, M. Mangili, F. Martignon, and D. Rossi, "Cost-aware caching: optimizing cache provisioning and object placement in ICN," in *IEEE Globecom*, 2014.

[18] A. Araldo and D. Rossi, "A per-application account of bufferbloat: Causes and impact on users," in *TRAC workshop*, 2014.

[19] A. G. Araldo and D. Rossi, "Dissecting Bufferbloat: Measurement and Per-Application Breakdown of Queueing Delay," in *CoNEXT Student Workhop*, 2013.

[20] D. Buckingham and R. Willett, *Digital Generations: Children, Young People and New Media.* Routledge, 2013.

[21] K. Orton-Johnson and N. Prior, Eds., *Digital Society: Critical Perspectives.* Palgrave Macmillan, 2013.

[22] R. Davis, *Web of Politics: The Internet's Impact on the American Political System*, 1st ed. Oxford University Press, 1999.

[23] P. Wallace, *The Psychology of the Internet*, 2nd ed. Cambridge University Press, 2016.

[24] K. Marx and F. Engels, *The Communist Manifesto*, 1848.

[25] S. Shakkottai and R. Srikant, "Economics of Network Pricing With Multiple ISPs," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1233–1245, 2006.

[26] A. Lodhi, N. Larson, A. Dhamdhere, C. Dovrolis, and R. Teixeira, "Using PeeringDB to Understand the Internet Peering Ecosystem," *ACM SIGCOMM CCR*, vol. 44, no. 2, pp. 20–27, 2014.

[27] P. Maillé, G. Simon, and B. Tuffin, "Impact of Revenue-Driven CDN on the Competition among Network Operators," in *IEEE CNSM*, 2015.

[28] E. Altman, P. Bernhard, S. Caron, G. Kesidis, J. Rojas-Mora, and S. Wong, "A model of network neutrality with usage-based prices," *Telecommunication Systems*, vol. 52, no. 2, pp. 601–609, 2013.

[29] "Moving Toward Usage-Based Pricing," Cisco, Tech. Rep., 2012.

[30] M. Geist, "Canada's Usage Based Billing Controversy: How to Address the Wholesale and Retail Issues," University of Ottawa, Tech. Rep., 2013.

[31] N. Hull, *The Guide to Internet Marketing.* Nathan Hull, 2008.

[32] P. Maillé, D. Châtaigneraie, and C. S. Cedex, "How Do Content Delivery Networks Affect the Economy of the Internet and the Network Neutrality Debate ?" Tech. Rep., 2014.

[33] M. Palacin, M. Oliver, J. Infante, S. Oechsner, and A. Bikfalvi, "The Impact of Content Delivery Networks on the Internet Ecosystem," *Journal of Information Policy*, vol. 3, pp. 304–330, 2013.

[34] A. Gravey, F. Guillemin, and S. Moteau, "Last Mile Caching of Video Content by an ISP," in *European Teletraffic Seminar*, 2013.

[35] *The Little Data Book on Information and Communication Technology.* The World Bank, 2015.

[36] A. Luotonen and K. Altis, "World-Wide Web proxies," *Computer Networks and ISDN Systems*, vol. 27, no. 2, pp. 147–154, 1994.

[37] V. Pacifici, "Resource Allocation in Operator-owned Content Delivery Systems," Ph.D. dissertation, KTH Royal Institute of Technology, Sweden, 2016.

[38] L. Saino, "On the Design of Efficient Caching Systems," Ph.D. dissertation, 2015.

[39] M. Mangili, "Efficient in-network content distribution : wireless resource sharing, network planning, and security," Ph.D. dissertation, Université Paris-Saclay and Politecnico di Milano, 2016.

[40] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *IEEE INFOCOM, Nomen Workshop*, 2012.

[41] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and Evaluation of the Optimal Cache Allocation for Content-Centric Networking," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 95–107, 2016.

[42] "Web Proxy Auto Discovery Protocol," 2011. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/security/web_security/connector/connector2972A/WPADAP.html

[43] A. Myers, P. Dinda, and H. Zhang, "Performance characteristics of mirror servers on the Internet," in *IEEE INFOCOM*, vol. 1, 1999.

[44] M. Chatel, *Classical versus Transparent IP Proxies.* IETF RFC 1919, 1996.

[45] N. Golrezaei, a. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, 2013.

[46] J. Zhang, "Tutorial on Small Cell/HetNet Deployment," in *IEEE Globecom Industry Forum*, 2012.

[47] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[48] F. Le, M. Srivatsa, and A. K. Iyengar, "Byte caching in wireless networks," in *IEEE ICDCS*, 2012.

[49] A. Anand, V. Sekar, and A. Akella, "SmartRE : An Architecture for Coordinated Network-wide Redundancy Elimination," *ACM SIGCOMM CCR*, vol. 39, no. 4, pp. 87–98, 2009.

[50] Y. Cui, C. Liao, I. Stojmenovic, D. Li, and H. Wang, "Cooperative Redundancy Elimination in Data Center Networks with Wireless Cards at Routers," in *International Conference on Distributed Computing Systems Workshops*, 2012.

[51] L. Wang, W. Wong, and J. Kangasharju, *In-Network Caching vs. Redundancy Elimination.* arXiv:1311.7421, 2013.

[52] N. T. Spring, D. Wetherall, and C. Science, "A Protocol-Independent Technique for Eliminating Redundant Network Traffic," *ACM SIGCOMM CCR*, vol. 30, no. 4, pp. 87–95, 2000.

[53] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

[54] A.-m. K. Pathan and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks," University of Melbourne, Tech. Rep., 2007.

[55] K. Cho, H. Jung, M. Lee, D. Ko, T. T. Kwon, and Y. Choi, "How Can an ISP Merge with a CDN?" *IEEE Communications Magazine*, vol. 49, no. 10, pp. 156–162, 2011.

[56] N. Kamiyama, T. Mori, R. Kawahara, S. Harada, and H. Hasegawa, "ISP-Operated CDN," in *IEEE INFOCOM Workshops*, 2009.

[57] D. Lee and J. Park, "ISP vs . ISP+CDN: Can ISPs in Duopoly Profit by Introducing CDN Services?" *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 2, pp. 46–48, 2012.

[58] T. Lin, Y. Xu, G. Zhang, Y. Xin, Y. Li, and S. Ci, "R-iCDN : an Approach Supporting Flexible Content Routing for ISP-operated CDN," in *ACM MobyArch*, 2014.

[59] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou, "More control over network resources: An ISP caching perspective," in *CNSM*, 2013.

[60] G. Pallis and A. Vakali, "Insight and Perspective for Content Delivery Networks," *Communications of the ACM*, vol. 49, no. 1, pp. 101–106, 2006.

[61] J. S. Otto, M. a. Sánchez, J. P. Rula, and F. E. F. E. Bustamante, "Content delivery and the natural evolution of DNS: remote dns trends, performance issues and alternative solutions," in *ACM IMC*, 2012.

[62] M. K. Mukerjee and D. Naylor, "Practical, Real-time Centralized Control for CDN-based Live Video Delivery," in *ACM SIGCOMM*, 2015, pp. 311–324.

[63] S. Hollenbeck, S. Veeramachaneni, and S. Yalamanchilli, *VeriSign Registry Registrar Protocol (RRP) Version 2.0.0.* IETF RFC 2832, 2003.

[64] "The Essential CDN Guide." [Online]. Available: https://www.incapsula.com/cdn-guide/what-is-cdn-how-it-works.html

[65] B. M. Maggs and R. K. Sitaraman, "Algorithmic Nuggets in Content Delivery," *ACM SIGCOMM CCR*, vol. 45, no. 3, pp. 52–66, 2015.

[66] B. Ahlgren and C. Dannewitz, "A survey of information-centric networking," *IEEE Communications Magazine*, no. July, pp. 26–36, 2012.

[67] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, and M. Varvello, "From content delivery today to information centric networking," *Computer Networks*, vol. 57, no. 16, pp. 3116–3127, 2013.

[68] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT*, 2009.

[69] D. Perino, M. Varvello, B. Labs, L. Linguaglossa, R. Laufer, and R. Boislaigue, "Caesar: A Content Router for High-Speed Forwarding on Content Names," in *ACM/IEEE ANCS*, 2014.

[70] W. So, A. Narayanan, D. Oran, and M. Stapp, "Named Data Networking on a Router: Forwarding at 20Gbps and Beyond Categories and Subject Descriptors," in *ACM SIGCOMM*, 2013.

[71] P. Goel, E. Holmberg, M. Konezny, R. Ayyagari, and D. Sillman, "CCNx Packet Processing on PARC Router Platform," in *ACM SIGCOMM ICN demo*, 2015.

[72] R. B. Mansilha, L. Saino, M. P. Barcellos, M. Gallo, E. Leonardi, D. Perino, and D. Rossi, "Hierarchical Content Stores in High-Speed ICN Routers," in *ACM SIGCOMM ICN*, 2015.

[73] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM CCR*, vol. 37, no. 4, p. 181, oct 2007.

[74] "Named Data Networking Project." [Online]. Available: http://www.named-data.net

[75] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed," *Computer Networks*, vol. 57, no. 16, pp. 3207–3221, nov 2013.

[76] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "QoE-Driven Cache Management for HTTP Adaptive Bit Rate Streaming Over Wireless Networks," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1431–1445, oct 2013.

[77] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–582, 2000.

[78] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal Content Placement for a Large-Scale VoD System," in *ACM CoNEXT*, 2010.

[79] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the Complexity of Optimal Routing and Content Caching in Heterogeneous Networks," in *IEEE INFOCOM*, 2015.

[80] G. Gursun, M. Crovella, and I. Matta, "Describing and Forecasting Video Access Patterns," in *IEEE INFOCOM*, 2011.

[81] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical Web caches," in *IEEE IPCCC*, 2004.

[82] G. Rossini and D. Rossi, "Coupling Caching and Forwarding : Benefits, Analysis, and Implementation," in *ACM SIGCOMM ICN*, 2014.

[83] G. Rossini, "Analysis of forwarding strategies for Host and Content Centric Networking," Ph.D. dissertation, Telecom Paristech, 2014.

[84] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128–3141, 2013.

[85] M. Arlitt, R. Friedrich, and T. Jin, "Performance evaluation of Web proxy cache replacement policies," *Performance Evaluation*, vol. 39, no. 1-4, pp. 149–164, 2000.

[86] S. Podlipnig and L. Böszörmenyi, "A survey of Web cache replacement strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, 2003.

[87] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *ACM CoNEXT, ReArch Workshop*, 2010.

[88] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *IEEE INFOCOM*, 2014.

[89] D. Lee, J. Choi, and S. H. Noh, "LRFU (Least Recently / Frequently Used) Replacement 1 Introduction," *IEEE Transactions on Computers*, vol. 50, no. 12, pp. 1352–1361, 1996.

[90] J. Jung, A. Berger, and H. Balakrishnan, "Modeling TTL-based Internet caches," in *IEEE INFOCOM*, 2003.

[91] M. Tortelli, D. Rossi, and E. Leonardi, "Model-Graft : Accurate , Scalable and Flexible Analysis of Cache Networks," Telecom ParisTech, Tech. Rep., 2016.

[92] H. S. Stone, J. Turek, and J. L. Wolf, "Optimal partitioning of cache memory," *IEEE Transactions on Computers*, vol. 41, no. 9, pp. 1054–1068, 1992.

[93] I. Megory-Cohen and G. Ela, *Dynamic Cache Partitioning by Modified Steepest Descent.* U.S. Patent No. 5,357,623, 1994.

[94] G. E. Suh, L. Rudolph, and S. Devadas, "Dynamic partitioning of shared cache memory," *Journal of Supercomputing*, vol. 28, no. 1, pp. 7–26, 2004.

[95] M. K. Qureshi and Y. N. Patt, "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches," in *IEEE/ACM MICRO*, 2006.

[96] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Evaluating per-application storage management in content-centric networks," *Computer Communications*, vol. 36, no. 7, pp. 750–757, apr 2013.

[97] S. Hoteit, M. E. Chamie, D. Saucez, and S. Secci, "On Fair Network Cache Allocation to Content Providers," *Computer Networks*, 2016.

[98] Z. Li and G. Simon, "Cooperative Caching in a Content Centric Network for Video Stream Delivery," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 445–473, 2014.

[99] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites," in *World Wide Web Conference*, 2002.

[100] A. Singla, B. Chandrasekaran, P. B. Godfrey, and B. Maggs, "The Internet at the Speed of Light," in *ACM HotNets*, 2014.

[101] P. K. Agyapong and M. Sirbu, "Economic Incentives in Information- Centric Networking : Implications for Protocol Design and Public Policy," *IEEE Communications Magazine*, 2012.

[102] M. Mangili, F. Martignon, and S. Paraboschi, "A cache-aware mechanism to enforce confidentiality, trackability and access policy evolution in Content-Centric Networks," *Computer Networks*, vol. 76, pp. 126–145, 2015.

[103] B. S. Davie and F. Le Faucher, *System and method for tracking request accountability in multiple content delivery network environments.* U.S. Patent Application No. 13/349,777, 2012.

[104] N. Zhang, T. Levä, and H. Hämmäinen, "Value networks and two-sided markets of Internet content delivery," *Telecommunications Policy*, vol. 38, no. 5-6, pp. 460–472, may 2014.

[105] A. Dhamdhere and C. Dovrolis, "The Internet is Flat: Modeling the Transition from a Transit Hierarchy to a Peering Mesh," in *ACM CoNEXT*, 2010.

[106] E. Limer, "This Box Can Hold an Entire Netflix," 2013. [Online]. Available: gizmodo.com/this-box-can-hold-an-entire-netflix-1592590450

[107] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the Air : Exploiting Content Caching and Delivery Techniques for 5G Systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[108] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, and A. Ghodsi, "Less Pain , Most of the Gain: Incrementally Deployable ICN," in *ACM SIGCOMM*, 2013.

[109] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "Less for More" in Information-centric Networks," in *IFIP Networking*, 2012.

[110] G. Rossini and D. Rossi, "Evaluating CCN multi-path interest forwarding strategies," *Computer Communications*, vol. 36, no. 7, pp. 771–778, apr 2013.

[111] C. Barakat, A. Kalla, D. Saucez, and T. Turletti, "Minimizing Bandwidth on Peering Links with Deflection in Named Data Networking," in *ICCIT*, 2013.

[112] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal Cache Allocation for Content-Centric Networking," *ICNP*, 2013.

[113] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," *Computer Communications*, vol. 25, no. 4, pp. 376–383, 2002.

[114] S. Zaman and D. Grosu, "A Distributed Algorithm for the Replica Placement Problem," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 22, no. 9, pp. 1455 – 1468, 2011.

[115] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed Selfish Replication," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 17, no. 12, pp. 1401–1413, dec 2006.

[116] M. Badov, A. Seetharam, and J. Kurose, "Congestion-Aware Caching and Search in Information-Centric Networks," in *ACM SIGCOMM ICN*, 2014.

[117] G. Carofiglio, L. Mekinda, and L. Muscariello, "LAC: Introducing Latency-Aware Caching in Information-Centric Networks," in *IEEE LCN*, 2015.

[118] M. Mangili, F. Martignon, and A. Capone, "A Comparative Study of Content-Centric and Content-Distribution Networks: Performance and Bounds," in *IEEE Globecom*, 2013.

[119] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," in *Usenix symposium on internet technologies and systems*, 1997.

[120] P. Marchetta, J. Llorca, A. M. Tulino, and P. Antonio, "MC3: A Cloud Caching Strategy for Next Generation Virtual Content Distribution Networks," in *IFIP Networking*, 2016.

[121] N. E. Young, "The K-Server Dual and Loose Competitiveness for Paging," *Algorithmica*, vol. 11, no. 6, pp. 525–541, 1994.

[122] C. Li and A. L. Cox, "GD-Wheel: A Cost-Aware Replacement Policy for Key-Value Stores," in *EuroSys*, 2015.

[123] V. Pacifici and G. Dan, "Content-peering Dynamics of Autonomous Caches in a Content-centric Network," in *IEEE INFOCOM*, 2013.

[124] T. Pham, S. Fdida, and P. Antoniadis, "Pricing in Information-Centric Network Interconnection," in *IEEE IFIP Networking*, 2013.

[125] F. Kocac, G. Kesidis, T. Pham, and S. Fdida, "The effect of caching on a model of content and access provider revenues in information-centric networks," in *IEEE SocialCom*, 2013.

[126] G. Dan, "Cache-to-Cache: Could ISPs Cooperate to Decrease Peer-to-peer Content Distribution Costs?" *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 22, no. 9, pp. 1469–1482, 2011.

[127] I. Castro, S. Member, R. Stanojevic, and S. Gorinsky, "Using Tuangou to Reduce IP Transit Costs," *IEEE/ACM Transactions on Networking*, vol. 22, no. 5, pp. 1415–1428, 2014.

[128] D. DiPalantino and R. Johari, "Traffic Engineering vs. Content Distribution: A Game Theoretic Perspective," in *IEEE INFOCOM*, 2009.

[129] K. Katsalis, V. Sourlas, T. Papapioannou, T. Korakis, and L. Tassiulas, "Content Placement in Heterogeneous End-to-End Virtual Networks," in *ACM Symposium On Applied Computing (SAC)*, 2015.

[130] S. Yun, D. Kim, X. Lu, and L. Qiu, "Optimized Layered Integrated Video Encoding," in *IEEE INFOCOM*, 2015.

[131] L. Toni, R. Aparicio-pardo, G. Simon, A. Blanc, and P. Frossard, "Optimal Set of Video Representations in Adaptive Streaming Categories and Subject Descriptors," in *ACM MMSys*, 2014.

[132] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an ISP network," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, pp. 239–250, 2009.

[133] J. Roberts and N. Sbihi, "Exploring the Memory-Bandwidth Tradeoff in an Information-Centric Network," in *IEEE ITC*, 2013.

[134] S.-E. Elayoubi and J. Roberts, "Performance and Cost Effectiveness of Caching in Mobile Access Networks," in *ACM SIGCOMM ICN*, 2015.

[135] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in HTTP Adaptive Streaming : Friend or Foe ?" in *ACM NOSSDAV*, 2014.

[136] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288–311, apr 2012.

[137] G. Cofano, L. De Cicco, and S. Mascolo, "A Control Architecture for Massive Adaptive Video Streaming Delivery," in *ACM VideoNext*, 2014.

[138] Y. Sun, S. K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, and S. Uhlig, "Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand Workloads," in *ACM CoNEXT*, 2014.

[139] A. Begen, K. Streeter, I. Bouazizi, and F. Denoual, "MPEG DASH Requirements for a webpush Protocol," Tech. Rep., 2014.

[140] "Open Caching: Problem Statement and Guiding Principles," Streaming Video Alliance, Tech. Rep., 2015.

[141] H. Xiong, X. Zhang, W. Zhu, and D. Yao, "CloudSeal: End-to-end content protection in cloud-based storage and delivery services," in *Security and Privacy in Communication Networks.* Springer, 2012, pp. 491–500.

[142] N. Fotiou, G. F. Marias, and G. C. Polyzos, "Access control enforcement delegation for information-centric networking architectures," in *ICN workshop on Information-centric networking*, 2012, p. 85.

[143] M. S. Manasse, L. A. McGeoch, and D. D. Sleator, "Competitive algorithms for server problems," *Journal of Algorithms*, vol. 11, no. 2, pp. 208–230, 1990.

[144] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon, "Transcoding live adaptive video streams at a massive scale in the cloud," in *ACM MMSys*, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2713168.2713177

[145] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," *ACM MMSys*, 2011. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1943552.1943573

[146] Y. Sánchez, C. Hellge, and T. Schierl, "Scalable Video Coding based DASH for efficient usage of network resources," in *W3C Web and TV Workshop*, 2011.

[147] S. E. Elayoubi and J. Roberts, "Performance Evaluation of Video Transcoding and Caching Solutions in Mobile Networks," in *ITC*, 2015.

[148] D. K. Krishnappa, M. Zink, and R. K. Sitaraman, "Optimizing the video transcoding workflow in content delivery networks," in *ACM MMSys*, 2015, pp. 37–48. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84942520755{&}partnerID=tZOtx3y1

[149] "Per-Title Encode Optimization," 2015. [Online]. Available: http://techblog.netflix.com/2015/12/per-title-encode-optimization.html

[150] J. W. Kleinrouweler, S. Cabrero, R. Van Der Mei, and P. Cesar, "Modeling Stability and Bitrate of Network-Assisted HTTP Adaptive Streaming Players," in *ITC*, 2015.

[151] A. Finamore, "YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience," in *ACM SIGCOMM IMC*, 2011.

[152] E. Thomas, M. O. V. Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, "Sand Dash," Tech. Rep.

[153] C. a. Wood and E. Uzun, "Flexible End-to-End Content Security in CCN," in *CCNC*, 2014.

[154] X. Zhang, K. Chang, H. Xiong, Y. Wen, G. Shi, and G. Wang, “Towards name-based trust and security for content-centric network,” in *ICNP*, 2011.

[155] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. M. Maggs, J. Rake, S. Uhlig, and R. Weber, “Pushing CDN-ISP Collaboration to the Limit,” *ACM SIGCOMM CCR*, vol. 43, no. 3, pp. 35–44, 2013.

[156] L. Muscariello, G. Carofiglio, and M. Gallo, “Bandwidth and storage sharing performance in information centric networking,” *ACM SIGCOMM, ICN Workshop*, 2011.

[157] M. Motiwala, A. Dhamdhere, N. Feamster, A. Lakhina, G. Tech, and C. Guavus, “Towards a Cost Model for Network Traffic,” *ACM SIGCOMM CCR*, vol. 42, no. 1, pp. 54–60, 2012.

[158] R. Stanojevic, N. Laoutaris, and P. Rodriguez, “On Economic Heavy Hitters : Shapley value analysis of 95th-percentile pricing,” in *ACM SIGCOMM IMC*, 2010.

[159] L. Gyarmati, R. Stanojevic, M. Sirivianos, and N. Laoutaris, “Sharing the cost of backbone networks,” in *ACM SIGCOMM IMC*, 2012, p. 509.

[160] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, third edit ed. MIT Press, 2009, no. 2. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2560149{&}tool=pmcentrez{&}rendertype=abstract

[161] “GNU Octave Web Page.” [Online]. Available: https://www.gnu.org/software/octave/

[162] E. G. J. Coffman and P. J. Denning, “Probability Models of Computer Sequencing Problems,” in *Operating Sysytems Theory.* Prentice-Hall, 1973, ch. 4, pp. 144–189.

[163] C. Fricker, P. Robert, and J. Roberts, “A versatile and accurate approximation for LRU cache performance,” in *ITC*, 2012.

[164] G. Zipf, *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, 1949.

[165] M. E. J. Newman, “Power laws, Pareto distributions and Zipf’s law,” *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

[166] L. A. Adamic, “Zipf Power Law and Pareto: A Ranking Tutorial,” 2000. [Online]. Available: http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html

[167] V. Valancius, C. Lumezanu, N. Feamster, R. Johari, V. V. Vazirani, and G. Tech, “How Many Tiers ? Pricing in the Internet Transit Market,” in *ACM SIGCOMM*, 2011.

[168] I. Psaras, W. K. Chai, G. Pavlou, and S. Member, "In-Network Cache Management and Resource Allocation for Information-Centric Networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 25, no. 11, pp. 2920–2931, 2014.

[169] E. Gelenbe, "A Unified Approach to the Evaluation of a Class of Replacement Algorithms," *IEEE Transactions on Computers*, vol. C-22, no. 6, pp. 611–618, 1973.

[170] R. Fagint and T. G. Price, "Efficient Calculation of Expected Miss Ratios in the Independent Reference Model," *SIAM Journal on Computing*, vol. 7, no. 3, pp. 288–297, 1978.

[171] B. Leonid and A. Yakov, "Some Results on Distribution-Free Analysis of Paging Algorithms," *IEEE Transactions on Computers*, vol. C, no. 7, pp. 737–745, 1976.

[172] R. Fagin, "Asymptotic Miss Ratios over Independent References," *Elsevier Journal of Computer and System Sciences*, pp. 222–250, 1977.

[173] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of CCN-caching trees," in *Lecture Notes in Computer Science*, 2011, vol. 6640, pp. 78–91.

[174] J. Garcia-Reinoso, I. Vidal, D. Diez, D. Corujo, and R. L. Aguiar, "Analysis and Enhancements to Probabilistic Caching in Content-Centric Networking," *The Computer Journal*, 2015.

[175] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate Models for General Cache Networks," in *IEEE INFOCOM*, 2010.

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

[177] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching," *ACM SIGCOMM CCR*, vol. 43, no. 5, pp. 5–12, nov 2013.

[178] M. Cha, H. Kwak, P. Rodriguez, Y.-y. Ahn, and S. Moon, "I Tube , You Tube , Everybody Tubes : Analyzing the World ' s Largest User Generated Content Video System," in *ACM SIGCOMM IMC*, 2007.

[179] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, "Optimizing cost and performance for content multihoming," *ACM SIGCOMM CCR*, vol. 42, no. 4, p. 371, sep 2012.

[180] "Sharp Website." [Online]. Available: http://www.sharpusa.com/ces-2015-recap.aspx

[181] B. Niven -Jenkins, F. Le Faucheur, and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement," IETF, Tech. Rep. September, 2012.

[182] L. D. Cicco, V. Caldaralo, V. Palmisano, S. Mascolo, and S. Member, "ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," in *IEEE Packet Video*, 2013.

[183] H. Susanto, B. Kim, and B. Liu, "User Experience Driven Multi-Layered Video Based Applications," in *IEEE ICCCN*, 2015.

[184] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–12, 1999.

[185] D. Kahneman, *Thinking, fast and slow.* Macmillan, 2011.

[186] S. Deering, "Watching the Waist of the Protocol Hourglass," in *IETF51 Plenary Talk*, 2001.

[187] D. Thaler, "Evolution of the IP Model," in *IETF73 Plenary Talk*, 2008.

[188] I. Popa, Lucian and Ghodsi, Ali and Stoica, "HTTP as the Narrow Waist of the Future Internet," in *ACM SIGCOMM HotNets Workshop*, 2010.

[189] G. Barish and K. Obraczke, "World Wide Web caching: trends and techniques," *IEEE Communications Magazine*, vol. 38, no. 5, pp. 178–184, 2000.

[190] Ü. Yüceer, "Discrete convexity: convexity for functions defined on discrete spaces," *Discrete Applied Mathematics*, vol. 119, no. 3, pp. 297–304, 2002.

[191] J. Carnicer and W. Dahmen, "Characterization of Local Strict Convexity Preserving Interpolation Methods by C1 Functions," *Journal of Approximation Theory*, vol. 77, no. 1, pp. 2–30, 1994.

[192] W. Wang and M. A. Carreira-Pepinan, *Projection onto the probability simplex : An efficient algorithm with a simple proof and an application.* arXiv:1309.1541v1, 2013.

[193] H. H. Bauschke and J. M. Borwein, "On Projection Algorithms for Solving Convex Feasibility Problems," *SIAM Review*, vol. 38, no. 3, pp. 367–426, 1996.

[194] S. M. Sefanov, *Separable Programming: Theory and Methods.* Springer, 2013.

[195] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004. [Online]. Available: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

[196] N. Z. Shor, *Nondifferentiable Optimization and Polynomial Problems.* Springer Science and Business Media, 1998.

[197] C. P. Niculescu and L.-E. Persson, *Convex Functions and Their Applications: A Contemporary Approach.* Springer Science and Business Media, 2004.

[198] D. S. Hochbaum, "Lower and Upper Bounds for the Allocation Problem and Other Nonlinear Optimization Problems," *Mathematics of Operation Research*, vol. 19, no. 2, pp. 390–409, 1994.

[199] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of Traffic Mix on Caching Performance in a Content-Centric Network," in *IEEE INFOCOM*, 2012.

[200] C. Imbrenda, L. Muscariello, and D. Rossi, "Analyzing Cacheable Traffic in ISP Access Networks for Micro CDN Applications via Content-Centric Networking," in *ACM SIGCOMM ICN*, 2014.

[201] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Joint Caching and Base Station Activation for Green Heterogeneous Cellular Networks," in *IEEE ICC*, 2015.

[202] Q. Wang, "Optimization with Discrete Simultaneous Perturbation Stochastic Approximation Using Noisy Loss Function Measurement," Ph.D. dissertation, John Hopkins University, 2013.

[203] M. Garetto, E. Leonardi, and S. Traverso, "Efficient analysis of caching strategies under dynamic content popularity," *IEEE INFOCOM*, 2015.

[204] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms," in *VideoNext Workshop*, 2014.

[205] Q. Wang and J. C. Spall, "Discrete simultaneous perturbation stochastic approximation on loss function with noisy measurements," *American Control Conference (ACC)*, pp. 4520–4525, 2011.

List of abbreviations

ABR	Adaptive Bit-Rate
AS	Achieved Saving
CDN	Content Delivery Network
CoA	Content-Aware Caching
CP	Content Provider
DNS	Domain Name System
DRM	Digital Rights Management
Gbps	Gigabits per second
HTML	HyperText Markup Language
IAB	Internet Architecture Board
IETF	Internet Engineering Task Force
ICN	Information Centric Network
ILP	Integer Linear Program
iNRR	ideal Nearest Replica Routing
IRM	Independent Reference Model
ISP	Internet Service Provider
IP	Internet Protocol
LAN	Local Area Network
LCE	Leave a Copy Everywhere
LRU	Least Recently Used
MILP	Mixed Integer Linear Program
NDN	Named Data Network
NRR	Nearest Replica Routing
PS	Potential Saving
QoE	Quality of Experience
RE	Redundancy Elimination
SPR	Shortest Path Routing
SVC	Scalable Video Coding
TB	Terabytes
VoD	Video on Demand
VoIP	Voice over IP
URL	Uniform Resource Locator

Index

access control, 37
Adaptive Bit-Rate Streaming, 16
Adaptive Streaming, 16
aggregation, 28

base memory, 27
border router, 66
byte caching, 30

cache hit, 27
cache miss, 27
Cache Partitioning, 35
catalog, 27
Che's Approximation, 67
content delivery, 14
Content Delivery Network, 30
Content Provider, 23
Cost Aware Caching, 63
cost fraction, 72

DASH, 16
data packet, 32
decision policy, 34
Digital Rights Management, 45
DNS redirection, 31

femtocells, 30
Fixed Probabilistic Caching, 34
flash crowds, 36
forwarding policy, 34

hit ratio, 27, 53, 72

ideal Nearest Replica Routing, 75
Independent Reference Model (IRM), 58
Information Centric Networks, 32
insertion policy, 34
insertion probability, 34
inter-domain link, 24, 25
interest packet, 32
Internet Service Provider, 24

Layered Video Coding, 43
Least Recently Used, 34
Leave a Copy Everywhere, 34
Local ISP, 25

metacaching policy, 34
metrocells, 30
microcells, 30
mirror servers, 29

Nearest Replica Routing, 75
network controller, 33
Network-Assisted Adaptive Streaming, 45

object, 27
on-the-fly transcoding, 44
origin servers, 27

pay-per-view payment, 26
peering agreement, 25
picocells, 30
Potential Saving, 70
Prob, 34
proxy cache, 28

QImpr, 108

Redundancy Elimination, 30
rendez-vous system, 32
replacement policy, 34

Scalable Video Coding, 43

Server and Network Assisted DASH, 45
Shortest Path Routing, 75
small cells, 30
snapshot approach, 33
Stochastic Dynamic Cache Partitioning, 119
surrogate servers, 30

targeted advertising, 26
Time-to-Live based cache, 35
trackability, 37
transcoding, 44
transit agreement, 25
Transit ISP, 25
transparent proxy cache, 29

URL rewriting, 31
usage based billing, 25

views, 26

Web Proxy Auto Discovery Protocol, 28
WebRTC, 16

Zipf distribution, 58

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Conception et Évaluation de Systèmes de Caching de Réseau pour Améliorer la Distribution des Contenus sur Internet

Mots clés : Caching de Réseau ; Distribution des Contenus ; Optimisation de Réseau

Résumé : Le caching de réseau peut aider à gérer l'explosion du trafic sur Internet et à satisfaire la Qualité d'Expérience (QoE) croissante demandée par les usagers. Néanmoins, les techniques proposées jusqu'à présent par la littérature scientifique n'arrivent pas à exploiter tous les avantages potentiels. Les travaux de recherche précédents cherchent à optimiser le hit ratio ou d'autres métriques de réseau, tandis que les opérateurs de réseau (ISPs) sont plus intéressés à des métriques plus concrètes, par exemple le coût et la qualité d'expérience (QoE). Pour cela, nous visons directement l'optimisation des métriques concrètes et montrons que, ce faisant, on obtient des meilleures performances.
Plus en détail, d'abord nous proposons des nouvelles techniques de caching pour réduire le coût pour les ISPs en préférant stocker les objets qui sont les plus chères à repérer. Nous montrons qu'un compromis existe entre la maximisation classique du hit ratio et la réduction du coût.
Ensuite, nous étudions la distribution vidéo, comme elle est la plus sensible à la QoE et constitue la plus part du trafic Internet. Les techniques de caching classiques ignorent ses caractéristiques particulières, par exemple le fait qu'une vidéo est représentée par différentes représentations, encodées en différents bit-rates et résolutions. Nous introduisons des techniques qui prennent en compte cela.
Enfin, nous remarquons que les techniques courantes assument la connaissance parfaite des objets qui traversent le réseau. Toutefois, la plupart du trafic est chiffrée et du coup toute technique de caching ne peut pas fonctionner. Nous proposons un mécanisme qui permet aux ISPs de faire du caching, bien qu'ils ne puissent observer les objets envoyés.

Title : Design and Evaluation of Enhanced Network Caching Systems to Improve Content Delivery in the Internet

Keywords : Network Caching ; Content Delivery ; Network Optimization

Abstract : Network caching can help cope with today Internet traffic explosion and sustain the demand for an increasing user Quality of Experience. Nonetheless, the techniques proposed in the literature do not exploit all the potential benefits. Indeed, they usually aim to optimize hit ratio or other network-centric metrics, e.g. path length, latency, etc., while network operators are more focused on more more practical metrics, like cost and quality of experience. We devise caching techniques that directly target the latter objectives and show that this allows to gain better performance.
More specifically, we first propose novel strategies that reduce the Internet Service Provider (ISP) operational cost, by preferentially caching the objects whose cost of retrieval is the largest.
We then focus on video delivery, since it is the most sensitive to QoE and represents most of the Internet traffic. Classic techniques ignore that each video is represented by different representations, encoded at different bit-rates and resolutions. We devise techniques that take this into account.
Finally, we point out that the techniques presented in the literature assume the perfect knowledge of the objects that are crossing the network. Nonetheless, most of the traffic today is encrypted and thus caching techniques are inapplicable. To overcome this limit, We propose a mechanism which allows the ISPs to cache, even without knowing the objects being served.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France