Optimisation d'un réseau de distribution de contenus géré par un opérateur réseau

Zhe Li

N° d'ordre : 2013telb0267

Sous le sceau de l' Université européenne de Bretagne

Télécom Bretagne

En habilitation conjointe avec l'Université de Rennes 1

Co-tutelle avec

Ecole Doctorale – MATISSE

Optimisation d'un Réseau de Distribution de Contenus Géré par un Opérateur Réseau

---

Thèse de Doctorat

Mention : Informatique

Présentée par **ZHE LI**

Département : Informatique

Laboratoire : IRISA

Directeur de thèse : ANTOINE BEUGNARD

Soutenue le jour 25/01/2013

Jury :

M. JAMES ROBERTS, Directeur de recherche, INRIA (Rapporteur)
M. OLIVIER FESTOR, Directeur de recherche, LORIA (Rapporteur)
M. ANTOINE BEUGNARD, Professeur, Télécom Bretagne (Directeur de thèse)
M. GABRIEL ANTONIU, Professeur, INRIA (Examinateur)
M. DIEGO PERINO, Chercheur, Alactel-Lucent (Examinateur)
M. GWENDAL SIMON, Maitre de conférence,Télécom Bretagne (Encadrant)

Acknowledgement

This thesis arose out of years of research work that has been done since the graduation internship of my master at Telecom Bretagne in Brest in France. During this period, I have worked with a great number of people who contribute in various ways to my research and the making of the thesis. It is a pleasure to convey my most profound gratitude to them all.

First of all, I would like to give my great thanks to my supervisor, Gwendal Simon, who offered me the great opportunity to work in the Computer Science Department of Telecom Bretagne. Without his help, this thesis would not have been possible. His ambition and energy to be excellent in the research works always encourage me to overcome the difficulties. I will never forget the days of the interesting discussions and the deadline rush with him.

I am indebted to my supervisor Annie Gravey for her warmly reception and insightful discussions and instructions on the research works.

I gratefully acknowledge Professor Antoine Beugnard, the director of this thesis, for his advice, supervision and constant support.

I also want to show my gratitude to Professor Di Yuan of Linköping University who gave me a lot of instructions on linear programming.

I would also like to thank the members of my committee for accepting the invitation and spent their precious time helping me to improve this thesis:

Many thanks to all members in our research group: Jimmy Leblet, Yiping Chen, Yaning Liu, Jiayi Liu, Wei You, Xu Zhang, Fen Zhou, and Eliya Buyukkaya. The group has been a source of friendships as well as good advice and collaboration. Except for that, we are more like a family to care for each other, love and help each other. Furthermore, thanks to Bertrand Mathieu and Professor Fabien Dagnat for the interesting discussion. I would also like to thank Adrien Cormier, Christophe Bramoullé and Antoine Fourmy who worked with me for their student projects.

My sincere appreciation to all members of Computer Engineering Department of Telecom Bretagne, especially Armelle Lannuzel who perfectly take care of my demands on the administrative procedures and the daily life, so that I have been able to concentrate on my research.

Nothing I can achieve without my family. Thanks to my girl friend Shasha Shi for her accompany and support during these years. I am proud of my parents, who tried all their best in improving my living and studying conditions, gave me their full confidence and encouraged me during the most difficult times. Additionally, I would like to thank my grand mother who took care of me a lot during my teenage.

Finally, I would like to thank everybody who helped me for the successful realization of thesis.

Abstract

The exploding HD video streaming traffic calls for deploying content servers deeper inside network operators infrastructures. Telco-CDN are new content distribution services that are managed by Internet Service Providers (ISP). Since the network operator controls both the infrastructure and the content delivery overlay, it is in position to engineer Telco-CDN so that networking resources are optimally utilized. In this thesis, we focus on the optimal resource placement in Telco-CDN.

We first investigated the placement of application components in Telco-CDN. Popular services like Facebook or Twitter, with a size in the order of hundreds of Terabytes, cannot be fully replicated on a single data-center. Instead, the idea is to partition the service into smaller components and to locate the components on distinct sites. It is the same and unique method for Telco-CDN operators. We addressed this k-Component Multi-Site Placement Problem from an optimization standpoint. We developed linear programming models, designed approximation and heuristic algorithms to minimize the overall service delivery cost.

Thereafter, we extend our works to address the problem of optimal video placement for Telco-CDN. We modeled this problem as a k-Product Capacitated Facility Location Problem, which takes into account network conditions and users' preferences. We designed a genetic algorithm in order to obtain near-optimal performances of such "push" approach, then we implemented it on the MapReduce framework in order to deal with very large data sets. The evaluation signifies that our optimal placement keeps align with cooperative LRU caching in term of storage efficiency although its impact on network infrastructure is less severe.

We then explore the caching decision problem in the context of Information Centric Network (ICN), which could be a revolutionary design of Telco-CDN. In ICN, routers are endowed with caching capabilities. So far, only a basic Least Recently Used (LRU) policy implemented on every router has been proposed. Our first contribution is the proposition of a cooperative caching protocol, which has been designed for the treatment of large video streams with on-demand access. We integrated our new protocol into the main router software (CCNx) and developed a platform that automatically deploys our augmented CCNx implementation on real machines. Experiments show that our cooperative caching significantly reduces the inter-domain traffic for an ISP with acceptable overhead.

Finally, we aim at better understanding the behavior of caching policies other than LRU. We built an analytical model that approximates the performance of a set of policies ranging from LRU to Least Frequently Used (LFU) in any type of network topologies. We also designed a multi-policy in-network caching, where every router implements its own caching policy according to its location in the network. Compared to the single LRU policy, the multi-caching strategy considerably increases the hit-ratio of the in-network caching system in the context of Video-on-Demand application.

All in one, this thesis explores different aspects related to the resource placement in Telco-CDN. The aim is to explore optimal and near-optimal performances of various approaches.

Keywords: Facility Location Problem, MapReduce, Caching Policies, CDN, ICN.

Contents

Acknowledgements **i**

Abstract **iii**

Contents **vii**

List of Figures **x**

List of Tables **xi**

Résumé **xiii**

1 Introduction **1**
1.1 Telco-CDN Architecture 2
1.1.1 Rationales behind Telco-CDNs 3
1.1.2 Sketching Telco-CDN architecture 3
1.2 Contributions 4
1.3 Organization of Dissertation 6

2 Resource Placement Overview: Practice and Theory **9**
2.1 Introduction 9
2.2 Practical Aspects 9
2.2.1 P2P Streaming Technology 10
2.2.2 Content Delivery Network 12
2.2.3 Information Centric Network 15
2.3 Theoretical Related Work 17
2.3.1 Facility Location Problem 17
2.3.2 Caching Policies 19
2.4 Summary 22

3 Application Components Allocation in Telco-CDN **23**
3.1 Introduction 23
3.2 Problem Analysis 24
3.2.1 NP-Completeness 25
3.2.2 Integer Programming 26
3.3 Approximation Algorithm 28

3.4 Heuristic Algorithms . . . 31
3.4.1 A Heuristic using Domatic Partition . . . 31
3.4.2 An Intuitive and Localized Heuristic . . . 33
3.4.3 A Heuristic Addressing Fairness . . . 35
3.5 Simulations . . . 36
3.5.1 Small Instances . . . 37
3.5.2 Large Instances . . . 39
3.5.3 Variance . . . 40
3.5.4 Fairness . . . 41
3.6 Summary and Conclusion . . . 42

4 Optimal Video Placement in Telco-CDN 45
4.1 Introduction . . . 45
4.1.1 Background on Genetic Algorithms . . . 45
4.1.2 MapReduce Overview . . . 46
4.1.3 Our Contributions . . . 46
4.2 Problem Formulation . . . 47
4.3 Algorithm Description . . . 49
4.3.1 Genetic Algorithm for Content Placement . . . 50
4.3.2 Parallelizing GA by MapReduce . . . 54
4.4 Evaluation . . . 58
4.4.1 Traces from a national VoD Service . . . 58
4.4.2 Network topology and management . . . 59
4.4.3 Evaluated Strategies . . . 60
4.4.4 Results . . . 61
4.5 Summary and Conclusion . . . 64

5 Cooperative Caching for CCN 67
5.1 Introduction . . . 67
5.1.1 Our Proposal: Cooperative In-Network Caching . . . 68
5.1.2 Our Contributions: Algorithms and CCN Protocol . . . 69
5.2 System Model . . . 70
5.2.1 Network Model . . . 70
5.2.2 Practical Details . . . 71
5.3 Initialization Stage . . . 72
5.3.1 Distributed Algorithm . . . 73
5.3.2 Correctness and Analysis . . . 74
5.3.3 Implementation in CCN . . . 74
5.4 Augmented CCN Protocol . . . 75
5.4.1 New Tables in CCN . . . 75
5.4.2 Segment Distribution in the Cooperative Cache . . . 75
5.4.3 CCS Consistency . . . 76
5.4.4 Implementation into CCNx . . . 78
5.5 Analysis . . . 79
5.6 Simulations . . . 81
5.6.1 Platform Setup . . . 81

5.6.2 Network and Service Configuration . . . 81
5.6.3 VoD Service . . . 82
5.6.4 Catch-up TV . . . 84
5.6.5 Overhead . . . 86
5.7 Design of CCNxProSimu . . . 86
5.7.1 Pre-configuration . . . 86
5.7.2 Scenario Description . . . 87
5.7.3 Input and Output of CCNxProSimu . . . 89
5.8 Summary and Conclusion . . . 90

6 Stochastic Model for In-Network Caching Policies 93
6.1 Introduction . . . 93
6.2 Simulation Based Study of LRFU . . . 94
6.2.1 Introduction to LRFU Policy . . . 94
6.2.2 Single Cache Performance . . . 94
6.2.3 Multi-Cache Performance . . . 95
6.3 Generalized Multi-cache Approximation . . . 97
6.3.1 Derivation of the analytical approximation . . . 97
6.3.2 Validation of the approximate model . . . 100
6.4 Optimizing LRFU in Multi-cache Network . . . 101
6.4.1 Optimizing cache policies on the Ebone example . . . 102
6.4.2 Relaxing the Independence assumption . . . 103
6.5 Approximation Under Dynamic Popularity . . . 105
6.6 Summary and Conclusion . . . 106

7 Conclusion 107
7.1 Synopsis . . . 107
7.2 Future directions . . . 109

Bibliography 122

Publications 123

Glossary 125

List of Figures

1 General Telco-CDN architecture xvi

1.1 General Telco-CDN architecture 4

2.1 Video Traffic Growth Signals Major Shifts in Consumer Media Consumption Patterns. 10
2.2 Bootstrapping in mesh-based P2P video streaming 11
2.3 Common CDN architecture 13

3.1 Impact of k on the number of added edges 32
3.2 Impact of n on the average number of added edges. 32
3.3 An example of a directed 3-nearest graph. 33
3.4 Random allocation. 36
3.5 Intuitive heuristic. 36
3.6 Performance evaluation for small instances: Impact of n on the average cost. 37
3.7 Performance evaluation for small instances: Impact of k on the average cost. 38
3.8 Impact of n on the average cost for large instances. 39
3.9 Impact of k on the average cost for large instances. 40
3.10 Impact of n on the average and variance cost for large instances. . . . 41
3.11 Impact of k on the average and variance cost for large instances. . . . 42
3.12 Impact of n on the maximum cost. 43
3.13 Impact of k on the maximum cost. 43
3.14 Impact of n on the fairness metric. 43
3.15 Impact of k on the fairness metric. 43

4.1 Example of MCMF 51
4.2 Creation of offspring: an example with three repositories, having respectively 3, 2 and 3 storage capacities 53
4.3 Service usage 59
4.4 Envisioned telco-CDN topology in France. Three telco-CDN PoPs enable inter-connections with multiple traditional CDNs. 59
4.5 Impact of prediction on Perf-Opt 62
4.6 Normalized overall cost of push strategies 62
4.7 Optional caption for list of figures 63
4.8 Telco-CDN bandwidth utilization 64

5.1 Two end-users u_1 and u_2 requesting the same movie 69
5.2 Cooperative caching 79
5.3 Individual parallel caches 79
5.4 Individual caches in series 79
5.5 Cache diversity for the VoD service 82
5.6 Stored segments for the most popular movies of the VoD application. . 83
5.7 Number of times each server is accessed for the VoD service 83
5.8 Caching diversity for the time-shifted TV service 85
5.9 Stored segments for the channels of the time-shifted TV service. 85
5.10 Number of times each server is accessed for the time-shifted TV 85
5.11 Execution Process of CCNxProSimu 88
5.12 Example of configuration file 89
5.13 Example of network map 90
5.14 Example of traffic 90
5.15 Segments stored in CS 90
5.16 Server load & RTT 90

6.1 Single-cache simulation with static object popularity (On the y-axis, absolute hit-ratio produced by simulation.) 95
6.2 Multi-cache simulation with static object popularity. 96
6.3 Impact of the number of incoming request streams. 96
6.4 Example of cooperative cache 100
6.5 Comparison of approximation and simulation for a single cache with static object popularity (On the y-axis, ratio of hit-ratio approximation to hit-ratio simulation. The same in Fig. 6.6 and Fig. 6.11) 101
6.6 Multi-cache approximation performance with static object popularity. 102
6.7 Best configuration of γ for CRs with different entering degree 102
6.8 Hit ratio for different cache policies 103
6.9 Gain in terms of hit ratio of LRFU (multi-γ) versus LRU and LFU . . 103
6.10 Number of times each server is accessed 104
6.11 Comparison of approximation and simulation for a single cache with dynamic object popularity (On the y-axis, ratio of hit-ratio approximation to hit-ratio simulation). 105

List of Tables

3.1 Average optimality gap. 38

4.1 Notations 48

5.1 Latency and Message Overhead 86

6.1 Notations 97

Résumé

Le majority des fournisseurs de services Internet (FSI) envisagent la mise en place d'un réseau de distribution de contenus (RDC) dans leur propre infrastructure de réseau. Un tel déploiement n'est pas evident pour les opérateurs de réseaux, qui ne sont pas utilisés à des serveurs d'exploitation et les centres de données, mais il est devenu une évidence d'un point de vue commercial car l'Internet se transforme en une infrastructure orientée contenu. Ainsi, la gestion d'un soi-disant Telco-RDC est aujourd'hui un sujet de préoccupation critiques de l'entreprise ainsi que d'un problème scientifique d'une importance croissante [JZSRC09, Ray11, SVS12, LSHA+12]. Dans la section 1.1, nous allons développer l'analyse de marché de RDC d'aujourd'hui et expliquer en détail les raisons derrière la prolifération de Telco-RDC.

D'un point de vue scientifique, la gestion de RDC a été étudié pendant plus d'une décennie [PB07]. Cependant, les caractéristiques de Telco-RDC diffèrent de celles d'un RDC traditionnelle et donc remettre en question certaines des certitudes tout le monde d'accord sur. En particulier, l'objectif d'un RDC traditionnelle consiste à optimiser les performances d'une overlay, sous réserve de certaines contraintes liées à une infrastructure réseau sous-jacente que elle ne gère pas. Au contraire, une Telco-RDC est sous la responsabilité d'un opérateur de réseau qui possède à la fois la superposition et l'infrastructure sous-jacente. La principale mesure de performance est fondamentalement différente.

Dans cette thèse, nous avons ré-ouvrir un débat bien connu [HSXG09, WL09, PB07, WL11, KKGZ11], qui est central dans la gestion d'un RDC en général, et elle est essentielle dans le cas spécifique d'un Telco-RDC. Est-ce que l'opérateur doit décider le placement de contenu sur ses serveurs, ou faut-il mettre en œuvre une stratégie de cache pour ses serveurs? Cette dernière stratégie a été adoptée par la plupart des RDC à ce jour [PV06], car il est très simple à mettre en œuvre, et proche de l'optimum en termes de cache-hit, qui est la seule métrique comptée dans la traditionnelle RDC [SPVD09]. Chaque serveur remplace de façon dynamique le contenu qu'il stocke selon d'une politique de cache qui tient compte des dernières demandes des clients. La politique de cache Least Recently Used (LRU) est connue pour être particulièrement efficace malgré sa merveilleuse simplicité [Kon12]. En comparaison, le "push" stratégie, où l'opérateur décide de l'emplacement du contenu dans chaque serveur, est plus difficile à mettre en œuvre pour un bénéfice négligeable. La stratégie push requiert en effet un algorithme efficace qui prédit les futures demandes de la population. En outre, la détermination de la meilleure place est un problème d'optimisation dans la famille des problèmes de localisation des installations (FLP), qui sont souvent NP-complet.

Nous rouvrir ce débat pour plusieurs raisons. Tout d'abord, comme dit précédemment, l'objectif d'une Telco-RDC est différent. L'impact de la DRC sur l'infrastructure devient une mesure importante. L'objectif est de minimiser ce que nous appelons le coût des infrastructures. Deuxièmement, les performances des algorithmes de prédiction de recommandation ont considérablement augmenté au cours des cinq dernières années. Troisièmement, la taille de la superposition est beaucoup plus faible que ce qui était communément considéré dans les études antérieures, qui visaient services RDC à travers le monde. Ainsi, il devient possible de mettre en œuvre des algorithmes sophistiqués. Finalement, le simple réseau de cache peut également être améliorée pour s'adapter mieux l'architecture Telco-RDC.

Architecture de Telco-RDC

Plusieurs modèles ont été proposés pour capturer la réalité des échanges entre les entités commerciales dans le cadre de la livraison de contenu massive [MCL+11, DD11, FCB+08]. Aucun d'entre eux est tout à fait convaincantes, comme en témoigne le fait que, dans ces modèles, certaines entités sont engagés dans des accords qui devraient se traduire par des pertes économiques [LDD12]. En outre, de nombreux acteurs jouent plusieurs rôles. Par exemple, dans le modèle proposé par Ma *et al.* [MCL+11], Telefonica est tous ensemble un globe oculaire (un FAI qui servent les utilisateurs résidentiels) FAI, un contenu FSI (fournisseur de services Internet qui a déployé un large infrastructures pour la livraison de contenu), un fournisseur de services Internet Transit et un fournisseur de contenu. Nous ne sommes pas intéressés à la modélisation des interactions précisément monétaires. Nous visons plutôt à comprendre les motivations derrière les stratégies qui sont actuellement développées et analysées par les parties prenantes. Nous distinguons quatre gros acteurs principaux:

- **Les fournisseurs de contenu** veulent servir leurs abonnés résidentiels (utilisateurs finaux), sans avoir à engager offres spécifiques avec les FAI (à l'exception d'exclusivité potentielle ou des opérations de syndication, ce qui est des cas d'utilisation particulières). Ils veulent aussi maintenir des relations exclusives avec leurs clients. Parce que la personnalisation est considérée comme une voie prometteuse pour monétiser les services, les fournisseurs veulent être informés de chaque action de leurs clients. Par conséquant, l'interception du trafic non autorisées n'est pas acceptable. Enfin, les fournisseurs de services souhaitent réduire considérablement leur structure de coûts.

- **Fournisseurs de RDC** ont déployé une infrastructure de diffusion de contenu (serveurs, réseaux backbone potentiel) et des accords avec des fournisseurs de contenu. Ils veulent rester dans le jeu en revenant à des niveaux plus sains de la rentabilité, en se débarrassant des coûts variables et en trouvant la valeur ajoutée dans la gestion de contenu. Fournisseurs de RDC voulons aussi participer aux investissements d'infrastructures mondiales, afin d'améliorer l'équilibre et la valeur de leurs actifs de base (technologies, les connaissances et les infrastructures actuelles).

- **FSI** recevoir de l'argent à partir d'habitation des utilisateurs finaux. Ils veulent maintenir les structures de coûts sous contrôle afin de rester compétitif sur le marché de l'accès Internet (à savoir le maintien de bas prix / niveau de marge). Ils veulent aussi contrôler la qualité de service globale offerte à certains services et de maintenir un certain niveau de différenciation.

- **Les utilisateurs finaux** veulent accéder à un service, à tout moment et de n'importe où dans le monde. Comme la plupart des services sont gratuits ou pas cher, les utilisateurs finaux ont tendance à confondre les acteurs de la chaîne de valeur, ce qui conduit à des demandes d'une seule facture mensuelle globale afin d'accéder à n'importe quel service sur n'importe quel appareil à travers n'importe quel réseau.

Le changement soudain dans cet équilibre précaire du marché provient de l'explosion du trafic multimédia. Depuis fournisseurs de RDC avons déployé la plupart de leurs serveurs des réseaux des FSI, des goulets d'étranglement commence à apparaître dans les liens de peering entre FSI et les réseaux RDC, ce qui a conduit à une série d'affrontements entre les acteurs bien établis [ora12, Gol10]. Pour surmonter ce problème, les fournisseurs de RDC souhaitez déployer plus de serveurs directement dans les réseaux FSI et à l'ingénieur du trafic entre les utilisateurs finaux et les serveurs dans le réseau. Cette situation est inacceptable pour les FSI parce que les fournisseurs de RDC n'ont aucune connaissance globale sur le réseau sous-jacent.

Les Principaux à derrière de Telco-RDC

L'analyse ci-dessus met en évidence que de nombreux acteurs du marché les objectifs convergent (en dépit de quelques écarts). L'intérêt global des acteurs de la chaîne de valeur est convergeant vers une meilleure collaboration.

Plusieurs analystes estiment que les FSI devraient fournir de nouveaux espaces de rangement avec garantie des liens vers des clients [Ray09]. Cela constitue la soi-disant Telco-RDC. Quelques pionniers, notamment Verizon [Ver] et Comcast [Ful12], ont déjà construit leur Telco-RDC. Pour autant que nous savons, d'autres opérateurs de réseaux européens sont également construire des centres de données.

L'augmentation de Telco-RDC n'est pas nécessairement dangereux pour les fournisseurs de RDC traditionnels. Le rôle des RDC est d'obtenir et de distribuer du contenu des fournisseurs de services soit par ses propres serveurs, ou en déléguant cette distribution à la Telco-RDC. Ils peuvent également bénéficier de partager leur expérience dans les domaines de la diffusion de contenu avec les FSI. En dépit de la création récente de groupes d'intérêts pour une ouverture "fédératio" de Telco-RDC [BSB+13], un fournisseur RDC est dans une bonne position pour coordonner les multiples localisée Telco-RDC et pour l'interfaçage de ces derniers aux fournisseurs de contenu. De leur côté, les fournisseurs de contenu peuvent partager l'investissement requis de Telco-RDC de construire les structures de coûts non variables . Ils devraient également prendre en considération le respect des formats de distribution en fonction du mode de livraison préféré contenu des opérateurs de réseaux, par exemple [ope12].

Une collaboration entre ces acteurs conserve l'essence de l'Internet, qui est de permettre à chaque opérateur de réseau de gérer son propre Telco-RDC, selon les

carac-téristiques de son réseau. Par exemple, un FSI en train d'investir dans une nouvelle ligne de distribution data-centers mettra à profit ces petit ensemble de centres de stockage de masse, tandis qu'un fournisseur de services Internet présentant une large base de clients équipés de passerelles résidentielles de stockage compatibles mettra l'accent sur les mécanismes qui exploitent ces ressources . L'agrégation de Telco-RDC coopérer avec les RDC traditionnels s'aligne avec les politiques spécifiques à un domaine.

Dessinant L'architecture de Telco-RDC

Nous considérons pour la simplicité d'une entité unique fournisseur de services Internet, qui fournit des Accès à Internet pour les utilisateurs finaux et les contrôles d'accès au réseau, de points de peering à le dernier mile. L'idée derrière Telco-RDC est de installer **répertoire** près de commutateurs et de routeurs. En plus, les FSI s'appuient sur le récent déploiement de centres de données au sein de leurs réseaux. Passerelles domestiques et les set-top-boxes peuvent également devenir serveurs de contenu, tel que suggéré dans [WL11] . Nous décrivons un Telco-CDN dans la figure 1.

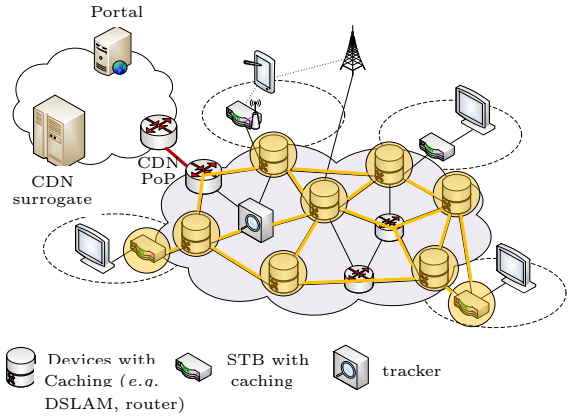


Figure 1: General Telco-CDN architecture

Nous appelons **tracker** l'entité qui gère le Telco-RDC et interfaces avec les réseaux RDC externes. Ces derniers télécharger de nouveaux contenus dans la Telco-RDC, et de déléguer leur distribution aux utilisateurs finaux qui êtes client du fournisseur d'accès.

Ce scénario est respectueux de tous les acteurs (fournisseur de contenu vidéo, RDC fournisseur et fournisseur d'accès Internet) et est déployable selon l'ordre du jour de chaque acteur. Il profite d'une meilleure utilisation des ressources du réseau, et, contrairement aux propositions basées sur le trafic interception [CJL+11], il ne contourne pas le prestataire de services et les RDC: ils ont encore reçu les demandes des clients, afin qu'ils sont en mesure de contrôler et de personnaliser leurs services. En outre, aux niveau de l'application, Réseau Orienté l'Information (ROI) est réalisable basé sur l'architecture de Telco-RDC décrit .

Contributions

Nous consacrons nos efforts pour le problème de placement des ressources de Telco-RDC. Nous nous appuyons sur le problème d'installation d'établissement (PIE) et le cache moins récemment ou fréquemment utilisés (CMR/CMR) de base visant à obtenir des solutions particulières pour l'attribution d'application et le contenu de Telco-RDC. Les principales contributions sont les suivantes.

Localisation des composants d'application. À l'ère des grands-données, la taille des services Internet modernes est extrêmement vaste. Il est fréquent que les services populaires comme les réseaux sociaux, des jeux en ligne et le streaming vidéo HD, appelez pour l'espace de stockage pour des centaines de téraoctets ou pétaoctets. Comme il est impossible de reproduire intégralement le service sur chaque serveur, l'idée est de le partitionner en éléments plus petits, puis localiser les composants sur des sites différents. C'est aussi la technique unique pour les Telco-RDC de réaliser à grande échelle de services de diffusion vidéo. Dans ce contexte, la rentabilité est un élément clé dans le déploiement de l'architecture décentralisée des services de livraison. Nous abordons cet aspect de l'optimisation et du point de vue algorithmique. Nous nous occupons de la mise en place des composants de service pour les sites du réseau, où la métrique de performance est le coût d'acquisition de composants entre les sites. Le problème d'optimisation qui en résulte, que nous appelons les k-composante mutisite placement problème (k-CMSP), s'applique à la distribution de services dans un large éventail de scénarios de réseaux de communication, y compris Telco-RDC. Nous fournissons une analyse théorique de la complexité algorithmique du problème, et de développer un modèle de programmation linéaire qui fournit un résultat de référence pour l'étalonnage des performances. Dans le côté algorithmique, nous présentons quatre approches: un algorithme d'approximation avec une garantie de trois algorithmes heuristiques. La première heuristique est dérivé de la théorie des graphes sur la partition Domatic. Le seconde heuristique, construite sur l'intuition, admet calcul distribué. Le troisième heuristique met l'accent sur l'équité dans la répartition des coûts entre les sites. Nous présentons les résultats de simulation pour les ensembles de réseaux, où le coût est représenté par temps aller-retour (TAR) provenant de mesures réelles. Pour les petits réseaux, le modèle entier est utilisé pour étudier la performance des algorithmes en termes d'optimalité. Grands réseaux sont utilisés pour comparer les algorithmes rapport à l'autre. Parmi les algorithmes, l'heuristique basée sur l'intuition a proximité à des performances optimales, et l'heuristique de l'équité constitue un bon équilibre entre un site unique et le coût global. En outre, les expériences démontrent l'importance de l'optimisation de réduction des coûts par rapport à une stratégie de répartition aléatoire.

Algorithme optimal pour le placement vidéo basé sur push. Pour un opérateur de Teclco-RDC qui veut optimiser l'utilisation de son infrastructure, il est nécessaire de déterminer le placement optimal de la vidéo pour la gestion du trafic sur ses réseaux sous-jacents. Nous affirmons ici que le réseau dans le monde réel impose la différenciation des liens avec une fonction de coût générique. En Telco-RDC, nous supposons que le FSI sait explicitement la capacité de service de chaque serveur et la

préférence lien interne. En outre, les demandes des utilisateurs sont prévisibles (*ie*, l'ensemble des vidéos qui seront demandés par un utilisateur peut être prévu par le fournisseur de contenu, RDC ou le FSI lui-même). Sur la base de ces hypothèses, nous présentons une pratique quasi-optimale algorithme pour le placement du contenu dans Telco-RDC. En d'autres termes, nous montrons qu'un opérateur de réseau est en mesure de mettre en œuvre une stratégie de pousser les contenus très efficace si elle juge que cette mise en œuvre est intéressante. Notre idée ici est d'utiliser un méta-heursitique bien connu, notament algorithme génétique (AG), pour atteindre un niveau de performances qui est très proche de l'optimum. En outre, nous présentons une implémentation de l'algorithme générale dans le cadre de MapReduce, ce qui permet le calcul de Telco-RDC dans un grand echelle sur une petite grappe des machines. Nous montrons dans une évaluation réaliste des avantages on peut attendre d'une stratégie poussée pour les Telco-RDC. Notre algorithme permet la comparaison avec les stratégies de cache traditionnels. Nous avons recueilli des traces réelles d'un service de vidéo à la demande qui devrait normalement être hébergé dans un Telco-RDC. Nous étudions un déploiement Telco-CDN qui est actuellement étudié par un major opérateur de réseau européen (Orange) et une politique de la circulation simple mais instructif de gestion. Notre principale observation est que la mise en cache LRU fonctionne aussi bien que notre stratégie poussée pour le taux de succès. Cependant, une stratégie poussée réduit considérablement l'impact sur l'infrastructure sous-jacente.

Protocole de cooperatif cacheing dans ROI. Le déploiement de routeurs Internet ayant des capacités de cache (CR pour le contenu ou cache routeur [LRH10]) est une opportunité pour réformer la redirection demande de Telco-RDC d'un mode réseau d'orientée contenu. Un réseau de CR est en particulier un facteur clé des projets liés à la ROI, où les demandes ne sont plus acheminés vers une destination unique et tout autre équipement peut agir en tant que serveur de contenu [JST+09]. Nous nous référons à la gestion des caches d'un réseau de CR comme *cache au seins du réseau*. La recherche dans ce domaine permet à l'exploitation des ressources de cache de CR. Jusqu'à présent, seule une politique LRU de base est mis en œuvre sur tous les CR proposé. Notre contribution est la proposition d'une stratégie de cache coopératif dans le cadre de Content-Centric-Network (CCN), un modèle typique du ROI. Il est conçu pour le traitement des flux vidéo de grande taille avec un accès à la demande. Cette stratégie de cache combine les stratégies traditionnels basés sur le hachage et le répertoire, et répond au besoin de le FSI en réduisant de moitié le trafic inter-domaines. Nous illustrons d'abord les changements qui doivent être portées à la CCN protocole pour mettre en œuvre cette stratégie. Par la suite, nous montrons les avantages de cette politique de coopération au cours standards des politiques non coopératives dans les structures de réseau simples. Enfin, nous décrivons une version augmentée du protocole CCNx, mise en œuvre de cette politique, et nous présentons un ensemble de simulations qui ont été effectuées sur une plate-forme expérimentale pour CCNx. L'avantage de notre protocole de mise en cache coopérative atteint 45% en terme de réduction de la trafic inter-domaine.

Modèle analytique pour évaluer multi-politiques. La dernière contribution de cette thèse est toujours effectué sur la cache en réseau. Après nous mettons en

œuvre notre protocole de coopération dans la cache CCNx, nous aimerions avoir une meilleure compréhension du comportement des politiques de cache autres que LRU dans un système de multi-cache. Malheureusement, les travaux antérieurs portaient sur l'analyse de la cache en réseau dans le CCN ont considéré que la LRU. Plus généralement, la communauté scientifique manque de méthodes d'analyse et d'évaluation des politiques de mise en cache dans le cache générique multi-topologies. Donc, voici nos contributions sont de deux ordres. Tout d'abord, nous présentons un outil d'analyse qui se rapproche de systèmes génériques de cache en réseau et la performance des politiques de cache qui sont basés sur l'analyse de récence et la fréquence des demandes. Pour valider cet outil d'analyse, nous comparons les performances théoriques des CR à celle estimée à partir des simulations. Deuxièmement, nous présentons un *multi-politique* en cache réseau, où chaque CR met en œuvre sa politique propre cache en fonction de son emplacement dans le réseau. Les résultats obtenus avec notre outil analytique donner une méthode simple pour déterminer les politiques optimales pour la cache des CR. Nous démontrons l'intérêt de notre multi-politique en réseau en mettant en œuvre l'approche de cache d'un réseau de nœuds CCNx dans le cadre d'une application de VoD. Comme présenté dans [CGMP11] , nous supposons que l'opérateur a réservé une partie des ressources de cache du CRS pour cette application. Par rapport à la seule politique LRU, nous montrons que l'approche multi-politique augmente les performances en termes de taux de succès du système de cache en réseau de 16%.

Organisation

Le reste de cette thèse est organisé comme la suit.

Le chapitre 2 donne à la fois le travail pratique et théorique sur de Telco-RDC. Dans les aspects pratiques, nous présentons une introduction brève de technologies de distribution de contenu y compris Peer-to-Peer systèmes, Réseau de Distribution de Contenu et le Réseau Orienté l'information. Grâce à ces modifications, nous soulignons le bénéfice potentiel peut être obtenue à partir de l'application de l'architecture Telco-RDC. Sur le aspect théorique, nous élargissons la recherche sur le problème d'installation d'établissement et caching au seins du réseau.

Le chapitre 3 présente la formulation mathématique de k-CMSP, et met en lumière des modèles et des algorithmes correspondants. Concrètement, nous avons d'abord fournir une analyse théorique approfondie du problème d'optimisation. En particulier nous montrons que, dans le cas général, k-CMSP est NP-complet. Comme conséquence immédiate, deux problèmes d'optimisation liés, où chaque site peuvent être attribuées plusieurs éléments, et où seul un sous-ensemble de tous les composants sont demandés par chaque site sont à la fois NP-complet aussi. Ensuite, nous formulons k-CMSP au moyen d'un modèle de programmation en nombres entiers. La résolution du modèle donne des solutions exactes de k-CMSP, qui est faisable dans les petits réseaux. Ces deux enquêtes fournissent une base scientifique pour des recherches approfondies

sur k-CMSP. Par la suite, nous illustrons plusieurs algorithmes et d'enquêter sur leur performance au moyen de simulations.

Le chapitre 4 décrit l'algorithme génétique qui calcule le placement optimal vidéo dans Telco-RDC. Puisque l'algorithme génétique devrait être parallélisés dans MapReduce afin de faire face à une entrée énorme ensemble de données, nous donnons une courte introduction sur l'algorithme génétique parallèle et le cadre MapReduce. Ensuite, nous formulons le problème par k-PCFLP et la conception de l'algorithme génétique centralisée. Ensuite, nous détaillons le processus de parallélisation de l'algorithme centralisé. Nous procédons à des évaluations et comparer les résultats fournis par notre algorithme et de répartition des autres à la fin.

Le chapitre 5 propose le protocole de coopération pour la cache CCN. Nous donnons tout d'abord le modèle de réseau et les informations détaillées sur notre protocole de cache coopératif. Ensuite, nous donnons l'analyse théorique qui met l'accent sur l'avantage de la cache coopérative. Par la suite, nous détaillons la réalisation de notre plate-forme de simulation, CCNxProSim, qui se déploie automatiquement prototype CCNx sur des machines réelles. Évaluation pratique fondée sur CCNxProSim est indiqué à la fin du chapitre.

Le chapitre 6 construit le modèle stochastique pour l'évaluation multi-politiques dans le réseau de cache. Nous introduisons la politique de mise en cache LRFU considérant à la fois la nouveauté et la fréquence des références aux objets dans un premier temps. Ensuite, nous présentons notre étude basée sur la simulation de la performance cache LRFU. Le modèle d'approximation pour le multi-cache politiques de mise en cache LRFU est défini et validé lors de la prochaine étape. Enfin, nous mettons en évidence l'utilité de notre rapprochement, et étend notre base rapprochement LRFU de sorte qu'il peut approcher les performances de cache à l'objet avec popularité varié.

Le chapitre 7 conclut la thèse et discuter de la perspective des travaux futurs.

Keywords: Facility Location Problem, MapReduce, Caching Policies, CDN, ICN.

Chapter 1

Introduction

Major Internet Service Providers (ISPs) are considering the deployment of a Content Delivery Networks (CDNs) in their own network infrastructure. Such deployment is not trivial for network operators, which are not used to operating servers and datacenters, but it has become an evidence from a business perspective since the Internet is turning into a content-centric infrastructure. Thus, the management of a so-called *Telco-CDN* is today a critical business concern as well as a scientific problem of growing importance [JZSRC09, Ray11, SVS12, LSHA+12]. In section 1.1, we will develop today's CDN market analysis and explain in details the rationale behind the proliferation of Telco-CDNs.

From a scientific standpoint, the management of CDN has been studied for more than a decade [PB07]. However, the characteristics of Telco-CDN differ from those of a traditional CDN and thus challenge some of the certitudes everybody agreed on. In particular, the goal of a traditional CDN is to optimize the performance of an *overlay*, subject to some constraints related to an underlying network infrastructure, which it does not manage. On the contrary, a Telco-CDN is under the responsibility of a network operator that owns both the overlay and the underlying infrastructure. The main performance metric is fundamentally different.

In this thesis, we re-open a well-known debate [HSXG09, WL09, PB07, WL11, KKGZ11], which is central in the management of a CDN in general, and is critical in the specific case of a Telco-CDN. *Should the operator decide the placement of content into its servers, or should it implement a caching strategy for its servers?* This latter strategy has been adopted by most CDNs so far [PV06] because it is very simple to implement, and near optimal in terms of cache hit, which is the only metric that matters in traditional CDN [SPVD09]. Every server dynamically replaces the content it stores based on a caching policy that takes into account latest requests from clients. The *Least Recently Used* (LRU) caching policy is known to be especially efficient despite its gorgeous simplicity [Kon12]. In comparison, the "push" strategy, where the operator decides the location of content into every server, is harder to implement for a negligible benefit. The push strategy requires indeed an efficient algorithm that predicts the future requests of population. Moreover, determining the best placement is an optimization problem in the family of Facility Location Problems (FLP), which are commonly NP-complete.

We re-open this debate for several reasons. Firstly, as previously said, the ob-

jective of a Telco-CDN is different. The impact of the CDN on the infrastructure becomes a major metric. The goal is to minimize what we call the *infrastructure cost*. Secondly, the performances of recommendation and prediction algorithms have significantly grown during the past five years. Thirdly, the size of the overlay is far smaller than what was commonly considered in previous studies, which targeted worldwide CDN services. Thus, it becomes possible to implement sophisticated algorithms. Finally, the simple in-network caching can also be improved to fit better the Telco-CDN architecture.

1.1 Telco-CDN Architecture

Several models have been proposed to capture the reality of exchanges between business entities in the context of massive content delivery [MCL+11, DD11, FCB+08]. None of them is fully convincing, as demonstrated by the fact that, under these models, some entities are engaged in agreements that should result in economical losses [LDD12]. Moreover, many actors play several roles. For example, in the model proposed by Ma *et al.* [MCL+11], Telefonica is all together an Eyeball ISP (an ISP that serve residential users), a Content ISP (an ISP that has deployed a broad infrastructure for content delivery), a Transit ISP and a Content Provider. We are not interested in precisely modeling monetary interactions. We rather aim to understand motivations behind the strategies that are currently developed and analyzed by stakeholders. We roughly distinguish four main actors:

- **Content providers** want to serve their subscribers (residential end-users) without having to engage specific deals with ISPs (excluding potential exclusivity or syndication deals, which are particular use cases). They also want to maintain exclusive relationships with their clients. Since personalization is considered as a promising way to monetize services, the providers want to be notified of each and every action of their clients. Therefore traffic interception by un-authorized third-parties is not acceptable. Lastly, service providers wish to drastically lower their cost structure.

- **CDN providers** have deployed a content delivery infrastructure (servers, possibly backbone networks) and have agreements with content providers. They want to remain in the game by returning to more healthy levels of profitability, by getting rid of variable costs and by finding added value in content handling. CDN providers also want to participate to global infrastructure investments, in order to better balance and value their core assets (technologies, knowledge and current infrastructures).

- **ISPs** receive money from residential end-users. They want to maintain cost structures under control in order to remain competitive on the Internet access market (*i.e.* maintaining low pricing/level of margin). They also want to control the global QoS offered to certain services and to maintain a certain level of differentiation.

- **End-users** want to access any service, at any time and from anywhere in the world. As most services are either free or cheap, end-users tend to confound the actors in the value chain, which leads to requests for one overall monthly bill in order to access any service on any device through any network.

The sudden shift in this precarious market equilibrium comes from the explosion of multimedia traffic. Since CDN providers have deployed most of their servers out of ISP networks, bottlenecks starts appearing in the peering links between ISP and CDN networks, which has led to a series of clashes between well-established actors [ora12, Gol10]. To overcome this problem, CDN providers would like to deploy more servers directly in ISP networks and to engineer the traffic between residential end-users and these "in-network servers". This is unacceptable for ISPs because CDN providers have no global knowledge on the underlying network.

1.1.1 Rationales behind Telco-CDNs

The above analysis highlights that many market players objectives converge (in spite of some disparities). The global interest of the value chain players is converging toward a better collaboration.

Multiple analysts consider that ISPs should provide new storage spaces with guaranteed links to the customers [Ray09]. This forms the so-called Telco-CDN. Some pioneers, including Verizon [Ver] and Comcast [Ful12], have already built their Telco-CDN. As far as we know, other European network operators are also building data-centers.

The raise of Telco-CDNs is not necessarily dangerous for traditional CDN providers. The role of the CDNs is to get and distribute contents from the service providers either by its own servers or by delegating this distribution to the Telco-CDNs. They may also profit in sharing their advanced experience of content delivery with ISPs. Despite the recent creation of groups of interests for an open "federation" of Telco-CDNs [BSB+13], a CDN provider is in a good position for coordinating multiple localized Telco-CDNs and for interfacing these latter to the content providers. On their side, content providers can share required investment of Telco-CDN to build non-variable costs structures. They would also consider respecting distribution formats to match the preferred content delivery mode of network operators, for example [ope12].

A collaboration between these actors preserves the essence of the Internet, which is to let every network operator manage its own Telco-CDN, according to the characteristics of its network. For instance, an ISP currently investing in a new line of distributed data-centers will leverage these small set of massive storage centers, whereas an ISP presenting a large base of customers equipped with storage-enabled home gateways will focus on mechanisms that exploit these resources. The aggregation of Telco-CDNs cooperating with traditional CDNs aligns with domain specific policies.

1.1.2 Sketching Telco-CDN architecture

We consider for simplicity a single ISP entity, which provides Internet access to end-users and controls the access network, from peering points to the last mile. The idea

behind Telco-CDN is to install **repositories** near switches and routers. More probably, ISPs leverage the recent deployment of data-centers within their networks. Home gateways and set-top-boxes can also become content servers, as suggested in [WL11]. We illustrate an a Telco-CDN in Figure 1.1.

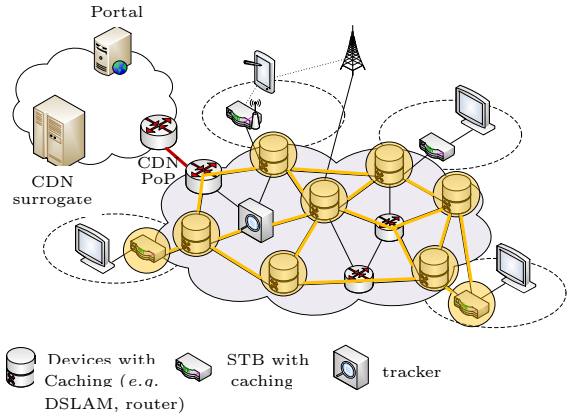


Figure 1.1: General Telco-CDN architecture

We call **tracker** the entity that manages the Telco-CDN and interfaces with the external CDNs. These latter upload new contents in the Telco-CDN, and delegate their distribution to the end-users that are client from the ISP.

This scenario is respectful of all actors (video provider, CDN provider and ISP) and is deployable according to the agenda of each actor. It takes advantage of a better network resource utilization, and, contrarily to proposals based on traffic interception [CJL+11], it does not bypass the service provider and the CDNs: they still received the requests from clients, so they are able to monitor and personalize their services. Moreover, an application level Information Centric Network (ICN) is achievable based on the described Telco-CDN architecture.

1.2 Contributions

We devote our efforts to the resource placement problem in Telco-CDN. We leverage the traditional facility location problem (FLP) and basic Least Recently or Frequently Used (LR/FU) caching policies to derive particular solutions for allocating application and content in Telco-CDN. The major contributions are as follows.

Application components allocation. In the age of big-data, the size of modern Internet services is extremely large. It is common that popular services as social networks, on-line games, and HD video streaming, call for the storage space in order of hundreds of Terabytes or Petabytes. Since it is impossible to fully replicate the service on each server, the idea is to partition it into smaller components, and then locate the components on distinct sites. It is also the unique technique for Telco-CDN to achieve large-scale video delivery service. In this context, cost efficiency is a key aspect in

deploying the decentralized service delivery architecture. We address this aspect from an optimization and algorithmic point of view. We deal with the placement of service components to network sites, where the performance metric is the cost for acquiring components between the sites. The resulting optimization problem, which we refer to as the k-Component Multi-Site Placement Problem (k-CMSP), applies to service distribution in a wide range of communication networking scenarios including Telco-CDN. We provide a theoretical analysis of the problem's computational complexity, and develop an integer programming model that provides a reference results for performance benchmarking. On the algorithmic side, we present four approaches: an algorithm with approximation guarantee and three heuristics algorithms. The first heuristic is derived from graph theory on domatic partition. The second heuristic, built on intuition, admits distributed computation. The third heuristic emphasizes on fairness in cost distribution among the sites. We report simulation results for sets of networks where cost is represented by round-trip time (RTT) originating from real measurements. For small networks, the integer model is used to study algorithm performance in terms of optimality. Large networks are used to compare the algorithms relatively to each other. Among the algorithms, the heuristic based on intuition has close-to-optimal performance, and the fairness heuristic achieves a good balance between single-site cost and the overall one. In addition, the experiments demonstrate the significance of optimization for cost reduction in comparison to a random allocation strategy.

Push-based optimal algorithm for video placement. For a Telco-CDN operator who wants to optimize the utilization its infrastructure, it is necessary to determine the optimal placement of video for managing traffic on its underlying networks. We argue here that real-world network engineering imposes differentiating links with a generic cost function. In Telco-CDN, we assume that the ISP knows explicitly the service capability of each server and internal link preference. Besides, users' requests are predicable (*i.e.*, the set of videos that will be required by a certain user can be foreseen by content providor, CDN or the ISP itself). Based on these assumptions, we present a practical, quasi-optimal algorithm for content placement in Telco-CDN. In other words, we show that a network operator is able to implement a very efficient push strategy if it judges that such implementation is worthwhile. Our idea here is to leverage a well-known meta-heuristic, namely genetic algorithm (GA), to reach a level of performances that is very close to optimal. In addition, we present an implementation of this GA on the MapReduce framework, which enables computation of large Telco-CDN instances on a small cluster of machines. We show in a realistic evaluation the benefits one can expect from a push strategy for Telco-CDN. Our algorithm enables the comparison with traditional caching strategies. We collected real traces from a VoD service that should typically be hosted in a Telco-CDN. We study a Telco-CDN deployment that is currently investigated by a major European network operator (Orange) and a simple but enlightening traffic management policy. Our main observation is that LRU caching performs as well as our push strategy for the hit ratio. However, a push strategy significantly reduces the impact on the underlying infrastructure.

Cooperative caching protocol in ICN. The deployment of Internet routers having caching capabilities (CR for Content or Caching Router [LRH10]) is an opportunity to reform the request redirection of Telco-CDN to a network-layer content oriented fashion. A network of CR is in particular a key component of projects related to ICN, where requests are no longer routed toward a unique destination and any equipment can act as server [JST+09]. We refer to the management of caches of a network of CRs as *in-network caching*. The research in this area enables the exploitation of the caching resources of CRs. So far, only a basic LRU policy implemented on every CRs has been proposed. Our contribution is the proposition of a cooperative caching strategy in the context of Content-Centric-Network (CCN), a typical model of ICN. It has been designed for the treatment of large video streams with on-demand access. This caching strategy combines the traditional hash-based and directory-based cooperative caching schemes, and addresses the need of ISP by halving the cross-domain traffic. We illustrate first the changes that have to be brought to the CCN protocol in order to implement this strategy. Thereafter, we prove the advantages of this cooperative policy over standard non-cooperative policies in simple network structures. Finally, we describe an augmented version of the CCNx protocol implementing this policy, and we present a set of simulations, which have been conducted on an experimental platform for CCNx. The benefit of our cooperative caching protocol reaches 45% in term of the reduction of inter-domain traffic.

Analytical model for evaluating multi-policies. The last contribution of this thesis is still conducted on the in-network caching. After we implement our cooperative caching protocol in CCNx, we would like to have a deeper understanding of the behavior of various caching policies other than LRU in a multi-cache system. Unfortunately, previous works related to the analysis of in-network caching in CCN have considered only the LRU. More generally, the research community lacks methods for analyzing and evaluating caching policies in generic multi-cache topologies. So, here our contributions are twofold. First, we present an analytical tool that approximates in generic in-network caching systems the performance of the caching policies that are based on the analysis of both recency and frequency of requests. To validate this analytical tool, we compare the theoretical performance of CRs to the one estimated from simulations. Second, we present a *multi-policy* in-network caching, where every CR implements its own caching policy according to its location in the network. The results obtained with our analytical tool yield a simple method to determine the optimum caching policies for the CRs. We demonstrate the interest of our multi-policy in-network caching approach by implementing a network of CCNx nodes in the context of a VoD application. As presented in [CGMP11], we assume that the operator has reserved a portion of the CRs' caching resources for this application. Compared to the single LRU policy, we show that the multi-policy approach increases the performance in terms of hit-ratio of the in-network caching system by 16%.

1.3 Organization of Dissertation

The rest of this thesis is organized as follows.

Chapter 2 gives both the practical and theoretical related work of Telco-CDN. In the practical aspects, we present brief introductions of content delivery technologies including Peer-to-Peer systems, Content Delivery Network and Information Centric Network. Through these revisions we highlight the potential benefit can be obtained from applying the Telco-CDN architecture. On the theoretical side, we expand the research on facility location problem and in-network caching.

Chapter 3 presents the mathematical formulation of k-CMSP, and elucidates corresponding models and algorithms. Concretely, we first provide a thorough theoretical analysis of the optimization problem. In particular we prove that, in the general case, k-CMSP is NP-complete. As an immediate consequence, two related optimization problems, where every site may be allocated several components, and where only a subset of all components are requested by each site are both NP-complete as well. Next, we formulate k-CMSP by means of an Integer Programming model. Solving the model gives exact solutions of k-CMSP, which is doable in small networks. These two investigations provide a scientific background for extensive research on k-CMSP. Thereafter, we illustrate several algorithms and investigate their performance by means of simulations.

Chapter 4 describes the genetic algorithm that calculates the optimal video placement in Telco-CDN. Since the genetic algorithm should be parallelized in MapReduce so as to deal with a huge input data-set, we give a short introduction about parallel genetic algorithm and the MapReduce framework. Next, we formulate the problem by k-PCFLP and design the centralized genetic algorithm. Then we detail the parallelization process of the centralized algorithm. We conduct evaluations and compare the results yielded by our algorithm and other allocation schemes at last.

Chapter 5 proposes the cooperative caching protocol for CCN. We give firstly the network model and the detailed information about our cooperative caching protocol. Next, we derive the theoretical analysis that emphasizes the advantage of our cooperative caching. Thereafter, we detail the realization of our simulation platform, CCNxProSim, which automatically deploys CCNx prototype on real machines. Practical evaluation based on CCNxProSim is shown at the end of the chapter.

Chapter 6 builds the stochastic model for evaluating multi-policies for in-network caching. We introduce the LRFU caching policy considering both the recency and the frequency of the references to the cached objects at first. Then, we present our simulation based study of LRFU caching performance. The approximation model for multi-cache LRFU caching policies is derived and validated in the next step. Finally, we highlight the usefulness of our approximation, and extends our basic LRFU approximation so that it can approximate the caching performance with changing object popularity.

Chapter 7 concludes the whole thesis and discuss the prospect of the future work.

Chapter 2

Resource Placement Overview: Practice and Theory

2.1 Introduction

This chapter presents the state-of-arts of resource placement problem in both practical and theoretical aspects. We describe several video delivery systems related to Telco-CDN, including P2P, CDN, and ICN. On the theoretical side, we provide some references of both push-based and pull-based approaches. Specifically, we introduce the *Facility Location Problem* (FLP), based on which we formulate our particular problems. Related work about caching will be given at the end of this chapter.

2.2 Practical Aspects

The rising popularity of video streaming services has resulted in increased volumes of network traffic. More than one billion of users access daily video content. YouTube statistics reported that the site hits 4 billion views per day, and deals with 60 hours of uploaded content every minute. Over 800 million unique users visit YouTube and over 3 billion hours of video are watched each month in 2012 [Smi12, You12]. Almost all video content providers profit from the mushrooming number of subscribers. Figure 2.1 shows the growing trend of video traffic over global IP network [RH12]. While the traffic in 2010 was $5,000$ petabytes per month, it tops $35,000$ in the prediction for 2015. The compound annual growth rate (CAGR) of various streaming services reaches 48%.

We classify streaming services into three categories: *live streaming, video-on-demand (VoD) and time-shifted*. Live streaming service offers real time video content to all users in a synchronized way. In comparison to live streaming, VoD systems allow users to request any video content in an interactive VCR fashion. That is, the user can start to watch a video in any position at any time, and execute other operations including pause, forward, random seek, *etc.* Time-shifted service has recently become important. It combines the characteristics of live streaming and VoD. On one

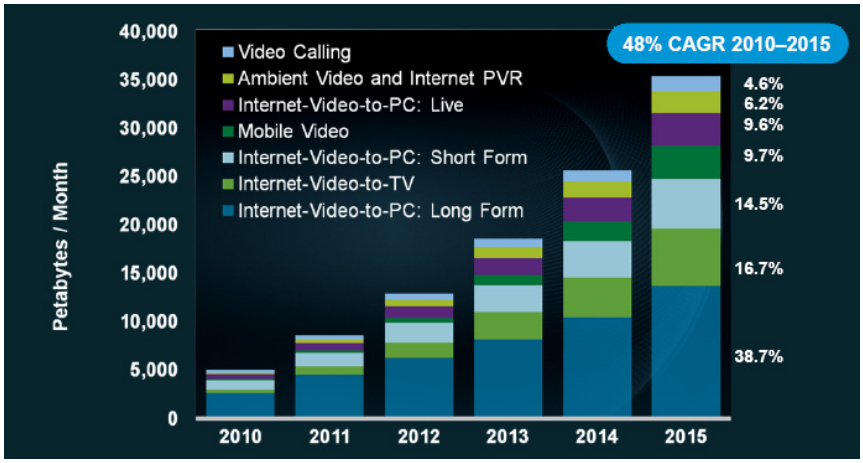


Figure 2.1: Video Traffic Growth Signals Major Shifts in Consumer Media Consumption Patterns.

hand, users can enjoy real time TV programs. On the other hand, the programs that have already been broadcasted are also available with VCR-like operations.

The sharp growth of video traffic, the newly emerging services and the development of last mile technology have moved the Internet bottleneck from access network to peering links [SRP+09, Lei09a]. The evolution of the Internet service is calling for shifting not only the current content delivery technologies but also the common business model in content delivery value chain. We argue that an efficient and profitable way to tackle the congestions is to make the contents available inside ISP's networks. We take advantage of various traditional technologies such as P2P, CDN as well as the clean-slate ICN proposition to conceive a new network architecture called Telco-CDN for streaming delivery service. Following paragraphs introduce the P2P, CDN and ICN systems. Through these introductions, we reveal the shortcomings of each of these technologies, and at the meantime, emphasis the necessity of Telco-CDN.

2.2.1 P2P Streaming Technology

For decades, P2P technology has been an interesting topic, which has received a special attention of scientific community when this technology addresses both VoD and live streaming services. Participants or peers in P2P system contribute their resources as storage capacity, processing power and download bandwidth to other users while they are enjoying the service. Therefore, peers can be regarded as both suppliers and consumers of resources. This dual identity of peers gives several advantages to P2P systems on traditional client-server systems. First of all, thanks to the contribution of participants, P2P system is self-scalable. Second, since cooperation between peers is usually implemented with multiple simultaneous downloading and uploading connections, P2P systems are more robust than client-server systems where the server is a single point of failure. With these intrinsic appealling features, P2P systems have attracted substantial efforts from research societies on different aspects of video streaming delivery, especially on the construction of P2P overlay network

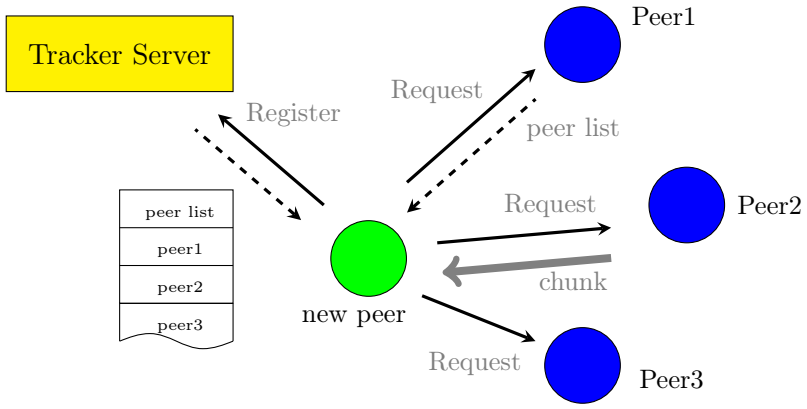


Figure 2.2: Bootstrapping in mesh-based P2P video streaming

[DHT08, CSJ+08, LQK+08, DN08, HBM+08].

There are three distinct structures of P2P overlay network which are *tree based, mesh based* and *distributed hash table (DHT) based*. Tree based systems organize peers and links as a tree rooted by the source. Video chunks are pushed from the parent node to its children. In a mesh based system, the streaming topology is elastic. Peers maintain the neighbor relationship with multiple peers that having the common interest on a video, and obtain streaming data from these neighbors simultaneously. DHT based structure is only proposed for VoD and time-shifted services. It is used to keep trace of owners hosting each video part and facilitate the search process of the owner. Among the three structures, only mesh based structure has been successfully deployed and widely used in large scale systems as CoolStreaming [LQK+08], PPLive [PPl] and PPStream [PPS]. Therefore, we give some further inspections of the implementation of mesh based structure in real world video streaming application.

A mesh based system is composed by the following major components: i) *Source servers* are responsible for delivering all video chunks to peers; ii) *Tracker servers* maintain the meta-data of video chunks and the overlay network of the system; iii) *Peers* are contributors of system resource and consumers of video content.

Tracker servers build and terminate the relationships between peers according to the data and bandwidth availability on peers. The data are video chunks that each of them presents a small period of playback. Usually, a video is divided into many uniformly sized chunks. Every chunk is associated with a sequence number or *chunk ID* represents the playback order of the chunk. The information about the chunk availability in the buffer of a peer can be shared among peers or obtained from tracker servers. The propagation of the information between peers is realized by the periodical exchange of *buffer maps*, which contains a series of bits indicating the availability of chunks.

The bootstrapping process, shown in Figure 2.2, allows new peers to join a P2P system and locate chunks that they require [Liu11]. While a peer enters the system, it obtains a *peerlist* containing a list of potentially active peers from the tracker server. Those active peers either accept directly the new arrival as their overlay neighbor, or recommend it to connect with other peers.

Peers discover and maintain their neighbors either through the overlay network managed by the tracker server or gossip protocol [JKvS04]. These processes enable peers to find the chunks that they request on other peers. Using the tracker server, it is mandatory for peers to inform the server when they finish downloading a chunk or the chunk is eliminated. The alternative is the gossip protocol, where every node sends a message about its buffer map to a set of randomly selected peers periodically. This knowledge is compacted and potentially re-diffused. Thus, the buffer maps of adjacent peers in the overlay network is relatively consistent.

Data transmission in mesh based systems is carried out by the pull approach. Explicit requests are sent by the demander to the holders of the chunk. The knowledge of the buffer map of neighbors is essential in such a request driven system.

Mesh based P2P streaming systems have been proved to be robust, scalable and is able to efficiently utilize the upload bandwidth of end users. Gossip protocols ensure quick neighbor discovery so that a peer can continue the download task from other neighbors in the case of peer churn. In the users point of view, P2P streaming is a good choice because of its outstanding performance while the uncontrollable P2P traffic is quite troublesome for ISP. Since the cross-domain traffic may cause high operation cost, the network friendliness of P2P system has been the main problem for years.

2.2.2 Content Delivery Network

Acting as intermediaries between content providers and consumers, CDNs have proliferated rapidly with the explosive increment of Internet service since the year of two thousand. They leverage the replication of content on edge servers close to users to guarantee the availability and the fast delivery of content. Over the years, CDNs have evolved themselves from the simple server-client service mode to the hybrid peer-assisted CDN mode due to the continuously augmenting users' requirements.

Figure 2.3 shows the common architecture of a traditional CDN. The system is composed by seven logical components and the relationships among them in the service process are detailed as follows:

1. The *origin server* sends its URI name space for document objects to the *request routing system*. Then the request routing system can work as a tracker.

2. The origin server pushes the content need to be delivered into the *distribution system*.

3. The distribution system is responsible for interacting with request routing system to select and move content to *edge servers*.

4. The request of *end user* is redirected to the request routing system.

5. The request routing system send back the IP address of a suitable edge server to the end user.

6. The selected edge server delivers the requested content to the end user. At the same time, it sends accounting information to the *accounting system*.

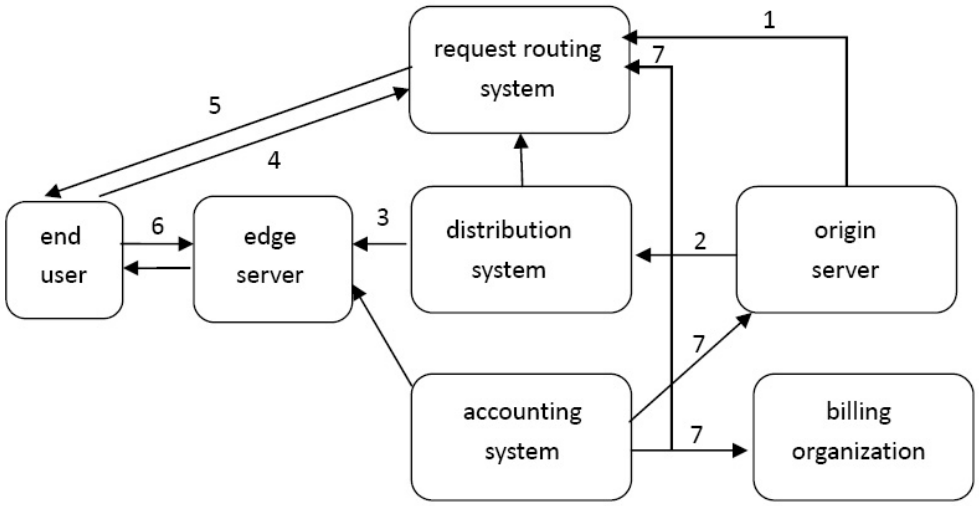


Figure 2.3: Common CDN architecture

7. The accounting system refines the accounting information and sends the refined information to different component as feedback.

In the deployment of such a traditional CDN, it is necessary to pay attention to several issues including *replica placement*, *server selection* and *request routing technique*. Replica placement decides where would be the best location of edge servers and which content should be assigned to them. These problems have been addressed by various studies [MRKR99, QNM01, GYH10]. Usually the decision of the best location of edge servers is modeled by the κ -media problem [JJJ$^+$00] or k-hierarchically well-separated trees problem. [Bar96]. Due to the computational complexity of solving these well form problems, some heuristics such as greedy replica placement[KRS00], topology-informed placement [JJK$^+$01], hot spot based strategy [QPV01] are proposed. The solutions for content placement falls into three classes: cooperative push-based, non-cooperative pull-based and cooperative pull-based, which are the focuses of the research in this thesis.

In server selection, two factors are usually taken into account: the server-client distance and the load balance on edge servers. In real systems, a request is often redirect to a server which is geographically proximate by comparing IP addresses. The request routing system gathers the information about load balance on servers by message exchange.

Many techniques have been proposed to redirect client requests to particular edge servers. These techniques can be summarized into the following four categories:

- End user multiplexing: End user receives the addresses of a set of candidate edge servers and chooses one to send the request. This technique does not consider the work load of edge server, since the client lacks the overall information of the CDN.

- HTTP redirection: The request of end user is received and redirected only by origin server. It is the simplest but the least efficient technique, because the origin server in this scheme could be a bottleneck and a single failure point.

- DNS indirection: This scheme uses DNS modifications to return the IP addresses of one or two suitable edge servers. The server selection is managed by CDN provider so that the heavy loaded servers can be avoided. This technique is used by some famous CDN providers as Akamai and Limelight.

- Anycasting: In this scheme, a *group* of edge servers is defined by an IP address or a URL of content. The anycast-aware routers are responsible for forwarding the client request to at least one edge server in the group. Then the contacted servers which have the content reply the request. This approach is similar to the routing protocol in CCN which will be introduced later.

As the nature of Internet evolved to streaming service, the traditional CDNs, which are optimized to delivery small sized web data as pictures and forms, faced scaling problem. The hybrid peer-assisted CDN architecture has emerged as a rescue. Various hybrid architectures for streaming service are studied in [CHR08, TSHP05, DXC06, MKH$^+$09]. In these architectures, an edge server acts not only as a streaming server offering video content but also a tracker server in P2P system. Users are usually organized in the mesh based fashion. The service of each video stream experiences three stages.

1. The initial stage begins when a video is pushed to a server. During the initial stage requests of users are served only by the edge server.

2. As the number of requesting peers increases, CDN server reaches its bandwidth limitation. Then the service turns to a peer assisted stage. The edge server selects a subset of peers holding the video to be the complementary suppliers.

3. After a large number of peers have downloaded the video, the contribution of supplying peers can satisfy the playback rate of requesting peers. Then the service enters a pure P2P stage for the video, and the edge server works only as a tracker of the original video.

Similar to the transformation from the pure CDN to the peer-assisted mode, recent evolutions of Internet again force CDN to reinvent itself. The ISP market reached denser and more complex levels of dependencies in peering with other networks. In the meantime, transit costs drastically decayed, especially through the development of direct peering between ISPs. Consequently, the CDN "market need" from the ISP side has at best stabilized. Facing direct peering between ISPs, Internet Transit Providers (ITPs) developed their own caching capacities, which dramatically lowered CDN prices to gain market shares. Moreover, the increasing traffic from CDN servers yields bottlenecks over the peering links toward CDN. All these realities draw the "scary future" of the current CDN architecture.

2.2.3 Information Centric Network

Besides the evolution technologies as P2P and CDN, some networking experts propose to redesign the Internet through clean-slate approaches as Information Centric Network (ICN) to tackle the increasing demand for highly scalable content. The ICN architectures function based on named data objects. They intend to shift the current host based connection to the content oriented communication. Relying on in-network caching, multi-party communication through replication and interaction models that decouple senders and receivers, the ICN architectures are supposed to achieve efficient and reliable distribution of content.

While the topic of ICN has attracted recently a lot of research efforts, the idea is hardly new. Over a decade ago, an ICN like design has been proposed by the TRIAD paper [CG00]. Then, the principle that the end-to-end connection for displaying web content should be moved to the delivery of named block data was reinforced by the 2002 IETF draft [Bac02]. However, these precursors utilized the existing DNS naming system with its inherent short comings and ignored other aspects of content delivery except the basic data transmission. Consequently, there was few works contributed to ICN in the following years. The first comprehensive and remarkable design of ICN appeared about five years later. The Data-Oriented Network Architecture (DONA) introduced the usage of self-certifying names and incorporating advanced cache functionality to address various other ICN issues [KCC+07]. But research communities was not aware of the importance of the ICN architecture since the global Internet traffic was only one-sixth as it is today [CIS]. As the traffic grows six times, and both the P2P and CDN become awkward to meet with the evolution of Internet service, the recent Content Centric Networking proposal injects vigor into the domain of ICN. Various projects as SAIL [Proc], 4WARD [Proa] and COMET [Prob] have devoted to the research on ICN and brought ICN into the networking research mainstream. As the most representative proposal of ICN, we give a brief introduction to CCN in the following paragraphs.

In CCN, communication is driven by the consumption of content. Communication links are replaced by the the anycast like transmission of two kinds of packet: *interest* and *data*. A user broadcasts its interest packet over all available connections. Any CCN node hosting the corresponding data can satisfy the interest by responding a data packet. Using the hierarchical content name, CCN nodes are able to redirect the interest toward the content source and send back the data via the reverse path of the interest[JST+09]. The proposed hierarchical structure of content name is similar to the structure of a URL or an IP address. A typical example is a name as */parc.com/videos/WidgetA.mpg/_v<timestamp>/_s3*. There are three main data structures in a CCN router:

- *Forwarding Information Base* (FIB) is used to forward Interest packets toward potential data source.
- *Content Store* (CS) works as a cache of content applying the Least Recently Used (LRU) replacement policies.
- *Pending Interest Table* (PIT) keeps track of Interests forwarded upstream toward content so that returned data can be sent downstream to its requester.

The rationale behind the CCN routing is inspired by the longest prefix match in IP network. But the operation is more complicated than the pure forwarding process in IP. Two different types of packet are treated in distinct ways. When an interest arrives at a CCN router, a longest-match lookup executes following the priority that CS match is preferred over a PIT match, and a PIT match is preferred over a FIB match. If a CS match is found, then the corresponding data in the cache should be directly sent through the interface that received the interest, and the interest is consumed. If there is a PIT match, the interest arrival interface is added to the PIT requesting list and the interest is discarded. Otherwise, if it is a FIB match, the interest is sent out all the faces, which can potentially offer the corresponding data. Then a new PIT entry is created based on the interest and its arrival interface. If no match is found, the interest is destroyed. The treatment of a data packet is relatively simple. Only when there is a PIT match, the data packet is sent out over all the faces in the requesting list and stored in CS. Otherwise, the packet is eliminated, since it is either duplicated or unsolicited.

Because CCN use the same forwarding model as IP network, any routing protocol that works well for IP should also have a good performance in CCN. Therefore current Link-state routing protocols such as IS-IS or OSPF can well satisfy CCN intra-domain routing. When a CCN router received an announcement from a provider saying that the provider can offer data with a certain prefix, the router installs a local CCN FIB entry for the prefix pointing at the interface where it heard the announcement, and packages the prefix into link state advertisement (LSA) and floods the LSA to all nodes. When another CCN router receives the LSA for the first time, it creates a CCN FIB entry for the prefix pointing at the original router. Then all the interests via this router for the prefix can be routed to the original router. The same scheme can be implemented on inter-domain level by BGP.

CCN greatly improves the network efficiency specially when a popular video is requested by a large number of sinks. For instance, a popular YouTube video will traverse the link between *youtube.com* and user's ISP millions of times in current network. But in CCN, just several transmissions were necessary since the video is stored in the closest CCN routers to users due to the in-network cache. Moreover, CCN is intrinsically against the attacks toward communication links since links do not exist in CCN. Because there is no single source and destination, it is also difficult to attack a particular object. The network is secure as long as the data content is well encrypted. As a result, CCN retains the simplicity and scalability of IP but can offer better security and delivery efficiency.

However, the CCN is not a panacea. On one hand, a naive, purely academic, vision of the evolution of Internet, that does not consider the main actors and their business models, may not be realistic. On the other hand, the restricted storage capacity on cache nodes may prevent the ICN to obtain the ideal performance in case of offering large-scale services as high quality streaming delivery. Advanced caching technologies implemented in Telco-CDN may compensate the short comings yielded by the limited capacity.

2.3 Theoretical Related Work

2.3.1 Facility Location Problem

The two problems that we address in our work (application component and content placement in Telco-CDN) are closely related to two well-known families of theoretical problems: Facility Location Problems (FLP) and κ-median problems (κMP). We detail both families in the following.

In the general (uncapacitated) FLP, we have a set of facilities and a set of clients. An opening cost is associated with every facility. Each client has an access cost to retrieve the service deployed at any facility. The objective is to open a subset of the facilities and to satisfy all demands of the clients so that the total cost is minimized. This problem has been widely considered in networks to locate replica of web content into caches. Various constant-factor approximation algorithms apply on a metric space. The algorithmic concepts can be based on LP-rounding [STA97, CS98] or on primal-dual approach [JMS02, MYZ02]. The 1.52 approximation ratio that has been obtained in most recent studies is close to the lower bound proved in [GK99]. Many variants of FLP have been studied; most of them are NP- hard [OD98]. The variant being closest to our problems is the k-Product UFLP (k-PUFLP), where each client needs to be supplied with k distinct products. Few papers have dealt with this problem, though, and the focus in the available references is classic optimal and heuristic algorithms [KLR86, KL87, HL08]. The heuristic algorithms proposed in [KLR86, KL87] are, in fact, techniques to improve the computation of the integer program. Most recently, a first approximate solution has been proposed, where the solution is at most $\frac{3}{2}k - 1$ times the optimal one [LL08].

In κMP, we do not consider the opening cost of a facility but the number of opened facilities is restricted to κ. Various approximation algorithms using a variety of techniques have been proposed: rounding of linear programs [CGST99], primal-dual methods [JV99] and local search [KPR98, AGKP98]. In [KPR98] the authors proposed an approximation algorithm for κMP using at most $(3 + 5/\epsilon) \cdot \kappa$ facilities and giving a total cost that is at most $1 + \epsilon$ times the optimal solution of at most κ facilities. This result is improved in [AGKP98] by a $3 + 2/p$-approximation algorithm with a time complexity of $O(n^p)$, where p is the number of facilities that can be changed in one swap operation.

In [LSO+07], service allocation in the Internet is modeled as both κMP and FLP. The authors proposed an efficient distributed algorithm, and demonstrated that the algorithm performs close to optimality even when information gathering is restricted to a small local environment.

Another problem related to our resource allocation problem is the replication placement problem in P2P and CDN networks. Many replication techniques are reported in the last decade. Uniform, proportional and square root replications are proposed in [dU07, GB06, LCC+02]. These techniques distribute the replication either uniformly throughout the network or by the query rate of the objects. In [AGM03, LDP06, Ora01], the authors presented several schemes based on peer selection policy, *e.g.*, random selection of peers, and selecting peers on the path from the providing peer to the requesting peers. Some dynamic replication schemes are given

in [RRL+06, RIF02]. These replication techniques aim at finding a good trade-off between redundancy and cost-efficiency of resources. The dynamics of peers is a main issue in P2P networks. The redundancy is used to deal with peer churn, but excessive replication causes wastage of peer resources.

However, none of the previous propositions can be directly applied to our use cases. The main difference between k-CMSP and κMP is that we do not limit the number of sites (facilities) serving one component, but on the other hand each site (client) must access to k sites including itself to get all the service components. If we regard the allocation of each component as an independent κ-median problem, then deciding the value of κ for any component becomes another combinatorial problem since every site can hold only one component. Thus, the problem we consider cannot be just treated as several independent κMPs. Note also that, in both κMP and FLP, the demand of a node (on the single type of service) is given, whereas in k-CMSP, every node has to access all the service components except the one allocated to itself. Hence which components a site has to fetch remotely depends on the allocation decision at the site. The replication placement scheme is not suitable for solving k-CMSP since in our context, all the sites are under the control of a service provider, thus peer churn is not considered as a critical problem. Moreover, most of the replication schemes for P2P systems are only effective for locating popular objects. But in our system, each component in the application is uniformly accessed. Thus, the replication techniques in P2P systems are not appropriate to k-CMSP.

To some extent, the k-CMSP can be seen as a special case of k-PUFLP as it is possible to transform any instance of k-CMSP into an instance of k-PUFLP. Indeed, every site can be seen as both a facility and a client with a null access cost between them. The approximation algorithm we describe later is a variant of the algorithm presented in [LL08], but the approximation ratio in our case is better.

The k-PCFLP is also a variant of k-PUFLP where the service capability of the facility is limited by some constraints. While the simple CFLP are thoroughly studied [WW12, ZCS10, LZZ10, MCLTC10], few of them have treated the CFLP with multiple product case. The only reference related to k-PCFLP is provided in [PJ98]. In this work, the PLANWAR model is built to solve the multi-commodity, multi-plant capacitated facility location problem that seeks to locate a number of production plants and distribution centers so that operating cost for the distribution network are minimized. Our k-PCFLP differs from the PLANWAR in several aspects. We do not care about the choice of setting up plant and warehouse, which in our case are the CDN servers and dServers. All the servers are already deployed and should always be available to offer the best service to clients. In PLANWAR, the service capability of warehouse is restricted only by the storage capacity of commodities while in k-PCFLP we consider multiple constraints on servers including the cache size and the access bandwidth. Moreover, our k-PCFLP takes into account the client-commodity relationship, that is the preference of each client in different videos. Last but not the least, since we consider each client's preference and each server's constraints in the country or metropolitan level, the algorithm used to solve the problem should be able to handle a large volume of input data. Our genetic algorithm parallelized by MapReduce is particularly designed for dealing with large input data set which can not be treated by the centralized heuristic given in [PJ98].

The replication placement techniques in P2P and CDN systems could be potential solutions for our optimal chunk placement in Telco-CDN. However, our optimal placement strategy differs from existing techniques in two aspects. Firstly, the techniques used in P2P and CDN systems are not able to manage the underlying network traffic. The only works related to traffic engineering is the P4P proposal [XKSY08] and the subsequent ALTO works. These works aim to reduce inter-domain traffic based on the hints offered by ISPs. But these contributions can hardly apply to fine-grain intra-domain traffic management in our context. Secondly, replication techniques in P2P aim at finding a good trade-off between redundancy and cost efficiency of resources. The dynamics of peers is a main issue in P2P networks. The redundancy is used to deal with peer churn, but excessive replication causes wastage of peer resources. However, in Telco-CDN, all the repositories are under the control of an ISP, thus peer churn is not considered as a critical problem.

2.3.2 Caching Policies

From Facebook to Youtube, from the traditional content delivery technology as CDN to the clean-slate approach as CCN, caching is ubiquitous in Internet services nowadays. However, the theoretical analysis of a generic topology of caches is a challenging topic. The behavior of LRU has been studied in simple topologies like trees [CTW02] but this model is inapplicable for the irregular inter-connected structures of ISP networks. In [FRR12], the authors designed an approximation for a random replacement policy using an approach similar to the one described in [CTW02]. But again, the approach is limited to 2-level cache hierarchies, and cannot be easily extended to mesh or larger tree networks. The authors of [PCL+11] developed a mathematical model based on continuous time Markov chains for a single CR. They showed that the performance of a complete multi-cache system can be approximated by their single cache model. Unfortunately, their model cannot be used to study the performance of each individual cache. Rosenweig *et al.* [RKT10] presented a model that approximates miss ratios for any multi-cache topology when the caching policy is LRU. This model allows to estimate the performance of each single cache in a multi-cache system. It is based on the single cache model given in [DT90]. All these analytical models are however specifically designed for the LRU policy.

Our objective is to extend the multi-cache model proposed in [RKT10] by incorporating other caching policies. Among the lightweight caching policies that could be reasonably implemented in a CR, we focus on the well-known *Least Frequently Used* (LFU) policy and the spectrum of policies based on a trade-off between recency and frequency. We use the model of *Least Recently/Frequently Used* (LRFU) caching policy as it has been presented in [LCK+01]. We derive an analytical steady-state model for LRFU in order to expand the multi-cache approximation to any caching policy based on both recency and frequency.

While the analytical model studies the replacement decision problem, our cooperative caching protocol deals with the cache decision problem. In the early age of the web, the study of cooperative caching had become a major research area [Wan99]. The flurry of research yielded the standard protocol for web caching, *Internet Caching Protocol* (ICP). In ICP, whenever a local cache miss occurs at a cache proxy, it mul-

ticasts a query message to all the neighboring caches so that a remote cache hit may be discovered. The cooperation among cache proxies augments the overall cache hit ratio of the caching system, and ICP can be regarded as the earliest and simplest cooperative caching protocol.

Although ICP is widely deployed in proxy server and web caching systems as *Squid* [Squ] and *Harvest* [Har], its lack of scalability has been highlighted by other works related to cooperative caching protocols [KSB+99, VR98, FCAB98, MNR+98, RCG98]. Both the communication and CPU processing overhead caused by ICP augment quadratically with the increasing number of proxies, which seriously prevents the caching system from scaling up. Two main types of schemes were proposed to alleviate the rapid increment of overhead in ICP: flat hash-based systems illustrated in [KSB+99, VR98], and directory-based systems described in [FCAB98, MNR+98, RCG98].

In hash-based systems, the URL of each client request is uniformly hashed, and associated to one of the caching proxies. If the required object is not stored in the cache, the request is transferred to the server. A copy of the object is then returned to the client and to the cache. Due to the hash function, there is no redundant replica in the caching system. Moreover, the workload is well balanced among the caching proxies. The caching capability of the whole system raises as the number of proxies increases. In directory-based systems, each caching proxy maintains a directory that memorizes the existence of objects stored at each of the other proxies in the system. Upon request reception, a proxy first searches its own storage. If the required object is not found, instead of forwarding the request to a server, the proxy checks its directory and forwards the request to another proxy according to the result of its directory scan. The consistency between a directory and a real cache is achieved by periodical message exchange across proxies. A URL routing framework is introduced in [MNR+98] to help proxies to make efficient forwarding decisions. Multiple replicas are allowed in directory-based systems to reduce the transmission delay from proxy to client.

Both hash-based and directory-based schemes are expected to make cooperative caching system scalable, but these efforts seems to be infructuous according to the analysis presented in [BCF+99, WVS+99, GDS+03]. These reports concluded that any kind of cooperative caching system is useless when the population behind the edge proxies surpassed the size of a medium-sized city with about half million clients. Therefore, it makes no sense to expand a cooperative caching system to a highly scalable one. These studies severely frustrated the motivation on the research of cooperative caching systems; nevertheless, the ideas raised by the previous work are not unprofitable. The results obtained in [WVS+99] are based on static document caching, and do not apply to video streaming delivery services. Since the size of the video segments in orders larger than the old static text-based content, caching efficiency and cooperation among proxies have become attractive topics again.

Previous work (and some other uncited papers) cannot be directly transplanted to CCN, either because of their limited scalability or their complexity. The two typical cooperative strategies, directory-based and hash-based strategy, were applied to video delivery system, and their limitations were recently illustrated in [GS09]. In the first one, every cache needs to maintain a knowledge about all video *blocks* that are currently cached by other caches. In order to reduce redundancy, a block that is

stored by more than two caches has a larger probability to be evicted. The consistency of lists is ensured by message passing. However the overhead grows with the number of caches and the number of video blocks. In the second one, a hash function is used to assign video *clips* to caches, which are then responsible to maintain pointers to all caches storing their allocated clip. Then, every cache applies the LRU policy to eject redundant clips. Such a strategy implies a overhead that cannot be handled by a set of CRs.

Among all works related to cooperative caching, some strategies share similarities with ours. For example, every cache node is explicitly in charge of storing *certain* parts of a video stream in [CHW03]. The assignment is dynamically decided by each cache based on the network condition and storing ability of neighboring caches. Again, this approach generates extra computations, which are unacceptable for a running CCN router. In [DLL+11, DLLL12], the authors present cooperative caching mechanisms for inter-domain P2P and wireless networks respectively. The correctness of the mechanisms are ensured by frequent exchange of control messages between cache servers, which is not suitable in CCN. In general, these papers do not provide any theoretical caching analysis of multi-cache topologies. Previous works also include cooperative schemes where the goal is to choose the best origin server or proxy (for example [WMM02]). These works, which superpose underlying routing protocols, cannot be implemented on CCN because routing and server discovering are ensured by the CCN protocol itself. The same remark also applies to theoretical works about content placement in Internet.

Several studies on the multi-cache structure in CCN have recently emerged. The authors of [PCL+11] have developed a mathematical model for a single CR based on continuous time Markov-chain. They have shown that the performance of an entire multi-cache system can be approximated by their single cache model. Rosenweig *et al.* [RKT10] have presented a model that approximates miss ratios for any multi-cache topology when the caching policy is LRU. However, these work discussed merely the performance of a basic multi-cache system, where each cache works individually. A similar idea to ours is raised in [HCP11], where only selected nodes on the forwarding path are allowed to store the content. The selection is based on the centrality of each CR (*i.e.,* the number of times the CR is located on the shortest path between all peers of nodes in the network), which will skew the load balance. Furthermore, there is no interaction between CRs, thus, it is still a basic multi-cache scheme. Another recent work on cooperative caching for ICN is the WAVE system proposed by [CLP+12]. WAVE uses counters stored in data packets and CRs to indicate the popularity of video segments. Each CR decides for its next hop CR whether the next one should cache the segment, based on the counter. To avoid the scaling problem, the popularity counter is associated with video but not with segment. In other words, segments of one video have the same popularity, which is obviously not the reality. Many statistics show that beginning segments of a video is usually more popular than others. The mis-match between the popularity indicator and the granularity of stored content may prevent the fast distribution of popular segments.

2.4 Summary

This thesis will focus on the resource placement problem in various large scale services introduced in this chapter. We will first investigate the component allocation problem in Telco-CDN in chapter 3. In chapter 4, we will address the optimal video placement problem. The in-network cache decision and replacement decision problems in CCN will be discussed in chapter 5 and 6.

Chapter 3

Application Components Allocation in Telco-CDN

This chapter deals with the issue of resource allocation of video delivery application in Telco-CDN. Generally, the solution can be easily applied on the distribution of other large-scale applications. We address this problem from an optimization standpoint. We formulate the problem and prove its NP-completeness. Then, we develop a linear programming model to provide benchmark for approximation and heuristic algorithms.

3.1 Introduction

Cost efficiency is a key aspect in deploying distributed service in networks within decentralized service delivery architectures. We consider the resource allocation in Telco-CDN, where applications are partitioned into k components hosted by n sites. Each site will *uniformly* access the $k-1$ components that they do not have. The aim is to reduce the overall *network cost* for accessing the remote components.

The network cost for any pair of sites i and j is a generic notion that may include the number of Autonomous Systems traversed by a packet from i to j, the round-trip delay time or the ecological impact implied by the load of all routers on the route. For convenience, the cost for a pair of sites is also referred to as the distance between them. The network cost for a given site i corresponds to the sum of the costs between i and the $k-1$ sites that are closest in cost/distance and host collectively the $k-1$ remote components. The goal is to minimize the total of these costs over the sites. We will refer to this problem as the $k-$*Component Multi-Site Placement Problem* (k-CMSP).

Our contribution consists in the following investigations. We first provide a thorough theoretical analysis of this optimization problem. In particular we prove that, in the general case, k-CMSP is NP-complete. As an immediate consequence, two related optimization problems, where every site may be allocated several components, and where only a subset of all components are requested by each site are both NP-complete as well. Next, we formulate k-CMSP by means of an Integer Programming

model. Solving the model gives exact solutions of k-CMSP, which is doable in small networks. These two investigations provide a scientific background for extensive research on k-CMSP.

Our research is largely driven by practical considerations. Thus the next part of our investigation has consisted in designing several algorithms that can be easily implemented by a service provider. We propose four algorithms, one approximation algorithm and three heuristics. The first algorithm is a $(\frac{3}{2}k - \frac{5}{2})$-approximation algorithm that is inspired by a recent work on a variant of the Facility Location Problem [LL08]. Our algorithm ensures that resulting total cost is no more than $\frac{3}{2}k - \frac{5}{2}$ times the optimal one. Our second algorithm, theoretically elegant but complex, originates from domination theory in graphs. The algorithm considers the nearest neighbor graph linking the closest sites together, followed by partitioning this graph into k distinct dominating sets, each being the sites allocated the same component. In order to obtain fast a deterministic graph partition, we use a technique performing graph augmentation. The heuristic gives a solution in which each site is likely to find all components among its nearest neighbors. The third algorithm, being intuitive and far simpler, considers also the nearest neighbor graph, and each site examines its two-hop neighbors to decide its own component allocation. Finally, the last heuristic aims to build a fair solution where no site is outrageously far from all components. The idea is to rank the total cost for each site and consider the most disadvantaged site first.

We present a generic set of simulations. For small networks, we report performance evaluation of the algorithms using the global optimum computed via the integer programming model. For large networks, the algorithms are compared to each other, and vis-a-vis a random allocation strategy. The main insight is that the intuitive heuristic appears to have near-optimal performances for k-CMSP, whereas the ranking heuristic provides convincing results in terms of fairness. Moreover, the very basic random allocation that is suggested in current studies is largely outperformed. These results show that implementing quick algorithms may provide a significant improvement in the service delivery performance metric of k-CMSP; the improvement can be translated into less latency for clients and less cost for network operators.

The rest of the chapter is arraged as follows. Section 3.2 analyzes the complexity of k-CMSP and builds the integer programming model. In section 3.3 we present the first approximation algorithm and its proof. From practical point of view, section 3.4 proposes several heuristic algorithms to quickly solve the problem. The set of simulations is descibed in section 3.5. We finally outline future research directions that will follow up the theoretical investigations and the experimental results of this work in section 3.6.

3.2 Problem Analysis

In this part, we provide an analysis of k-CMSP, including a proof showing that the corresponding decision problem is NP-complete. That is, unless $P = NP$, this optimization problem can not be reasonably solved for large instances (here for a large number of sites). Then, we develop an integer programming model. For small net-

works, the integer model can be used to find exact solutions of k-CMSP. This allows for gaining insights into the performance of other algorithms, particularly distributed algorithms, in terms of optimality.

As a preamble, we recall some definitions of domination theory that will be used throughout the chapter. Consider a graph G linking the system elements, the node set is $\mathcal{V}$. A set $V \subseteq \mathcal{V}$ is called a *dominating set* if, for every node $i \in \mathcal{V}$, i is either an element of V or is adjacent to at least one element of V. Thus, if the nodes in V (called dominators) are all allocated the same component, then the component becomes available within one hop for all sites. An extension of dominating set is a *domatic partition*, where the graph G is partitioned into several distinct dominating sets. If the dominating distance of a dominator is extended to s hops, we call the set of dominators an *s-dominating set*. Given a graph G and a positive integer K, the problem of determining if there exists a domatic partition or s-domatic partition of size K is NP-complete [GJ79, PP06].

3.2.1 NP-Completeness

In the following, we show that the decision problem associated to k-CMSP is NP-complete. We recall that n denotes the number of sites and k is the number of components.

k-Component Multi-Site Placement Problem

Instance : A positive integer $n \in \mathbb{N}^*$, a positive integer $k \in \{1, \cdots, n\}$, a distance function $d : \{1, \cdots, n\} \times \{1, \cdots, n\} \to \mathbb{R}^+$, where $d(i, i) = 0$ for any $i \in \{1, \ldots, n\}$, and a positive integer $R \in \mathbb{N}$.

Question : Is there an allocation $\varphi : \{1, \cdots, n\} \longrightarrow \{1, \cdots, k\}$ such that the sum of all distances to other components is less than or equal to R, that is :

$$\sum_{i=1}^{n} \sum_{c=1}^{k} \min \{d(i, j) : j \in \{1, \cdots, n\}, \varphi(j) = c\} \leq R$$

Theorem 1 *k-CMSP is NP-complete.*

Proof. Given an instance of k-CMSP and a labeling φ, verifying that this function is valid can be clearly done in polynomial time in the size of the problem, hence k-CMSP belongs to NP. We assume that $R' > 1$ allover the proof.

We reduce the Domatic Partition problem to k-CMSP. Given a graph $G = (V, E)$ and a positive integer $R' \in \mathbb{N}^*$ where R' is the number of sets of the domatic partition, let n be the number of vertices of G (and assume that $V = \{1, \cdots, n\}$), let $k = R'$, $d(i, j) = 1$ if $\{i, j\} \in E$ and $d(i, j) = R' \cdot n + 1$ otherwise, $d(i, i) = 0$, and $R = R' \cdot n$. Clearly the instance of the k-CMSP can be constructed in polynomial time in the size of the instance of the domatic partition. By construction, we obtain a special case of k-CMSP. If this special case is equivalent to the domatic partition instance (in terms of a yes/no answer), which is the bulk of the proof, then the general case of k-CMSP can not be easier than the NP-complete domatic partition problem. In the proof, we show that the two instances are indeed equivalent in the sense that there is a domatic partition of size R' (or k), if and only if the k-CMSP instance admits a valid mapping φ, i.e., the existence of a solution with total cost no more than R = R'n = kn.

For the forward implication, assume that G admits a domatic partition of size $k = R'$, for example $V_1, \cdots, V_k$. Let φ be the function such that for any $x \in V$, we have $x \in V_{\varphi(x)}$, we show that φ is also a valid function for the instance of k-CMSP. Indeed, for any $i \in V$, by definition of φ and the fact that we have $d(i,j) \in \{1, R' \cdot n+1\}$ for $i \neq j$, we obtain that for any $c \in \{1, \cdots, k\} \setminus \varphi(i)$, $\min\{d(i,l) : l \in \{1, \cdots, n\}$, $\varphi(l) = c\} = 1$. As a result,

$$\sum_{i=1}^{n} \sum_{\substack{c=1 \\ c \neq \varphi(i)}}^{k} \min_{l \in \{1,\cdots,n\}} \{d(i,l) : \varphi(l) = c\} = n \cdot (k-1) \leq R.$$

Now, as $d(i,i) = 0$, φ is a valid allocation for our problem.

For the backward implication, assume that φ is a valid function for the constructed instance of k-CMSP, and assume, by contradiction, that there does not exist a domatic partition of size k in G. This implies that there exists $i' \in V$ and $c' \in \{1, \cdots, k\} \setminus \{\varphi(i')\}$ such that $\varphi^{-1}(c') \cap N_G(i') = \emptyset$, where $N_G(i')$ denotes the neighborhood of i' in G. Thus, by definition of $d(i', j)$, we obtain that $\min\{d(i', l) : l \in \{1, \cdots, n\}, \varphi(l) = c'\} = R' \cdot n + 1$, which is strictly greater than R and hence $\sum_{i=1}^{n} \sum_{c=1, c \neq \varphi(i)}^{k} \min_{l \in \{1,\cdots,n\}} \{d(i,l) : \varphi(l) = c\} = n \cdot (k-1) \leq R' \cdot n + 1$ which is strictly greater than R and hence contradicts the definition of a valid allocation. □

As an immediate consequence of Theorem 1, the more general problem, where each server can store l components for any integer $l \geq 1$, is also NP-complete, because k-CMSP corresponds to $l = 1$. A second observation is that the case where each client has to retrieve k' components with $k' \leq k$ is NP-complete as well.

3.2.2 Integer Programming

In order to solve k-CMSP, we first explore the domain of Integer Programming. k-CMSP can be formulated in form of an integer linear model using two sets of binary variables.

$$x_{ic} = \begin{cases} 1 & \text{if component } c \text{ is allocated at site } i \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij}^{c} = \begin{cases} 1 & \text{if } i \text{ obtains component } c \text{ from } j \\ 0 & \text{otherwise} \end{cases}$$

Problem k-CMSP can be modeled as follows.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{c\in\mathcal{C}}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} d(i,j)y_{ij}^{c} \\
\text{Subject to} \quad & \sum_{c\in\mathcal{C}} x_{ic} = 1, && \forall i \in \mathcal{V} && (3.1)\\
& \sum_{j\neq i} y_{ij}^{c} = 1 - x_{ic}, && \forall i \in \mathcal{V}, \forall c \in \mathcal{C} && (3.2)\\
& y_{ij}^{c} \leq x_{jc}, && \forall i,j \in \mathcal{V}, \forall c \in \mathcal{C} && (3.3)\\
& x_{ic} \in \{0,1\}, && \forall i \in \mathcal{V}, \forall c \in \mathcal{C} \\
& y_{ij}^{c} \in \{0,1\}, && \forall i,j \in \mathcal{V}, \forall c \in \mathcal{C}
\end{aligned}
$$

In the model, the set of components is noted $\mathcal{C}$. Thus $k = |\mathcal{C}|$. Constraint (4.3) ensures that every site will host exactly one component. Constraint (4.4) ensures that if site i does not obtain component c from any other site $j \neq i$, then i itself must have c. Conversely, if i is not allocated c, then i has to get c from exactly one of the other sites. Hence (4.4) is an equality, and implies, together with (4.3), that each site has access to all the components in $\mathcal{C}$. Constraint (3.3) states that i can obtain a component from another site j, only if that component is located at j.

Let $n = |\mathcal{V}|$. In the integer model, the number of variables and constraints equal $nk + n^2k$ and $n + nk + n^2k$, respectively. Since each site may potentially host any component, the number of x-variables can not be reduced. Variables $y_{ij}^{c}, c \in \mathcal{C}$ can be excluded from model, if the distance $d(i,j)$ is so large so it can not be optimal to deliver component from j to i. This requires however some a priori knowledge on the overall cost. From the fairness perspective, it may be of relevance to introduce a bound B_i on the service delivery time at each site i. The bound B_i constrains the distance to any site from which a component is delivered. To the integer model, the effect of the bound is size reduction, as all y-variables with $d(i,j) > B_i$ can be eliminated from consideration. On the other hand, too stringent bounds will lead to infeasibility. Examining the model constraints, the size of constraint (4.3) can be reduced by a factor of n, because it can be replaced by $\sum_{i\in\mathcal{V}} y_{ij}^{c} \leq n x_{jc}, \forall j, c$. From a computational efficiency standpoint, the reduction is not preferred because it significantly weakens the bound from the linear programming relaxation, causing the size of the tree that has to be explored by branch-and-bound to grow by magnitudes.

For some network structures, the optimum solution may introduce unbalanced workload at the sites. Although load balancing is not within the scope of the current work, we remark that additional constraints can by deployed to address this aspect. For example, we can add the following constraint:

$$\sum_{i\in\mathcal{V}} y_{ij}^{c} \leq L, \forall j, c.$$

By construction, the constraint imposes an upper bound L that restricts the number of sites served by a give site, in order to prevent site congestion.

3.3 Approximation Algorithm

Given an instance of k-CMSP, we first introduce a method to find a fractional optimal solution for the linear program of the integer model. We connect every site with its $k-1$ nearest sites to form a $(k-1)$-nearest graph G_k. We denote the set of the $k-1$ nearest sites of node i as $N_{G_k}(i) = \{j_1, j_2, \ldots, j_{k-1}\}$ and $N_{G_k}[i] = N_{G_k}(i) \cup \{i\}$. We form the fractional optimal solution S by setting, for all i in $\mathcal{V}$ and all c in $\mathcal{C}$, the value of x_{ic} to $\frac{1}{k}$, and, for all j in $N_{G_k}[i]$, the value of y^c_{ij} to $\frac{1}{k}$ too.

Lemma 1 *Solution S is a fractional optimal solution to k-CMSP.*

Proof. It is obvious that S satisfies all the constraints in the model in Section 3.2.2, except the integrality requirement of the variables. Moreover, every node retrieves all the components from its $k-1$ nearest sites and itself. Based on this observation, it is easily realized that there is no (fractional) solution with lower total cost, and hence S is an optimal fractional solution to k-CMSP. □

We denote by $\bar{d}_i = \sum_{c=1}^{k} \sum_{j \in N_{G_k}[i]} d_{ij} \frac{1}{k}$ the cost of node i in the above fractional solution. For the approximation algorithm with output allocation φ (which corresponds to an integer solution to the model in the previous section), let $d_i = \sum_{c=1}^{k} \min\{d_{ij} : j \in \{1, \cdots, n\}, \varphi(j) = c\}$, and $F(i)$ be the set of nodes serving i, i.e., i collects the $k-1$ absent components from the nodes in $F(i)$ in solution φ.

The algorithm contains two phases. Two sets are created and processed in algorithm execution: V_o and V_s. For all $i \in V_o$, the algorithm has determined in phase one the components for all nodes in $N_{G_k}[i]$. Set V_s contains nodes that have not been allocated any component in phase one. Both sets are empty when the algorithm starts. In the first phase, the algorithm iterates over the nodes. In one iteration, it selects the untreated node i with $\bar{d}_i = \min\{\bar{d}_i | i \in V\}$. If some of the neighbors in $N_{G_k}(i)$ have been allocated components, the algorithm checks the condition that *no pair of nodes in $N_{G_k}[i]$ hold the same component*. If $N_{G_k}[i]$ satisfies the condition, an allocation is performed to the remaining nodes in $N_{G_k}[i]$, such that no two nodes store the same component. As a result, all components appear in $N_{G_k}[i]$. Node i is then put in V_o. If $N_{G_k}[i]$ does not satisfy the condition, i is placed in V_s. The first phase terminates when all nodes have been processed. In the second phase, the algorithm considers nodes that are in V_s and have no component allocated. Each of these nodes is allocated the component that is farthest away in the current allocation. The algorithm is described in Algorithm 1.

We will now prove that this algorithm provides a guaranteed approximation of the optimal allocation. The proof requires the following lemma.

Lemma 2 *Any $(k-1)$-nearest graph admits a 3-domatic partition of size k.*

Proof. In a 3-dominating partition, the nodes are divided into a number of subsets (k in our case), and every node can reach at least one node in each subset within 3 hops. We need to show here that, after the execution of Algorithm 1, every node can access all k components in at most 3 hops. If this holds, we can obviously view each set of nodes holding the same component as a 3-dominating set, therefore we would like to prove that Algorithm 1 produces a k-sized 3-domatic partition of G_k.

Algorithm 1: Approximation algorithm for $k-$CMSP

$V \leftarrow \mathcal{V}$
$V_o = \emptyset$
$V_s = \emptyset$
while $V \neq \emptyset$ **do**
    select i having the smallest $\bar{d}_i$ in V
    if no $j, j' \in N_{G_k}[i]$ hold the same component **then**
        for all nodes in $N_{G_k}[i]$ without component, allocate components such that no $j \in N_{G_k}[i]$ and $j' \in N_{G_k}[i]$ hold the same component
        $V \leftarrow V \setminus \{i\}, V_o \leftarrow V_o \cup \{i\}$
    else
        $V \leftarrow V \setminus \{i\}, V_s \leftarrow V_s \cup \{i\}$
for every $j \in V_s$ such that j holds no component **do**
    allocate the farthest component c to j

After the execution of the first phase of Algorithm 1 on G_k, a node is either in V_o, or in V_s. We know that every node in V_o can access the k components in only one hop. For nodes in V_s, we show the result by contradiction.

Let i be a node in V_s. Assume that i is selected in iteration t, and, after the execution of the algorithm, a component is absent within 3 hops of i. The assumption implies that there is no node in V_o being at two hops from i. Since i belongs to V_s, we know that, at iteration t, there are at least two nodes in $N_{G_k}[i]$, say j and j', that hold the same component. In the algorithm, the allocation of component during the first phase occurs only when a node is put in V_o. So, j and j' are either in V_o, or neighbors of a node in V_o. Together with the fact that once a node is included in V_o or V_s, it will never be moved out, we conclude that there must exist a node in V_o being less than two hops from i. Hence a contradiction and the lemma follows. □

Theorem 2 *For any $k \geq 3$, Algorithm 1 gives an integer solution with a total cost that is no more than $\frac{3}{2}k - \frac{5}{2}$ times that of the fractional optimal solution for k-CMSP.*

Proof. For every node i' in V_o, we know that $d_{i'} = \bar{d}_{i'}$. For a node i in V_s (by the end of phase one), there are two cases: 1) the component at i is assigned in processing another node in phase one, and i's component coincides with the component held by one of its $k-1$ nearest sites, 2) two nodes in $N_{G_k}(i)$ hold the same component. We will treat the two cases separately.

In the first case, the component at i has been assigned in treating some node $i' \in V_o$. We also know that $i \in N_{G_k}(i')$. If i is selected at iteration t, then i' is selected at iteration $t' < t$, so $\bar{d}_{i'} < \bar{d}_i$. Assume that the component at i is 1, and components at $j' \in N_{G_k}(i')$ are $c = \{2, \cdots, k\}$. Then the cost of i can be calculated as follows:

$$\sum_{j \in F(i)} d_{ij} \leq \sum_{j' \in N_{G_k}(i')} d_{ij'} = \sum_{c=2}^{k} d_{ij'_c} \leq \sum_{c=2}^{k} (d_{ii'} + d_{i'j'_c}) =$$

$$(k-1)d_{ii'} + \sum_{c=2}^{k} d_{i'j'_c} = (k-2)d_{i'i} + \sum_{c=1}^{k} d_{i'j'_c} = (k-2)d_{i'i} + \bar{d}_i$$

Since $i \in N_{G_k}(i')$ and $\bar{d_{i'}} < \bar{d}_i$, we obtain:

$$(k-2)\bar{d_{i'}} + \bar{d_{i'}} \leq (k-1)\bar{d}_i$$

In the second case, from the proof of Lemma 2, we know that there must be a node i' in V_o within 2 hops from i, and the inequality $\bar{d_{i'}} < \bar{d}_i$ holds. Assume that j_1 and j_2 are the two nodes that prevent i from entering V_o, and $d_{ij_1} \leq d_{ij_2}$. Without loss of generality, we can assign component 1 to j_1. If i is assigned a component in the second phase, then this component can not be component 1, as j_1 is among the nearest neighbors of i. According to the algorithm, we have $j_1 \in N_{G_k}(i) \cap N_{G_k}(i')$. When the placement of components is finished, node j_1 is included in $F(i)$, since i and j_1 hold different components. Then the cost of i can be calculated as follows:

$$\sum_{j \in F(i)} d_{ij} \leq \sum_{j' \in N_{G_k}(i')} d_{ij'} = \sum_{c=1}^{k-1} d_{ij'_c} \leq d_{ij_1} + \sum_{c=2}^{k-1} (d_{i'j'_c} + d_{i'j_1} + d_{ij_1}) =$$

$$\sum_{c=1}^{k-1} d_{i'j'_c} + (k-3)d_{i'j_1} + (k-1)d_{ij_1}$$

Since $j_1 \in N_{G_k}(i')$, we have $d_{i'j_1} \leq \bar{d_{i'}}$, and therefore $(k-3)d_{i'j_1} + \bar{d_{i'}}$ is lesser than or equal to $(k-2)\bar{d_{i'}}$. Moreover, since $d_{ij_1} < d_{ij_2}$, and both j_1 and j_2 belong to $N_{G_k}(i)$, we have:

$$\sum_{c=1}^{k-1} d_{i'j'_c} + (k-3)d_{i'j_1} + \frac{k-1}{2}(d_{ij_1} + d_{ij_2}) \leq$$

$$(k-2)\sum_{c=1}^{k-1} d_{i'j'_c} + \frac{k-1}{2}\sum_{c=1}^{k-1} d_{ij_c} =$$

$$(k-2)\bar{d_{i'}} + \frac{k-1}{2}\bar{d}_i \leq (\frac{3}{2}k - \frac{5}{2})\bar{d}_i$$

As $(k-1) \leq (\frac{3}{2}k - \frac{5}{2})$ for any k greater than 3, the solution returned by the algorithm has a performance ratio of $\frac{3}{2}k - \frac{5}{2}$ in relation to the fractional optimal solution to k-CMSP. □

In the execution of the approximation algorithm, every site i needs first determine $N_{G_k}(i)$. Let $n = |\mathcal{V}|$ as before, using the same method proposed in [EPY97], the determination of $N_{G_k}(i), \forall i \in \mathcal{V}$ can be finished in $n \log n$. Then we use merge sort to arrange $\bar{d}_i$ in an ascending order, and the process costs $n \log n$ computations. Finally, $\frac{k(k-1)}{2}$ comparisons is necessary to allocate the component on one site. Therefore, as k is a constant, the complexity of the algorithm is $\mathcal{O}(n \log n)$.

3.4 Heuristic Algorithms

We describe now three heuristic algorithms. Each of them is motivated by a specific rationale: domination theory in graph, intuition and distributed computation, and fairness consideration. The first two heuristics attempt to minimize the overall cost of k-CMSP. The third heuristic emphasizes on the largest site cost to achieve better fairness. Their respective performances are evaluated in Section 3.5. The main reason of using heuristic algorithms is that they can produce very fast solutions to large instances of NP-hard problems. Heuristics do not have a performance guarantee. Empirically, however, heuristics sometimes deliver better solutions than an approximation algorithm (see also Section 3.5). In addition, a heuristic may enable localized computation, which is a useful feature in distributed systems.

3.4.1 A Heuristic using Domatic Partition

One idea of constructing a heuristic algorithm for k-CMSP is to build a nearest-neighbors graph and compute a domatic partition of it, with the underlying motivation that a domatic partition is composed by a set of dominating sets. If all the nodes in a dominating set are allocated the same component, then the component becomes available to all nodes in one hop. Thus the ideal case is a successful domatic partitioning into k sets on a $k-1$-nearest neighbor graph. Yet, domatic partition of a general graph is NP-complete. However, exact solutions can be computed for several classes of graphs [FHKS02]. We focus in this work on one of these classes, namely the *interval graph*, because a linear-time algorithm is known for computing a domatic partition on any interval graph, and an interval graph is *domatically full*, *i.e.*, k domatic partitions can be found in an interval graph with a minimal degree equal to $k-1$. The idea is to *augment* a $(k-1)$-nearest graph by additional edges, so that it becomes an interval graph. This technique is called *interval completion*. Next, we compute the domatic partition of this interval-completed $(k-1)$-nearest graph.

A few recent papers have dealt with domatic partition in ad-hoc networks [MW05, PP06] and peer-to-peer networks [Dan08]. These are works related to our algorithm development, although our objective is different. In these references, a given graph is partitioned into a number of partitions equal to the minimal graph degree, and the solution approaches are based on variants of a well-known randomized approximation algorithm [FHKS02]. In our case, we partition a complete weighted graph into k partitions in respect of the weights.

Recall that $G_k = (\mathcal{V}, E_k)$ is the k-nearest graph associated with our distance function, *i.e.* an edge $\{u, v\} \in E_k$ means that either u belongs to the k nearest sites of v, or v belongs to the k nearest sites of u. Determining any domatic partition greater than three for k-nearest graphs in polynomial time is open. Our proposal is to transform G_k into an interval graph. A graph is an interval graph if, to each vertex, we can assign an interval of the real line such that there is an edge between two vertices if and only if their respective intervals have a non-empty intersection. For interval graphs, a domatic partition of size equal to the minimal graph degree plus one can be found in linear time in the size of the graph [RR89].

A graph is an interval graph if and only if there exists an *interval ordering* of its

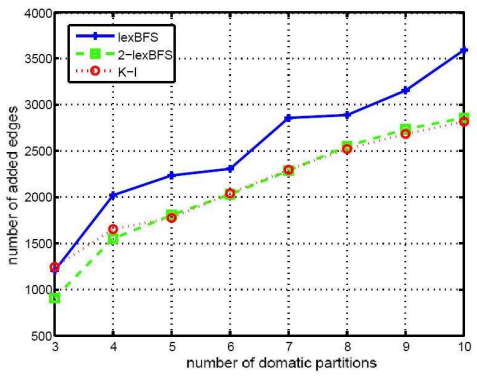


Figure 3.1: Impact of k on the number of added edges

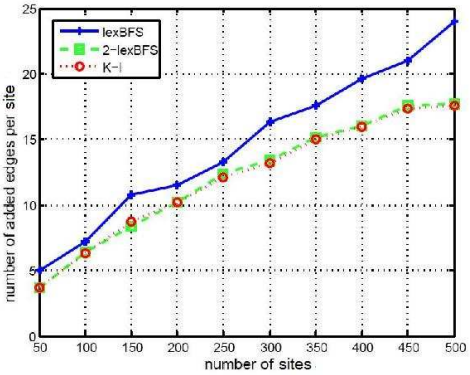


Figure 3.2: Impact of n on the average number of added edges.

vertices, *i.e.* a linear ordering $\sigma = (x_1, \cdots, x_n)$ of $\mathcal{V}$ such that if there is an edge between x_i and x_k with $i < k$, then for any j such that $i < j \leq k$ there is an edge between x_i and x_j. From a k-nearest graph $G_k = (\mathcal{V}, E_k)$ and an ordering σ of the vertices, we can define an interval graph $G_k^\sigma = (\mathcal{V}, F_k)$, where for any $1 \leq i < j \leq j' \leq n$, edge (x_i, x_j) belongs to F_k if edge $(x_i, x_{j'})$ belongs to E_k. We remark that all edges of G_k are also present in G_k^σ. Hence G_k^σ is obtained by augmenting G_k.

To use interval graph in our study, design of an interval ordering that results in interval completion and optimizes a given objective is of key importance. Yet, determining the minimal number of edges for an interval completion is NP-hard. In our study, we analyze three known algorithms for interval ordering. The first one, based on a *Lexicographic Breadth First Search* (Lex-BFS) [HMPV00], has a linear time complexity. The last vertex of this ordering can be chosen as an extremal one for an interval representation of an interval graph. The second algorithm consists of applying Lex-BFS two times (2-Lex-BFS) because the choice of the starting vertex has a strong impact on the effectiveness of Lex-BFS. Finally, we implement a recent algorithm, denoted K-I [ST09], which determines a *minimal interval completion*, that is, no subgraph of the generated interval graph is an interval graph. The K-I algorithm runs in $\mathcal{O}(n \cdot m)$ time with m being the number of edges in G_k.

Our algorithm consists in the following steps: (i) build the $k-1$ nearest graph G_{k-1}, (ii) determine a linear ordering σ of $\mathcal{V}$, (iii) compute the interval graph G_{k-1}^σ, (iv) compute a domatic partition of G_{k-1}^σ in time $\mathcal{O}(m)$ [RR89], and (v), if the number of the domatic partition is greater than k, merge the smallest partitions. Note that a k-nearest graph can be computed in $\mathcal{O}(n \log n)$ time [EPY97] and the number of edges m is bounded by $k * n$ in G_k, so the complexity of the entire algorithm is $\mathcal{O}(n \log n)$.

We have compared the performance of the three variants of the domatic heuristic, Lex-BFS, 2-Lex-BFS, and K-I, on a basic simulator where the instances are obtained by uniform distribution of sites in a two-dimensional Euclidean space. Intuitively, a small number of added edges in graph augmentation means that the domatic partition uses preferentially the edges of the k-nearest graph. Therefore the number of added edges is a relevant performance indicator.

In Fig. 3.1, instances of 250 sites are used to illustrate the number of added edges when the number of partitions k grows from 3 to 10. Not surprisingly, the number of added edges increases by graph density. Both 2-Lex-BFS and K-I outperform Lex-BFS algorithm by around 20%. To one's disappointment, the K-I algorithm has no noticeable advantage in performance over the linear-time 2-Lex-BFS algorithm. Fig. 3.2 shows how the average number of added edges per site for a 6-nearest graph varies over the number of sites. It can be seen that all algorithms produce high numbers of added edges, and this number increases by problem size. Both figures reveal that an interval completion substantially expands the k-nearest graph, and 2-Lex-BFS exhibits, with a linear-time complexity, a performance comparable to that of K-I. We therefore choose to use 2-Lex-BFS in the following experiments.

3.4.2 An Intuitive and Localized Heuristic

We observe that applying k-nearest graph on undirected edges may lead to poor solutions. See, for example, Figure 3.3 with 4 components and a 3-nearest graph. Assume that the left part of the graph is firstly allocated components. Vertex i is assigned c_4. Now we need to decide the component for j. It is easy to see that the overall cost is minimized by assigning c_1 to j, whereas the worst selection is c_4. If we consider the edges as undirected, node i is currently dominated by all components, and therefore a randomly chosen component will be allocated to j, potentially leading to the worst selection c_4. However, if we consider the directed graph instead, only the three nearest vertices of i are considered. In this case i is not dominated by c_1, which becomes the choice of allocation at j.

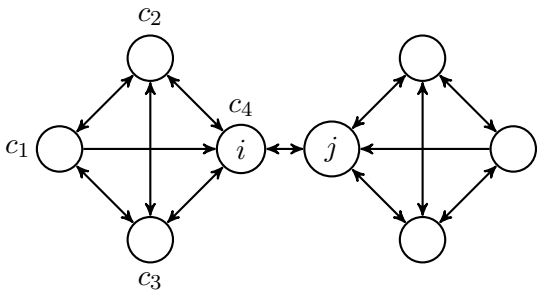


Figure 3.3: An example of a directed 3-nearest graph.

Before describing the algorithm, we remark that it needs an ordering for traversing all the sites in the network. In order to be consistent with the interval completion approach, we use 2-Lex-BFS to calculate the ordering.

The algorithm is composed by two phases. In the first phase, each site chooses its component based on the components of its one-hop neighbors, and tries to optimize the cost function in component selection. If the components cannot be distinguished by the cost function at a site, the choice becomes pending, and the site is put into a list to be processed later. In the second phase, the sites in the list choose the component allocation to maximize the benefits (*i.e.*, the cost saving) of themselves and their neighbors. We denote by, respectively, $N_{out}(i)$ and $N_{in}(i)$ the k nearest

sites of i and the sites that consider i as one of their k nearest sites. Note that a site regards itself to be one of the k nearest sites.

The first phase, performed by an individual site $i \in \mathcal{V}$, is depicted in Algorithm 2. Obviously, site i should avoid choosing a component that is already allocated to any of its nearest sites. The set containing these components is defined in line 1. In addition, site i should not pick a component that has been chosen by any site in $N_{in}(i)$, because these sites have i as a nearest site. Likewise, any component allocated to any nearest site of the sites in $N_{in}(i)$ should be avoided at i. These observations lead to the two sets of components defined in lines 2–3. If there are components remaining, one of them is randomly selected by i (line 6), otherwise the allocation of i is pending, meaning that the selection will be made in the next phase.

Algorithm 2: Intuitive Heuristic (site i) – first phase

1 $AllocC_{out} = \{\varphi(j) : j \in N_{out}(i)\}$
2 $AllocC_{in} = \{\varphi(j) : j \in N_{in}(i)\}$
3 $AllocC_{inout} = \{\varphi(j') : \exists j \in N_{in}(i), j' \in N_{out}(j)\}$
4 $PossibleC = \mathcal{C} \setminus (AllocC_{out} \cup AllocC_{in} \cup AllocC_{inout})$
5 **if** $|PossibleC| \geq 1$ **then**
6 randomly choose $c \in PossibleC$

Algorithm 3 shows the procedure used to select component at site i that is pending by the end of phase one. The idea is to evaluate the overall gain in terms of cost saving among all possible components. For compactness, we denote by $d(j|c)$ the cost for site j to obtain component c from the closest site currently hosting the component, *i.e.*, $d(j|c) = \min_{\{j \neq i : \varphi(j) = c\}} d(i, j)$. The computation of the gain for site j in $N_{in}(i)$, provided that site i picks component c, is the difference between $d(j|c)$ and $d(j, i)$. It should be remarked that this calculation requires site j to acquire $d(j|c)$. In a distributed system, the calculation will impose some signaling overhead.

Algorithm 3: Intuitive Heuristic (site i) – second phase

1 **for** $c \in \mathcal{C}$ **do**
2 $gain(c) = d(i|c) + \sum_{j \in N_{in}(i)} (d(j|c) - d(j, i))$
3 pick c with maximal $gain(c)$

Note that each site decides its own component based on information related to sites in two hops from itself. Hence, the algorithm satisfies our requirement that only local information is necessary for each site to select its component. This characteristic fits the characteristics of site-to-site and ad-hoc networks.

Since the heuristic can execute in a distributed manner, we examine the complexity per site. Every site needs to decide its $k-1$ nearest neighbors, which costs $\mathcal{O}(n \log n)$ comparisons. The complexity of the component allocation process in the first phase is $\mathcal{O}(k)$. In the second phase, a site i needs to execute a breadth first search (BFS) in two hops to determine the missing component of $N_{in}(i)$ and itself. Then $\mathcal{O}(n)$ computation is needed for i to choose its component. Therefore, the complexity for one site to run the algorithm is $\mathcal{O}(n \log n)$.

3.4.3 A Heuristic Addressing Fairness

Both previous heuristics exhibit the risk to promote the overall performance over fairness, by sacrificing some sites in order to reduce the total network cost. This may lead to great difference in the costs of sites. In a practical system, this unfairness has the risk to frustrate the users connected to the most poorly served sites. The heuristic we describe now looks for solutions based on the principle of max-min fairness, that is, we try to minimize the network cost of the most disadvantaged site with priority. Another motivation for this heuristic comes from the observation that a site being located close to $k-1$ other sites is probably close to many additional sites, so it is not affected by a sub-optimal component allocation in its surroundings, whereas the site that is far from other sites is more sensitive to the location decision.

The algorithm goes through n rounds. In each round, it determines the site that has not been allocated any component and is potentially the most disadvantaged one, measured in its distances to the $k-1$ nearest sites that may serve it. Formally, for a site v, we note S_v^k the set of sites having the potential to be selected by v for service delivery, and $d(S_v^k)$ the cost to access the service. This set should obviously contain k elements including v itself, that is, $\left|S_v^k\right| = k$ and $v \in S_v^k$. At the beginning of the round, any site $u \in S_v^k$ satisfies the condition that either $\varphi(u) = NULL$ or, if $\varphi(u)$ is not null, then there is no site $w \in S_v^k$ with $\varphi(w) = \varphi(u)$.

Once a site, say v, has been selected, we assign distinct components to all sites in S_v^k. Therefore, at the end of the round, sites in S_v^k have components allocated. The components at $S_v^k \setminus \{v\}$ are all different from each other. By considering preferentially the sites that seem to be the most disadvantaged, the algorithm tries to promote fairness in the allocation.

In Algorithm 4 we describe the procedure of ranking and allocation. At line 4, the algorithm calculates the potential cost of all the sites not having component allocated. At line 5, the algorithm chooses the site v_{max} with the highest potential cost. We determine the components which have not appeared in the neighbor set of v_{max} at line 8, then these missing components are allocated to the free sites in the neighbor set at line 10.

Algorithm 4: Fairness Heuristic

```
1  set V ← 𝒱
2  while V ≠ ∅ do
3      for v ∈ V do
4          determine S_v^k
5      determine v_max = argmax_{v∈1,...,n} d(S_v^k)
6      C = 𝒞 \ {c : c = φ(u), u ∈ S_{v_max}^k}
7      V' = {u | u ∈ S_{v_max}^k ∩ φ(u) = NULL}
8      for c ∈ C do
9          randomly choose u ∈ V'
10         set φ(u) = c
11         V' ← V' \ {u}
12     V ← V \ {v_max}
```

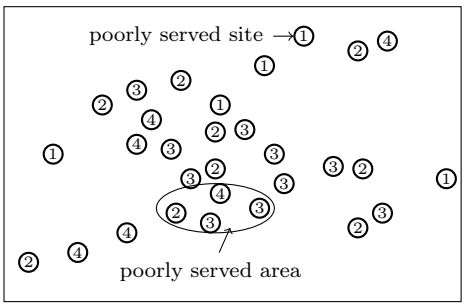


Figure 3.4: Random allocation.

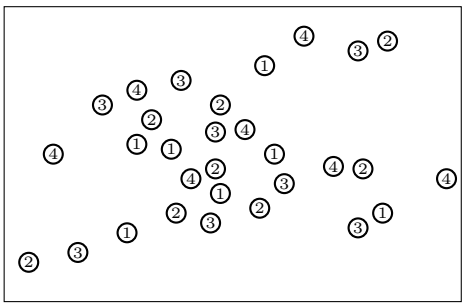


Figure 3.5: Intuitive heuristic.

A drawback of this algorithm is that it requires full knowledge of the network. However, a multi-site service provider typically has accurate information of the locations of the servers and can hence supply the algorithm with this input. This algorithm has the same complexity as the approximation algorithm $\mathcal{O}(n \log n)$.

3.5 Simulations

Many recent works, including some in standards organizations, have dealt with matching overlay networks and the Internet. In our simulation, we use as input the measurement results from one of these works. The underlying network is a matrix of latencies between all pairs of $2,500$ nodes from the Meridian project[1]. For each run, we choose randomly n nodes, then, for each pair of nodes, the network cost between them is the actual round-trip time (RTT) that has been measured. Motivated by the importance of delivery time, latency is a typical metric used by service provider to characterize performance. Moreover, it is one of the few metrics that a service provider can easily measure. As this work addresses the point of view of service provider, we adopt this metric for this set of simulations. Note however that other metrics could have been chosen, for example cross-domain traffic that is of relevance to network operators [CLS09a].

Previous works related to multi-site distributed services have suggested to use random strategies in allocating components to sites [VLM+09, NMM08]. This strategy is indeed easy to implement. In addition to comparing our algorithms among themselves, we will illustrate the gain of optimization in comparison to random allocation. Therefore, we display all results under the same form: we compute the average RTT cost for all sites in the allocation given by an algorithm, then we compare it to random allocation, of which the cost is normalized to 1.0. Hence, when a point is close to 1.0, it means that the gain of implementing the algorithm for this parameter setting is negligible. To reduce random effects, more than 20 different instances are tested for each comparison. In addition to the algorithms presented in the previous sections, we include the $3/2k - 1$ approximation algorithm, originally developed for k-PUFLP, in the simulations.

[1]Measurements have been done in May 2004. For more information, see `http://www.cs.cornell.edu/People/egs/meridian/`

Before showing the simulation results, we first give a small example to gain understanding of allocation solutions and highlight the benefit of our algorithms. In Fig. 3.4 and Fig. 3.5, 30 nodes are chosen from the Meridian matrix and displayed geographically. We decompose an application into 4 components and assign them to these sites. The number on each site represents the component that the site holds. A random allocation is shown in Fig. 3.4. In the figure, we find poorly served sites. For instance, the highlighted poorly served site is far from component 3, and the set of poorly served sites in the ellipse has long distance to component 1. Fig. 3.5 shows the location solution by the intuitive heuristic. It is apparent that the sites are closer to all the components in comparison to the random allocation.

3.5.1 Small Instances

We apply the Gurobi solver [Gur] to the integer programming model to shed light on the performance of the algorithms in respect of global optimum. In our computations, the time limit for calculating the result for each instance is 5 hours. As k-CMSP is NP-hard, the exponential-amount nature of the computation in solving the model means that exact solution is within reach for relatively small instances only. Nevertheless, the results give some indication on how far from optimality our algorithms and the random allocation scheme perform.

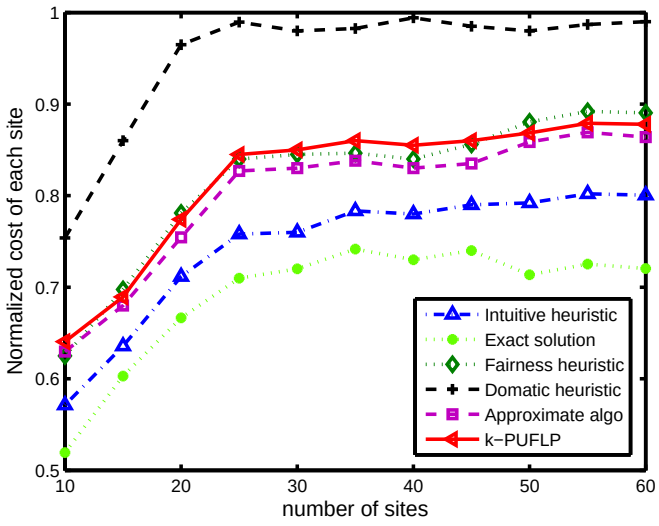


Figure 3.6: Performance evaluation for small instances: Impact of n on the average cost.

We show in Fig. 3.6 curves representing the normalized average costs for various numbers of sites. The number of components is fixed to be 6. This setting is typical for a large-scale service provider managing a set of servers. Note that for the instances with more than 45 sites, the Gurobi solver cannot reach the optimal solution in the time limit. The average optimality gap is reported in Table 3.1. We have also applied

Instance size	45 sites	50 sites	55 sites	60 sites	100 sites
Gap	3.96%	4.19%	4.39%	4.73%	11.32%

Table 3.1: Average optimality gap.

Gurobi to instances with 100 sites. However, the large gap indicates that the results are not useful for comparison.

For a small number of sites, all algorithms perform far better than the random one. It is possible to halve the average latency. As the number of sites increases, the improvement becomes less but remains considerable. In particular, the exact solution performs a quarter better than the random one. This, together with the fact that random allocation is currently a common approach in engineering practice, highlight the potential of gains by optimization. Except for the domatic heuristic, the performances of the algorithms seem to stabilize with an approximate gain of 20% over random allocation for $n \geq 25$.

A clear hierarchy of the algorithms' performance is revealed. As can be expected, no algorithm can perform as well as the exact one. We observe however that the intuitive heuristic exhibits performance that is remarkably close to optimum. On the contrary, the domatic heuristic is especially disappointing. Although this algorithm relies on theoretical concepts and results, it performs only slightly better than random allocation. Even for small instances, the number of added edges is so large that the allocation does not significantly leverage on k-nearest graph. The fairness heuristic and the approximate algorithm have almost identical performances. Moreover, our approximate algorithm performs slightly better than the algorithm for k-PUFLP because of the improvement in the approximation ratio.

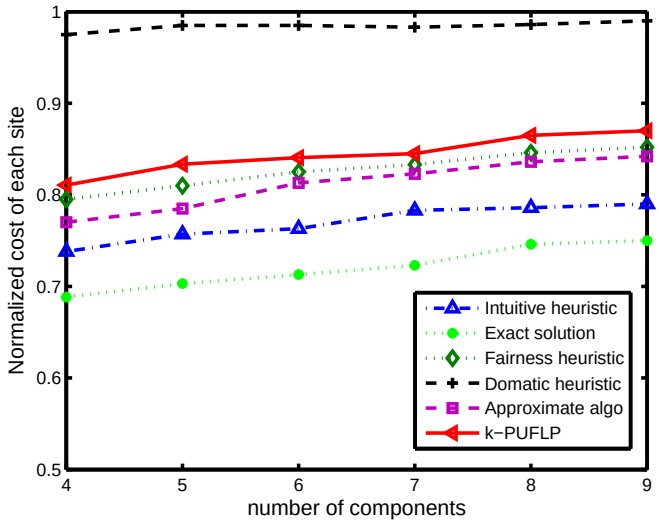


Figure 3.7: Performance evaluation for small instances: Impact of k on the average cost.

In Fig. 3.7, the number of sites is fixed to be 40, and the number of components varies. Besides the theoretical interest of measuring the impact of the number of components on algorithm performance, this scenario occurs typically for virtual games where several servers have to synchronize regularly.

The gain that we can see from the algorithms tends to slightly decrease when the number of components increases. This trend can be explained by the fact that the relative distance difference between the pth and the $(p+1)$th closest neighbors of a site tends to diminish when p increases. Indeed, consider concentric rings around a site. The area of a ring grows by its radius, thus the number of neighbors belonging to a large ring is bigger than that of a small one when the density is approximately constant. As we compute the average RTT, the benefits from a clever algorithm tends to decrease because there are many closely located neighboring sites.

The major result we extract from both Figures 3.6 and 3.7 is that our intuitive heuristic performs remarkably well. Indeed, it gives results that are almost equal to the optimal values. Hence the performance of this heuristic deserves future investigations in various application contexts. On the other hand, the fairness heuristic, aimed at reaching fairness among the sites, exhibits also good performance in the overall cost. It performs close to the approximation algorithm that has a performance guarantee for any metric distance function.

3.5.2 Large Instances

We compare now the algorithms to the random allocation scheme on larger instances. The main point we would like to show here is scalability, *i.e.*, whether and how the relative performance of the algorithms change in the numbers of sites and components.

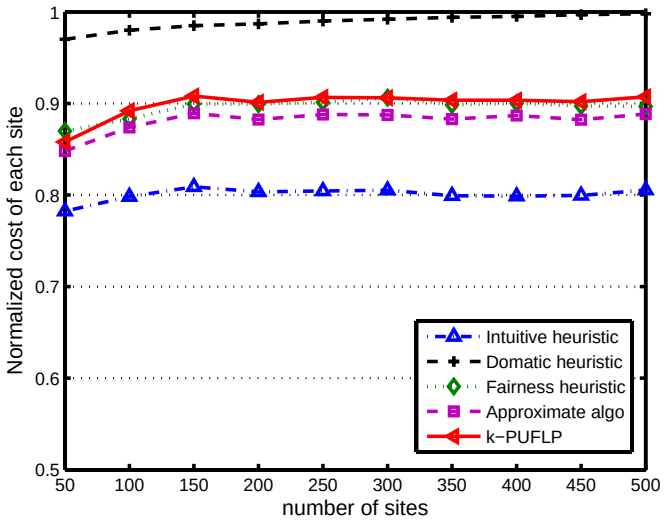


Figure 3.8: Impact of n on the average cost for large instances.

In Fig. 3.8, the number of sites grows while the number of components is fixed to

6. The results are consistent to what we have obtained for small instances. When the number of sites becomes large, the domatic heuristic performs almost as poorly as the random allocation scheme. Indeed, the augmented graph contains so many added edges that it looks like a random graph. On the contrary, none of the intuitive heuristic or the fairness heuristic suffers from any scaling effect. The former generates solutions that are 20% better than the random allocation. For a service provider, such an amount of gain in the delivery time is noteworthy.

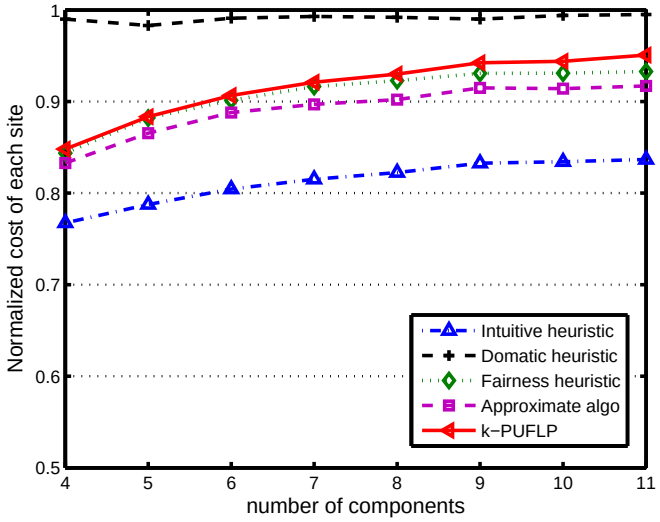


Figure 3.9: Impact of k on the average cost for large instances.

In Fig. 3.9, the number of components varies, while the number of sites is fixed to 250. Again, the figure verifies most of the observations made earlier. Note that the figure indicates, in addition to algorithm performance, the trade-off between two aspects in service distribution: the storage size of the sites, which directly links to the number of components, and the cost of collecting all components for delivering service. Here we emphasize that, for a few components, the gain is apparent, whereas it becomes less impressive for a service with many components. In this latter case, the random allocation, which is easier to implement, is preferable. In other words, it's not likely that our algorithms find their key applications in the case of boxes hosting small portions of a video.

3.5.3 Variance

Now we show the variance of cost for large instances. For clarity, we do not present the curves of all the algorithms, but the variance of the random allocation, the intuitive heuristic, and the fairness heuristic.

We find from Fig. 3.10 that the ratio of the maximum cost of sites over the average one increases gradually. The ratio changes from 1.8 to 2.2 for the random allocation, and from 1.75 to 2 for the intuitive heuristic. As it is expected, our fairness heuristic

gives the best performance in terms of small variance. The ratio for the fairness heuristic varies from 1.45 to 1.6. The result shows that both the intuitive heuristic and the fairness heuristic outperform the random allocation. We further analyze the benefit in Section 3.5.4. In fact, the ratio like 1.75 or 2 is acceptable for certain services as google search engine, since the service provider does not pay much attention on special cases, but the average response time of the engine. And the ratio about 1.5 is reasonable for more applications. Fig. 3.11 reveals the same results as those of Fig. 3.10, when the number of sites is fixed and the number of components varies.

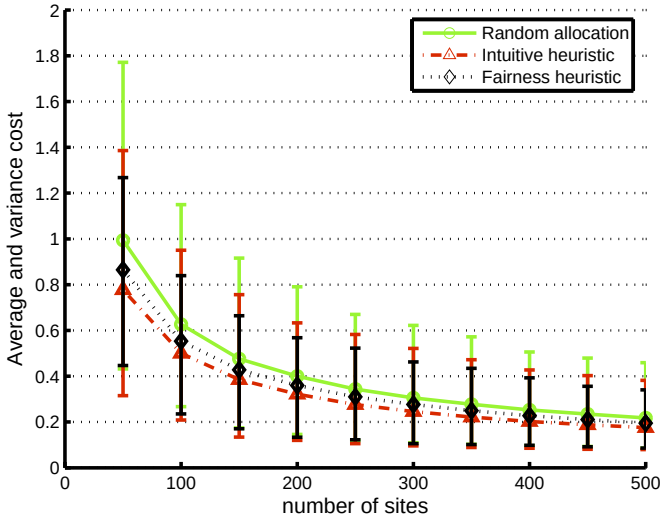


Figure 3.10: Impact of n on the average and variance cost for large instances.

Moreover, in Fig. 3.10, the cost decreases with the augmentation of site number since the density of sites increases and the distance between two sites becomes less. On the other side, the cost increases with the increment of component number in Fig. 3.11 because each site has more components to be accessed.

3.5.4 Fairness

Although the intuitive heuristic gives good performances in the overall cost, we observe that some sites are not fairly treated. In Fig. 3.12 and Fig. 3.13 we investigate the maximum cost generated by different algorithms. That is, for each algorithm and each configuration, we pick up the site with the maximum cost. Then we normalize its cost by the maximum cost produced by the random allocation.

We observe that, as can be expected, the largest cost among sites is better in the allocation by the fairness heuristic. In comparison to the random allocation, the gain is around a quarter. At the mean time, the benefit of the intuitive algorithm is around 15%. However, the performance of the approximate algorithm is close to the random allocation.

The relative performance between the algorithms in Fig. 3.12 are similar to what

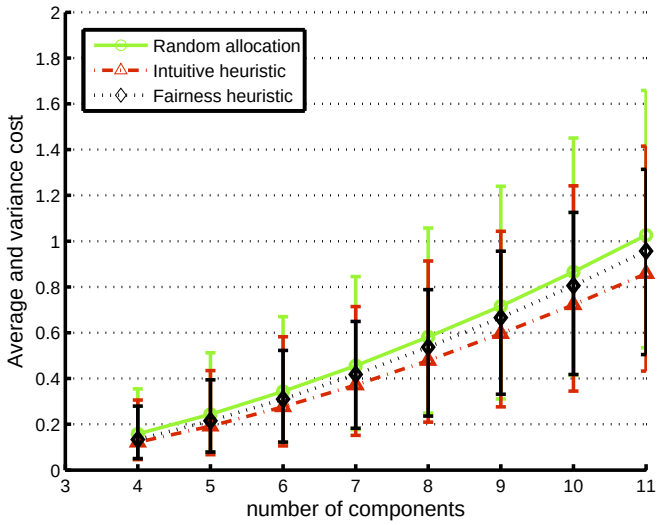


Figure 3.11: Impact of k on the average and variance cost for large instances.

has been observed earlier. In Fig. 3.13, the random allocation does not perform better when the number of components increases. We note that, in the instances, typically some nodes have larger distances values in their k-nearest neighbors than others. For these nodes, allocation of components should be made with priority to the nearest neighbors. Thus the benefit of optimization in the fairness metric remains when k grows.

In Fig. 3.14 and Fig. 3.15, we examine the standard deviation given by different algorithms. The same as before, we normalize the standard deviation of our algorithms by the standard deviation produced by the random allocation. Both figures indicate that the gain of the fairness heuristic is significant (around 40%). The intuitive and approximate algorithm also outperform the random allocation.

Unlike the fairness algorithm, whose performance is stable, the benefits of the intuitive and approximate algorithm decrease with the increment of sites. The gain of the intuitive heuristic stabilizes at 20%, while the performance of approximate algorithm becomes similar with the random allocation since the approximate algorithm give the priority to the sites with small cost. The Fig. 3.15 shows that the benefits of our algorithms do not vary with k, which is consistent with the result of Fig. 3.13.

3.6 Summary and Conclusion

We have studied theoretical and algorithmic aspects of an optimal network locality problem in the application context of locating and delivering distributed services in Telco-CDN architecture. The main characteristic of the problem is the distribution of a (possibly large) number of distinct service components. The objective of the optimization is the cost for service delivery. In addition to studying problem complexity, we have presented an integer programming model for the purpose of performance

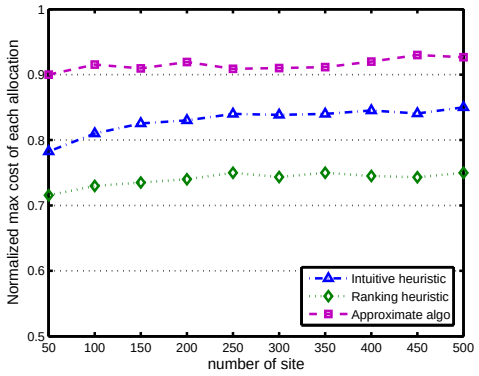


Figure 3.12: Impact of n on the maximum cost.

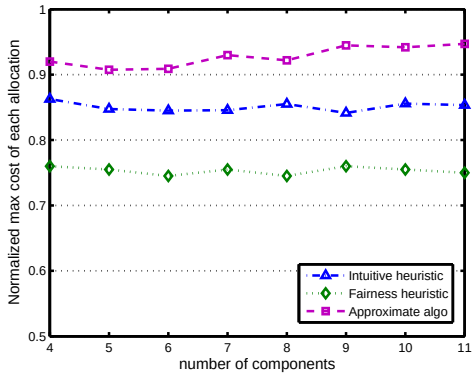


Figure 3.13: Impact of k on the maximum cost.

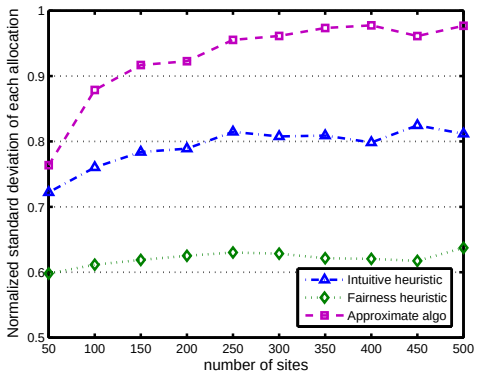


Figure 3.14: Impact of n on the fairness metric.

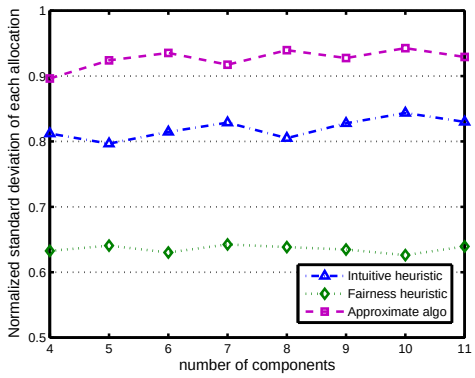


Figure 3.15: Impact of k on the fairness metric.

evaluation. One approximation algorithm and three heuristic algorithms are developed and studied numerically. Among them, the intuitive heuristic performs close to optimality for the tested networks where optimum can be computed via integer programming. For large networks this heuristic yields significantly better solutions than the random allocation. Moreover, the fairness heuristic achieves a good balance between the overall cost and the maximum one over sites.

There are several potential extensions of the work. One is the investigation of the impact of distance functions, the distribution of sites, and topology structures in various types of networks on the location solution. In view of the NP-completeness of the problem class, one issue is the design of approximation algorithms with better performance ratio, and the identification of classes of cost functions where exact solutions are within reach in polynomial time. Another extension is the inclusion of constraints modeling the sites' ability of serving each other in terms of the upstream communication capacity or the computational capacity, in order to avoid overload-

ing. Finally, allocating multiple components on one site and fault tolerance are also interesting extensions.

Chapter 4

Optimal Video Placement in Telco-CDN

This chapter leverages genetic algorithm and MapReduce framework to optimize the placement of video content in Telco-CDN. We first give a short introduction on the two techniques. Then, we formulate our optimization problem, and detail the design and implementation of the algorithm. Finally, extensive experimental results are shown as benchmarks for Telco-CDN constructors.

4.1 Introduction

To optimize the utilization of the Telco-CDN infrastructure necessities smart video placement that jointly consider network conditions, especially the infrastructure cost, and users' preference. Confronting the large data set containing the information of thousands of equipments, millions of end-users and exploding content volume, traditional centralized algorithms are not sufficiently powerful to produce satisfactory solutions. To overcome the difficulty, we propose the first genetic algorithm (GA) parallelized by MapReduce (MR) that efficiently tackles the big-data and finds the near optimal solution. We present brief revisions of GA and MR before detail the algorithm.

4.1.1 Background on Genetic Algorithms

A Genetic Algorithm mimics the events of the process of natural evolution. It consists of the following steps:

- *Encoding method* transforms a solution into its genetic representation called *individual*, which is usually a series of digital numbers.
- *Fitness function* evaluates the quality of each individual so that the best solutions survive the competition.
- *Selection*, *crossover* and *mutation* are three operations that allow to combine several valid individuals and to produce one so-called *offspring* from them.

The typical procedure of a GA starts with a randomly generated population of v individuals. Next, the algorithm computes the fitness value of each individual in the generation. Then, the three operators are repeated to produce v offspring to evolve the current generation into a new one. The production of generation executes iteratively until some convergence condition is reached. For more information about GA, please refer to [Mit96].

4.1.2 MapReduce Overview

MR frame work is inspired by *map* and *reduce* operations in functional languages, such as Lisp. Any algorithm that transform into these two operations can be parallelized automatically by MR. All data processed by MR are in the form of *key/value* pairs. A simple MR algorithm usually consists of two stages. In the first stage, map function extracts the data in each key/value pair and generate new key/value pairs as intermediate results, that is:

$$map :: (key_1, value_1) \rightarrow list(key_2, value_2)$$

Then all the intermediate results are merged and regrouped by keys. Reduce function is called once for each key with attached values and produce the final output as:

$$reduce :: (key_2, list(value_2)) \rightarrow list(value_3)$$

Detailed introductions of MR can be found in [LD10]. While MR is originally designed for handling basic data-intensive applications, recent studies highlight the potential of using MR to resolve large instances of optimization problems [JVB08]. Since MR allows users to distribute computing tasks without worrying about the difficulty of coordination, we benefit from this feature to accelerate the process of determining optimal video placement in Telco-CDN.

4.1.3 Our Contributions

This study makes two major contributions:

We present a practical, *quasi-optimal* algorithm for content placement in Telco-CDN. In other words, we show that a network operator is able to implement a very efficient push strategy if it judges that such implementation is worthwhile. Our idea here is to leverage a well-known *meta-heuristic*, namely genetic algorithm (GA), to reach a level of performances that is very close to optimal. In addition, we present an implementation of this GA on the *MapReduce* framework, which enables computation of large Telco-CDN instances on a small cluster of machines.

We present in a realistic evaluation the benefits one can expect from a push strategy for Telco-CDN. Our algorithm enables the comparison with traditional caching strategies. We collected real traces from a VoD service that should typically be hosted in a Telco-CDN. The data set consists of more than 700,000 requests from more than 20,000 end-users distributed over 13 geographical areas. We study a Telco-CDN deployment that is currently investigated by a major European network operator (Orange) and a simple but enlightening traffic management policy. Our main observation is that LRU caching performs as well as our push strategy

for the hit ratio. However, a push strategy significantly reduces the impact on the underlying infrastructure.

We organize the chapter as follows. We first formulate the optimal placement problem in section 4.2. Then, section 4.3 details the genetic algorithm for the joint optimal content placement that is achievable in Telco-CDN. Thereafter, we evaluate the performance of our optimal placement in section 4.4. Finally, we conclude this work in section 4.5.

4.2 Problem Formulation

In Telco-CDNs as well as in traditional CDN, the operator can decide either to push content in its servers, then to redirect the requests from end-users to the right server, or to assign end-users to a set of servers, then to implement a caching strategy in servers. In this section, we focus on the former strategy from an optimization perspective.

We consider here one given ISP network with a Telco-CDN that interacts with one PoP with a CDN. This latter has assigned a set of videos K ($l = |K|$). The ISP should allocate these videos to a set of repositories $j \in J$ ($m = |J|$). These videos can otherwise be retrieved from a virtual CDN PoP but this would be associated with monetary compensations. The set of end-users is noted I ($n = |I|$).

The objective of an ISP is at least twofold. On the one hand, an ISP has to preserve the quality of its network infrastructure because it is the core business of an ISP. On the other hand, an ISP that deploys a Telco-CDN should do its best to utilize its repositories, which might lead to generating more traffic within the network since requests may be re-directed to potentially far repositories. Combining two opposite objectives in the same problem calls for some simplifications and compromises. We thus introduce two critical parameters that have to be set by the network operator according to its network management strategies.

- **The assignment cost** a_{jk} of pushing a given content $k \in K$ in a given repository $j \in J$. When a content is already in the repository, this cost is null. The main difficulty is to set the cost to push a content. The cost of pushing a new content in a residential box is typically bigger than in a data-center, where it is close to null. Nevertheless assignment cost is commonly neglected with regards to the importance of other costs.

- **The service cost** e_{ij} of fetching a content from the repository $j \in J$ to a client $i \in I$. A network operator is used to setting a price that is proportional to the distance for every wired connection between two end-machines. It is part of the intelligence of the ISP to incorporate in such a simplistic vision some weight according to the quality of the connection and the amount of traffic this link should carry.

Our main idea is to integrate the *miss* (when a request cannot be fulfilled by the Telco-CDN) by overweighting the service cost of the link to the PoP. That is, when no repository stores a given content, a client has to utilize a PoP link having a weight that penalizes the overall service cost. Therefore an ISP can find a trade-off

between the infrastructure cost and the Telco-CDN benefit by adjusting the weight of the different service costs.

This problem is a k-Product Capacitated Facility Location Problem (k-PCFLP). It is NP-complete [LS11]. To our knowledge, neither efficient heuristic nor approximate algorithms have been studied for this variant of the FLP family. See in Table 6.1 the notations we use.

a_{jk}	assignment cost for repository j to retrieve video k from CDN.
e_{ij}	service cost for end-user i to obtain a video from repository j.
p_{ik}	recommendation that end-user i requests video k.
s_j	storage capacity of repository j.
b_j	maximum number of end-users that can be allocated to repository j.
x_{jk}	binary variable indicating whether video k is stored on repository j.
y_{ijk}	binary variable indicating whether end-user i obtain video k from repository j.

Table 4.1: Notations

The binary variable p_{ik} is the output of the predictions produced either by content providers or by CDNs based on the statistic of user behavior. Predictions of user behavior related to a given service has been spectacularly progressing. Moreover, the recommendation engines that are used by most video providers reinforce the quality of the predictions. The variable p_{ik} indicates whether the system should consider that it is highly probable that end-user $i \in I$ will request video $k \in K$ soon. In fact, for a given end-user i, a small subset of videos $K' \subset K$ verifies $p_{ik} = 1, \forall k \in K'$. These videos are typically the dozen of videos recommended by the video service for this user i.

The problem addresses both the placement of video into repositories and the assignment of end-users to repositories. We have two binary variables:

$$x_{jk} = \begin{cases} 1 & \text{if video } k \in K \text{ is stored on repository } j \in J, \\ 0 & \text{otherwise.} \end{cases}$$

which is the placement variable. Then, the redirection of end-users to repositories is captured by:

$$y_{ijk} = \begin{cases} 1 & \text{if end-user } i \text{ obtain video } k \text{ from repository } j, \\ 0 & \text{otherwise.} \end{cases}$$

where a request from user $i \in I$ for a video $k \in K$ should be redirected to server $j \in J$ when $y_{ijk} = 1$.

The storage capacity of each repository $j \in J$ is restricted by s_j video. With the recent development of rate-adaptive video streaming, the size of a video is now bigger than 20 Gbits because a server must store multiple representations of the same movie.

We consider a typical value b_j to express that a given repository $j \in J$ cannot be associated to more than b_j clients, which is a common dimensioning constraint even for services at the time scale of a day.

We formulate our k-PCFLP as follows:

$$\begin{aligned}
&\text{Minimize} \sum_J \sum_K a_{jk} x_{jk} + \sum_I \sum_J \sum_K e_{ij} p_{ik} y_{ijk} \\
&\text{subject to} \sum_J y_{ijk} = p_{ik}, \forall i \in I, \forall k \in K && (4.1) \\
&\qquad x_{jk} \geq p_{ik} \cdot y_{ijk}, \forall i \in I, \forall j \in J, \forall k \in K && (4.2) \\
&\qquad \sum_K x_{jk} \leq s_j, \forall j \in J && (4.3) \\
&\qquad \sum_I \sum_K p_{ik} \cdot y_{ijk} \leq b_j, \forall j \in J && (4.4) \\
&\qquad x_{jk} \in \{0,1\}, \forall j \in J, \forall k \in K \\
&\qquad y_{ijk} \in \{0,1\}, \forall i \in I, \forall j \in J, \forall k \in K
\end{aligned}$$

With constraint (4.1), the Telco-CDN must satisfy each end-user request. In the meantime, every recommended video request is treated by only one repository. Constraint (4.2) specifies that a repository can provide a video only if the video is placed on it. Finally, constraints (4.3) and (4.4) guarantee that the load on a repository does not exceed its service capabilities.

4.3 Algorithm Description

The number of repositories can be relatively small, but the number of videos in a typical VoD service, as well as the number of clients to serve, make the computation of an optimal solution to the problem described in section 4.2 impossible in practice. Our motivation is to design an algorithm that is quasi-optimal and that can be reasonably implemented by ISP for the management of their Telco-CDN. We use the term "quasi-optimal" to refer to *meta-heuristics*, which have shown in the past that they achieve optimality for most of the instances of a given problem. However, meta-heuristics commonly require a long computation time for large problem instances, which is our case here. To tackle this issue, some previous works have explored the design of meta-heuristics into massively parallel cloud architectures.

Since we are interested in implementation of quasi-optimal algorithm that allow computing large problem instances, we have implemented a meta-heuristic, namely Genetic Algorithm (GA), on the MapReduce framework. We have chosen GA because we expected GA to be both highly parallelizable and efficient for our problem. In the following, we will first recall the main idea behind GA, then we will present how we model our problem in GA, finally we will describe the MapReduce implementation of this GA.

4.3.1 Genetic Algorithm for Content Placement

The general idea of our GA is as follows. We generate individuals that correspond to a placement of videos to the repositories. Then we use a fitness function that computes an optimal assignment from end-users to repositories, subject to the constraints of k-PCFLP. Finally, we leverage the inner operations of the genetic algorithm to converge toward a better solution by modifying the best proposed solutions.

Encoding method

We create individuals that correspond to a given placement of videos to the repositories, with respect to the storage constraint. In other words, an individual corresponds to a set of $x_{jk}, \forall j \in J, \forall k \in K$. Our goal is to use a representation that enables easy implementation of the operations (selection, crossover, mutation). We chose a $(\sum_{j \in J} s_j)$-uple of integers, which represents the identifiers of the video on the different "storage slots" of repositories. Simply put, this encoding is a list of all $x_{jk}, \forall j \in J, \forall k \in K$ verifying $x_{jk} = 1$, which is sorted in ascending order of j's identifier.

For example, given an instance with $m = 2$ repositories, where the storage capacity of server j_1 (respectively j_2) is $s_{j_1} = 3$ (respectively $s_{j_2} = 2$). Let assume an individual $E = (e_1, \ldots, e_5) = (1, 3, 4, 2, 5)$. The three first entries e_1, e_2 and e_3 corresponds to the three videos in K that are allocated to repository j_1. Here repository j_1 stores videos k_1, k_3 and k_4. It also implies that $x_{j_1 k} = 0$ for all k in $K \setminus \{k_1, k_3, k_4\}$. The two last entries e_4, and e_5 are the videos allocated to repository j_2 (here k_2 and k_5).

Lemma 3 *Any individual respecting the proposed encoding obeys constraints 4.3 of the k-PCFLP.*

Proof. Since the overall length of an individual is exactly the sum of all storage capacities, there exists a m-decomposition of an individual where the length of each subset equals the storage capacities of each repository. □

Fitness function

The fitness fonction takes in input an individual E and should decide the value of each $y_{ijk}, \forall i \in I, \forall j \in J, \forall k \in K$. If the individual yields a possible solution (every recommendation of video for end-users is assigned to a repository storing this video), the fitness function outputs a final score, which enables competition among individuals.

The fitness function aims at assigning recommendation to repositories. It is another optimization problem related to a matching problem. We propose here a polynomial-time algorithm that computes the optimal solution to this matching problem. We transform the original problem into a *Minimum Cost Maximum Flow* (MCMF) problem as follows. Please refer to Figure 4.1 for an example.

We build a graph containing six classes of vertices. Two of them enable flow computation: (*i*) a virtual source and (*ii*) a virtual sinks. We also have the main targets of our computation (*iii*) end-user nodes and (*iv*) repositories. Then we add two vertices, which ensure that an end-user is linked to a repository only if this

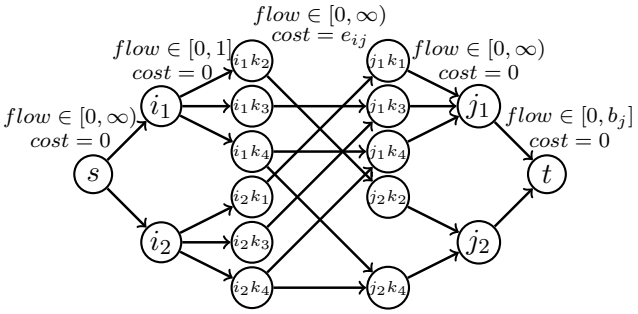


Figure 4.1: Example of MCMF

repository stores a video that is recommended to the said end-user: (v) user-video and (vi) repo-video. There exists a user-video vertex noted u_{ik} if and only if $p_{ik} = 1$. Similarly, there exists a repo-video vertex noted r_{jk} if and only if $x_{jk} = 1$ in the individual E that is evaluated by this fitness function.

Links in the graph should primarily make sure that assignment can only be done between an end-user and a repository "sharing" one video (the repository stores a video recommended to the user). We stipulate that a user-video u_{ik} has a link to a repo-video $r_{jk'}$ if and only if $k = k'$. Thus there is a path between the end-user i and the repository j.

We introduce then the flow on the edges. The main idea is that we force each end-user to fetch its video from only one repository, and we force each repository $j \in J$ to not serve more than b_j end-users. To achieve such result, the flow between end-user vertices and user-video vertices are restricted to one unit and the flow between repo-video and repositories vertices are limited by the bandwidth of the repositories.

Finally, the cost is set on the links between user-video and repo-video vertices. On a link between a user-video u_{ik} and a repo-video r_{jk}, the cost is set to e_{ij}, for any $k \in K$.

In Figure 4.1, end-user i_1 is recommended videos in $\{k_2, k_3, k_4\}$, and i_2 videos in $\{k_1, k_3, k_4\}$. The videos k_1 and k_3 are stored on server j_1, while k_2 is offered by j_2. Finally video k_4 is stored by both j_1 and j_2.

The fitness function is an algorithm in two rounds. First, we compute the maximum achievable flow $f(E)$ in the graph built from the individual E. We have:

$$f(E) = \begin{cases} \sum_{i \in I} \sum_{k \in K} p_{ik} & \Rightarrow \text{individual } E \text{ is feasible} \\ \text{otherwise} & \Rightarrow \text{individual } E \text{ is not feasible.} \end{cases}$$

If E is feasible, then we determine the solution that has the minimum cost overall. The fitness function returns this final cost as an output. Otherwise, the fitness function output ∞. Various methods can be used to obtain the optimal solution of MCMF. We implemented the push-relabel method proposed in [CG95].

Lemma 4 *Any individual (solution) with a fitness value different than ∞ satisfies constraints 4.1 and produces binary values for $y_{ijk}, \forall i, \forall j, \forall k$.*

Proof. Two cases can explain a violation of constraint 4.1. The first case is:

$$\sum_{j \in J} y_{ijk} > p_{ik}, \forall i \in I, \forall k \in K,$$

The value of y_{ijk} is given by the flow in the link between u_{ik} and r_{jk}. However, this flow is bounded by the flow prior to u_{ik}. Since there is only one link toward u_{ik}, and since the flow on this link is bounded by 1, the flow going out from u_{ik} can be either 0 or $1, \forall i \in I, \forall k \in K$. Therefore y_{ik} is binary and cannot be greater than p_{ik}.

The second case is:

$$\sum_{j \in J} y_{ijk} < p_{ik}, \forall i \in I, \forall k \in K.$$

In this case, u_{ik} is necessarily 0, which means that the flow beyond u_{ik} is null too. However, the flow is expected to be maximum with $f(E)$ equal to $\sum_{i \in I} \sum_{k \in K} p_{ik}$. This can be achieved if and only if the flow going from every $u_{ik}, \forall i \in I, \forall k \in K$ is equal to one. This contradicts. □

Lemma 5 *Any individual (solution) with a fitness value different than ∞ satisfies constraints 4.2.*

Proof. The satisfaction of constraint 4.2 is guaranteed by the construction of the flow graph. The violation of the constraint indicates that at least one unit of flow arrives at server node without passing any repo-video vertex. However, it is impossible since there is no direct link between user-video and repository vertices. □

Lemma 6 *Any individual (solution) with a fitness value different than ∞ satisfies constraints 4.4*

Proof. The left side of constraint 4.4 is the overall incoming flow at server node j. Since the flow going from each repository vertex j to the virtual sink is bounded by the bandwidth capacity b_j, the total flow in the cut prior to repo-video cannot be greater than b_j. □

Operations on individuals

With the above algorithms, we are able to encode individuals (solutions) and to evaluate their feasibility as well as their performances. We thus have the basic elements of a genetic algorithm. Now, we present how to generate individuals. Firstly, we show our method to produce one offspring from two individuals. Secondly, we describe our approach for the initial generation of individuals.

The three basic operations for offspring generation are selection, crossover, and mutation. For these operations, the individual is treated per repository. In other words, we extract from the individual encoding of each parent the storage of each repository.

1. The *selection* consists in ensuring that videos that are stored in both parents still appear in the offspring. Such selection process should not be automatic, though. Commonly in GA, the selection process is executed under probability, with regard to a given threshold.

2. The *crossover* operation consists in selecting one video for any free storage spot after the selection operation. Here, videos are randomly picked from the union of the videos stored by both parents.

3. The *mutation* adds some randomness to the process. After the crossover operation, each video has a very low probability (for example $p_{mut} = 0.001$) to mutate to another video that does not exist on the server.

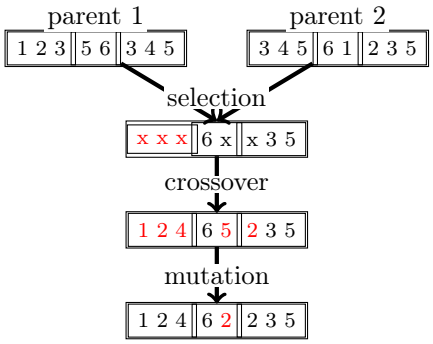


Figure 4.2: Creation of offspring: an example with three repositories, having respectively 3, 2 and 3 storage capacities

We give an example in Figure 4.2. We take two valid individuals in inputs. The selection is done for repositories 2 and 3, but the random pick makes that repository 1 should be generated from scratch (although video 3 is shared by both individuals and would have been kept). For the crossover, the free spots has filled with videos that are randomly picked in the union of sets of both individuals. Finally, the mutation produces that one video is mutated into another, although none of both parents stored it on this repository.

Lemma 7 *The individual produced by the operations from several individuals verifies Lemma 3.*

Proof . In each operation that produces or changes an individual, we prevent duplication of one video to several slots in the same repository. In particular, in the selection operation, we pick videos that are shared among individuals, and in the crossover operation, we select videos in the union of all videos from parents. □

The efficiency of a genetic algorithm depends also on *initial generation*, which should be at least extensible to the complete search space, otherwise, the GA may end without finding any feasible solution. Moreover, an initial generation close to the optimal solution can greatly save multiple iterations of the GA. Our method to endow the initial generation with good quality comprises two steps. Firstly, we

randomly choose l videos in each individual to represent the l distinct videos. Then, the number of replicas of a video is decided by the total storage capacity of servers and the popularity of the video. Specifically, the number of replicas for a given video $k \in K$ is set to:

$$\max(\left\lceil (\sum_{j \in J} s_j - l) \cdot g(k) \right\rceil, (m-1)),$$

where $g()$ is the probability density function of the video popularity distribution. Then, we determine the location for each replica so that two replicas of one video should never exist on the same server.

Overall description

We implement the elitist **generation revolution** (replacement) strategy to implement our GA. We first create an initial population of v individuals. Then, we produce v offsprings from this population. Out of the $2 \times v$ individuals (v parents and v offsprings), we select the v individuals that have the best fitness functions. If none of them is an offspring, the algorithm terminates, otherwise it reiterates this process. The best individual is eventually chosen. We give the pseudocode of our GA in Algorithm 5.

Theorem 3 *Any optimal solution of the MCMF instance with a fitness value less than ∞ is also a sub-optimal solution for k-PCFLP with fixed x_{jk} values.*

Proof. From lemma 3, 4 and 7, we conclude that any solution with a fitness value less than ∞ satisfies all constraints in k-PCFLP and binary attribute of two variables. Therefore, it is at least a feasible solution. Moreover, the only cost introduced in MCMF is the transmission cost from user-video node to repo-video node. Since the unit of flow from upr_{ik} to spr_{jk} corresponds to $y_{ijk} = 1$, the total cost of the MCMF is $\sum_I \sum_J \sum_K e_{ij} p_{ik} y_{ijk}$, which is exactly the unfixed part of the objective function. Thus, the optimal solution of MCMF is also the sub-optimal solution for k-PCFLP. □

4.3.2 Parallelizing GA by MapReduce

We are interested in using MapReduce (MR) because of the huge search space and large data set yielded by our k-PCFLP instance. Readers can refer to [KK10] for a tutorial on parallel genetic algorithm (PGA). We implemented the *dynamic demes* model in our parallelization since it fits better the structure of MR. In this model, the whole population is treated as a single collection of individuals during the evolution. After a new generation is produced in each iteration, the first task of the PGA is to dynamically reorganize demes, which matches the mapping phase in MR. Other operators are independently applied on each deme by reducers. At the beginning of each iteration, the mapper randomly regroups the entire population into r subpopulations (demes), where r is also the number of reducers in the MR system. Each reducer takes care of v/r individuals, and executes GA operators independently. Each reducer produces also v/r offspring, so that v offspring are produced by the whole

Algorithm 5: Genetic Algorithm for k-PCFLP

```
generate initial population G_0
continue = true; t = 0
while continue and t < threshold do
    continue = false; t = t + 1
    newPopulation= G_{t-1}
    minFitness= min{fitness(e) : ∀e ∈ G_t}
    for 1 to v do
        offspring ← Mut(Crossover(Select(G_t)))
        if fitness(offspring) <minFitness then
            continue = true
        add offspring to newPopulation
    sort newPopulation according to fitness
    G_t = the v first individuals in sorted newPopulation
return the first individual in G_t
```

system. When an offspring is produced, its fitness score is immediately calculated. Only qualified offspring are output by reducers. When all the reducers finish producing offspring, and there is no output, the algorithm stops. Note that the evaluation of the initial population cannot be integrated into the main loop of the algorithm, we illustrate the MR algorithm for the two parts in Algorithm 6 and 7.

MR tackles Initial Population

The individuals in the initial population are pre-generated outside of the MR algorithm, so Algorithm 6 does not contain the selection, crossover and mutation operators. The objective of this MR is to distribute evaluation tasks and find the global minimum fitness score.

We assume that the initial population is stored by several chunks in *Hadoop File System* (HDFS). The input of the map function is the chunk id and the subpopulation G'_{id} stored in the chunk. Then, the map function extracts individuals from the subpopulation. Each individual is attached by the first mapper a random number rv whose value takes from 1 to $r-1$. According to rv, individuals are assigned to different reducers.

The task of the first reducer is to evaluate the fitness value of each individual and report the local minimum fitness in its subpopulation. Particularly, each reducer computes concurrently the fitness value of every individual. When the fitness value of an individual is obtained, it is attached at the end of each individual and compared with a local minimum fitness. If the obtained fitness value is less than the local minimum, the value of the local minimum is updated. After all the individuals are processed, each reducer outputs the local minimum fitness and the set of individuals with their fitness scores in HDFS. Each part of the output data is further divided into two parts: local minimum and individuals. The two parts are stored in two different files. The former is the input of the second phase of MR, which finds the global minimum fitness score.

Algorithm 6: MR algorithm evaluates the initial population and finds the global minimum fitness value

```
    class Mapper: Mapper1
        method Map(id, G'_id)
            for all individual ∈ G'_id do
                Emit(rv, individual)
 5
    class Reducer: Reducer1
        method Reduce(rv, individual)
            localMin ← ∞
            for all individual ∈ [individual] do
10              fit ← Evaluate(individual)
                individual ← (individual,fit)
                if fit < min then
                    if (fit < localMin) then
                        localMin ← fit
15          Emit(1, (localMin, [individual,fit]))

    class Mapper: Mapper2
        method Map(1, [localMin], [individual,fit])
            for all localMin ∈ [localMin] do
20              Emit(1, localMin)

    class Reducer: Reducer2
        method Reduce(1, [localMin])
            globalMin ← ∞
25          for all localMin ∈ [localMin] do
                if localMin < globalMin then
                    globalMin ← localMin
            Emit(1, globalMin)
```

In the second phase, local minimum fitness of each subpopulation is gathered by the mapper. All the local minimum values are forwarded to a single reducer to calculate the global minimum fitness value. The global minimum is then regarded as the criteria for qualifying offspring.

MR produces offspring

The aim of Algorithm 7 is to produce, evaluate offspring, and update the global minimum fitness score. The input of the algorithm consists of individuals in the current generation G_t with their fitness values, and the global minimum fitness. Again, the first map function is used to regroup the subpopulations. The objective of the reorganization is not only to distribute the reduce task but also to exchange individuals in demes and avoid the convergence at a local minimum point.

The main function of the algorithm is undertaken by the first reduce phase. It is

Algorithm 7: MR algorithm produces offspring and finds the global minimum fitness value

```
    class Mapper: Mapper1
        method MAP id, G'_t_id
            for all individual ∈ G'_t_id do
                EMIT(rv, (individual, fit))
5
    class REDUCER: REDUCER1
        method REDUCE(rv, [individual, fit])
            localMin ← GlobalMin
            for 1 to ⌈v_p/r⌉ do
10  offspring ← Mut(Crossover(Select([individual, fit])))
                if (fit ←Evaluate(offspring))< min then
                    offspring ← (offspring,fit)
                    if (fit < localMin) then
                        localMin ← fit
15          EMIT(1, (localMin, [offspring,fit]))

    class Mapper: Mapper2
        method MAP(1, [localMin], [individual,fit])
            for all localMin ∈ [localMin] do
20              EMIT(1, localMin)

    class REDUCER: REDUCER2
        method REDUCE(1, [localMin])
            globalMin ← ∞
25          for all localMin ∈ [localMin] do
                if localMin < globalMin then
                    globalMin ← localMin
            EMIT(1, globalMin)
```

responsible for generating and qualifying offspring, and determining the local minimum fitness score. All the qualified offspring and their fitness values are written to the HDFS system with the local minimum after $\lceil v/r \rceil$ offsprings are produced. Then, the local minimum fitnesses are sent to the second MR phase to determine the global one.

Complete PGA

Algorithms 6 and 7 successfully transform the centralized GA 5 into its complete parallelized version as shown in Algorithm 8.

Algorithm 8: Parallelized Genetic Algorithm for k-PCFLP

```
generate initial population G_0
continue = true; t = 0
minFitness = Algorithm6(G_t)
while continue and t < threshold do
    continue = false; t = t + 1
    newPopulation= G_{t-1}
    (newMinFitness, {offspring}) = Algorithm7(G_{t-1})
    if newMinFitness<minFitness then
        continue = true
    add {offspring} to newPopulation
    sort newPopulation according to fitness
    G_t = the v first individuals in sorted newPopulation
return the first individual in G_t
```

4.4 Evaluation

The inner goal of this section is to demonstrate that our GA enables large-scale evaluation of real-world cases of Telco-CDN serving a lage population of users. Since the management of both a Telco-CDN and an ISP network requires tuning various specific parameters, we do not produce a comprehensive evaluation of both push and caching strategies in Telco-CDNs. We rather focus on one specific case: the Orange network with a possible deployment of a Telco-CDN and a typical VoD service. Through this example, we show that caching strategy is not necessarily the best implementation for the management of Telco-CDN since such strategy is blind to the underlying infrastructure.

4.4.1 Traces from a national VoD Service

We utilized traces from the Orange VoD service, which is a typical national VoD service offered to the clients of the ISP Orange. The measurement period ranges from June 5th, 2011 to June 19th 2011. We picked 728,931 download requests for 21,385 distinct video from 22,305 unique end-users. More interestingly, we were able to associate each end-users to one of the 13 *regions* that correspond to a metropolitan network in ISP's network. We were unable to track the session duration, so we assume that session duration follows a uniform distribution from 60 to 120 minutes.

We show the service usage on the seven last days in Figure 4.5. We observe the typical daily pattern, which is common to all localized entertainment service. Two daily peaks happen at around 2:00pm and 11:00pm, and the low activity is around 4:00am. Moreover, we also observe the difference between week days and weekends. These characteristics matches our expectations that our traces are representative of a local, popular service in real world.

We separated these traces into two *periods*. The *warm-up* period represents the input of the prediction of the end-user preference. It also allows to fill the cache in the caching strategy. This period lasts for the seven first days of the traces.

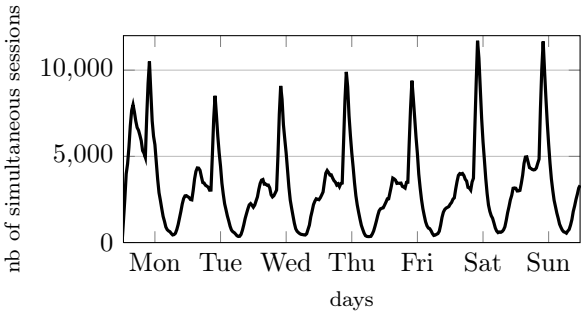


Figure 4.3: Service usage

The *test* period starts immediately after the warm-up period and lasts x days, for $x \in \{1, 3, 5, 7\}$.

4.4.2 Network topology and management

A typical topology of a telco-CDN is given in Figure 4.4. The traditional CDN interfaces with Telco-CDN on **Point-of-Presence (PoP)**. There are three major PoPs in France. For this implementation, which is the most probable according to stakeholders, an ISP deploys small data-centers near the routers that connect the backbone core network and the metropolitan access networks of every region (the blue, circular nodes). Thus each Telco-CDN repository is in charge of a regional area.

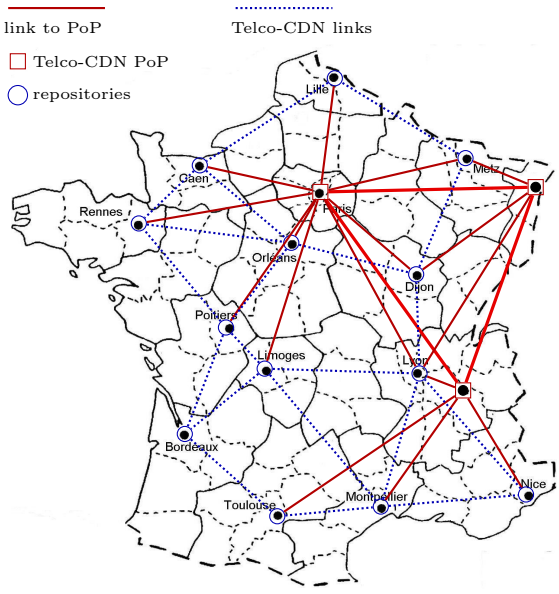


Figure 4.4: Envisioned telco-CDN topology in France. Three telco-CDN PoPs enable inter-connections with multiple traditional CDNs.

We use the network topology illustrated in Figure 4.4, except that we consider only one PoP since we are considering only one VoD service. That is, the only PoP

is located at Paris. We consider that 13 repositories have been deployed by the ISP so that each router connecting a metropolitan area to the backbone is equipped with a repository. All the Telco-CDN repositories are homogeneous and possess the storage capacity of $5,000$ videos and out link bandwidth for streaming $1,000$ sessions simultaneously.

As for the management of this network, we consider a policy, which is today representative of the management decision that have to be taken by an ISP. There exist three categories of links. The costs are indicated in a unit of money (euros in our case) per kilometer and per transmitted video.

The **peering links** connect each repository to the PoP. The traffic on this link goes through the PoP, so it generates peering costs as well as possible monetary compensations to the CDN. We suppose the cost of peering link is 1,000. These are red plain lines in Figure 4.4.

The **internal links** connect a pair of geographically adjacent repositories in the backbone. They are dashed blue lines in Figure 4.4. These links are intra-domain links, which are much cheaper than peering links. Therefore, the cost of internal link is set to 1.

The **low-priority internal links** are regular internal links but the network operator experience troubles on these links (they are over-used, or subject to faults). In our simulations, we chose three links: from Lille to Metz, from Lyon to Nice and from Limoges to Poitiers. Since the network operator prefers to use them in low priority, we set a cost of 1,000 but these links are shorter than peering links, so they still represent an opportunity to avoid going to the PoP.

The Telco-CDN system operates as follows: an end-user's request is firstly directed to its regional repository. If the regional repository does not have the required video or its bandwidth capacity is over, this repository explores its *cooperation group*. A cooperation group is defined for each repository j as a set of repositories whose distance to j is smaller than the distance from j to the PoP. Any repository in the group having the requested video and free bandwidth can serve the client. If the requested video is stored within the cooperation group, the request is redirected to the matching repository. Otherwise, it is a *miss* and the request is forwarded to the PoP.

4.4.3 Evaluated Strategies

Random placement extracts the videos requested during the warm-up. Then each unit of storage space is randomly filled with one video.

Proportional placement distributes the replicas of each video according to its popularity. The number of replicas is proportional to its request rate during the warm-up, and each repository holds only one replica of a video.

Push strategies result from our PGA. To investigate the impact of the recommendation accuracy, we produce the prediction of users' preference by mixing up warm-up and test part. Specifically, we replace a certain percentage of records in the test part with records from the warm-up part to generate predictions with various qualities. In particular:

- A **Perfect Optimal** (or Perf-Opt) takes in input of the computation for he k-PCFLP all the requests that will actually be requested during the test period. It is like the recommendation system was prophetic.
- A **Realistic Optimal** (or Real-Opt) takes in input of the computation only requests from the warm-up period. The recommendation engine does not predict anything but just rely on the past.

Perf-Opt and Real-Opt can be regarded as the upper and lower bound of our optimal push strategy.

LRU caching strategy implements the traditional and widely adopted caching strategy. To avoid unfairness due to initially empty caches, we measure from the first request of the test period. When a request cannot be fulfilled at a cache, the repository looks for possible hit in its cooperation group. However, in case of miss, the video is not re-forwarded to these repositories.

4.4.4 Results

We enter now in the evaluations of the strategies in our toy-Telco-CDN. The main criteria that matters in our study is the overall cost. For each placement strategy, we compare the cost to the one without repositories, that is, we compute the *normalized overall cost* as the ratio of the overall cost from fetching the video to the cost using only peering links.

Computation time of quasi-optimal tracker

We do not aim here to analyze the performances of our genetic algorithm, and to explore the benefits of our parallelized implementation. However, we can report some basic computation time to express that the implementation of a quasi-optimal tracker is possible using today's technologies.

The algorithm has been implemented on a MR cluster consisting of 10 machines with dual 2.70 GHz Pentium processor and 4 GB RAM. We set the population of each generation to 500 individuals and the initial population is stored in files whose overall size is about 150 MB. For all instances, the algorithm converges in about 350 generations, which takes less than 12 hours. With regard to the relatively sub-optimality of our implementation and to the small number of computers involved in this computation, it is clear that the implementation of a quasi-optimal tracker can easily be implemented at a large-scale, typically in the context of such service where placement is re-done on a daily basis.

Impact of the recommendation accuracy

Our tool enables the comparison of various parameters on the overall network performances in a fair manner. Here we explore as an example the importance of the accuracy of the recommendation engine.

The recommendation engine provides a set of p_{ik} that reflects that end-user i will request video k in the near future. We did not implement any recommendation engine, but we emulated a recommendation engine with various levels of performance.

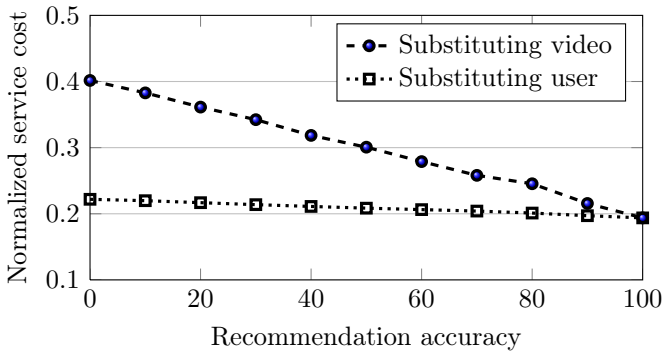


Figure 4.5: Impact of prediction on Perf-Opt

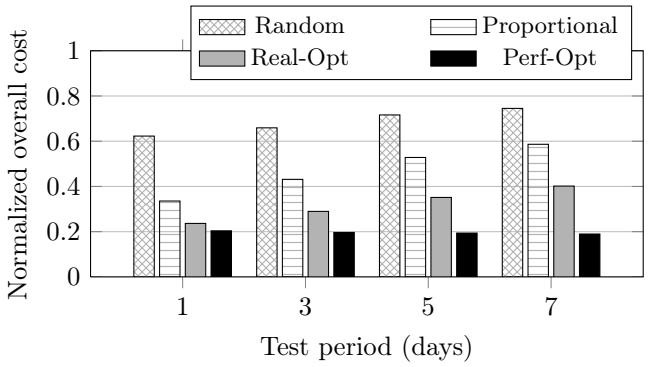


Figure 4.6: Normalized overall cost of push strategies

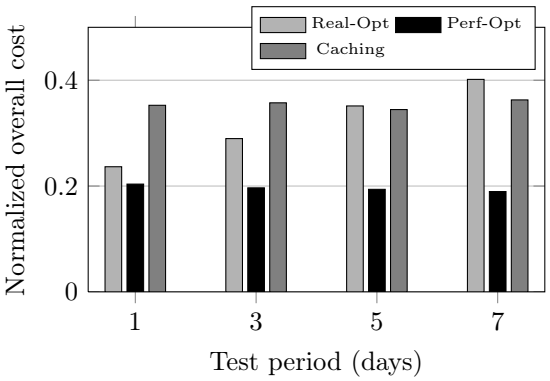
In the set of non-null p_{ik} that are given in input of the algorithm, a percent of the requests have been replaced by requests from the warm-up part. The perfect recommendation engine (accuracy $a = 100\%$) is thus able to predict all requests. The realistic recommendation engine (accuracy $a = 0\%$) does not predict any request.

When we replace a recommendation, we can either substitute either the requested video or the requester. We show in Figure 4.5 the performances for a series of accuracies.

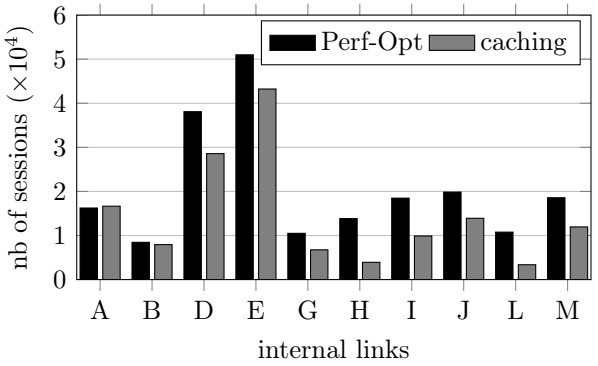
Substituting videos gives a larger bad effect on the efficiency of our optimal placement. The influence of the mistakes on users may be canceled by the integration of users' requests at the regional repository. Therefore, only marginal increment of cost is yielded. However, the error of pushing wrong videos is not compensable, which doubles the overall cost. Thus, we emphasize here the importance of addressing the videos that will be popular in the near future at a regional scope rather than at individual ones.

Performances of push strategies

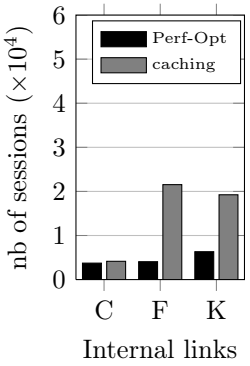
We study the push strategies, especially both random and proportional to the quasi-optimal strategies. Figure 4.6 shows the prominent performances of our quasi-optimal computation. Although the proportional random strategy is widely accepted as a de-



(a) Overall cost of Optimal and LRU



(b) Internal link utilization



(c) Low priority link utilization

Figure 4.7: Optimal placement versus LRU Caching

cent heuristic for placement in terms of hit ratio, we show here that the performances are bader than the optimal placement in terms of Telco-CDN management.

When the test period is longer, the performances degrades as some videos that were not requested during the warm-up enter in the catalog, thus are requested. Obviously, the Perf-Opt strategy is not affected by such novel videos, but we observe a significant degradation of the performances of Real-Opt (almost twice less efficient). We mitigate this observation by arguing that placing videos on a weekly basis is not serious when it is possible to do it on a daily basis.

Push strategies versus caching

What follows is the main outcome of this paper: we compare, on a given network configuration and a given VoD service, the advantages of a push strategy versus a caching one. The overall performances of LRU caching in term of cost is compared with push strategies in Figure 4.7(a). Please note that the performances of both caching and Perf-Opt are not affected by the duration time, the latter because it is prophetic, the former because it dynamically uploads the content in the repositories. On the contrary, performances of Real-Opt degrades when no new computation is

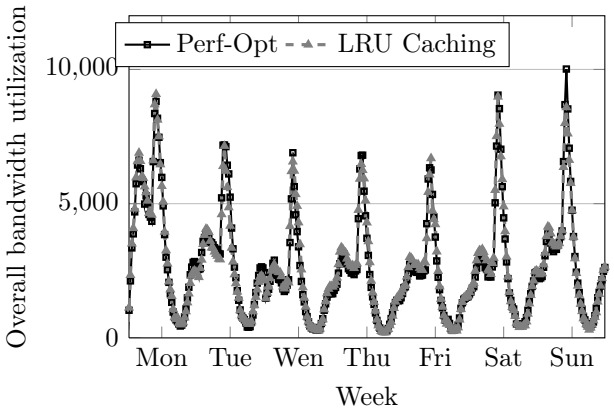


Figure 4.8: Telco-CDN bandwidth utilization

done.

For the test period of one day, which is the most probable to implement, the performances of push strategies are much better than caching. They achieve an overall cost that is 1.8 times smaller than LRU caching. This result demonstrates that LRU caching is a blind strategy that performs remarkably well for hit-ratio, with almost no implementation cost. However caching is far from ideal for an ISP that wants to actually manage its network as well as its Telco-CDN.

The reason of the predominance of Perf-Opt is revealed in Figure 4.7(b) and Figure 4.7(c). Both figures count the total number of sessions passed through each Telco-CDN internal link. In Figure 4.7(b), it is visible that a Telco-CDN implementing LRU caching uses less internal links than implementing Perf-Opt. In the meantime, Figure 4.7(c) shows that Perf-Opt utilizes in low priority the low-priority links, which is exactly the desire of the ISP, although LRU caching utilizes these links as regular links (note that they are still cheaper than peering links). In other words, while Perf-Opt allows to engineer the traffic (here to avoid using low priority links but more accurate policies can obviously be implemented), a cooperative LRU caching acts naively.

We represent in Figure 4.8 the overall bandwidth utilization on Telco-CDN repositories. As could be expected, LRU caching performs similarly to Perf-Opt. A naive scientist looking only at such results would definitely adopt LRU caching since it is far easier to maintain, but the results given in Figure 4.7 emphasizes the opposite. It is remarkable that Perf-Opt performs as well as LRU caching although its impact on the network infrastructure is 1.8 times less severe.

4.5 Summary and Conclusion

This chapter focuses on the design of Telco-CDN, which is a inevitable trend in the development of future network operators. Since ISPs are interested in both hit-ratio and traffic management, we re-open the debate about strategies for pushing video content to repositories. Our new facts are twofold: first, pushing content in an quasi-

optimal manner is practically feasible. Second, it makes sense since it enables smart traffic management. We thus open a new perspective of development for research in the area of Telco-CDNs.

Chapter 5

Cooperative Caching for CCN

This chapter proposes the cooperative caching protocol for CCN, which can be regarded as a clean-slate design of Telco-CDN. In the proposition, we describe the network model and give the detailed information about our cooperative caching protocol. Then, we derive the theoretical analysis that emphasizes the advantage of this cooperative caching. Thereafter, we show the evaluations obtained by CCNxProSim, a simulation platform automatically deploys CCNx prototype on real machines. Finally, we detail the realization of CCNxProSim.

5.1 Introduction

Over the last decade, the Internet has been *patched* to support the delivery of content at large-scale. The original flaws (especially the poor performances of communication links traversing several Autonomous Systems (AS) [Lei09b]) have been overcome by the deployment of large-scale Content Delivery Networks (CDN) such as the Akamai network [NSS10]. The recent works toward Telco-CDN introduce new architectures, which call for embedding massive storage spaces into ISP networks. *Content Centric Network* [JST+09], which allow to route queries and data based on content name, provides a clean-slate approach for conceiving Telco-CDN or our Telco-CDN in the future. The overall function of CCN is presented in section 2.2.3. These protocols enable the exploitation of the storage resources of any machine in the network, in particular the content router (CR).

Implementing a caching strategy on a potentially large set of inter-connected CRs is referred to as *in-network caching*. The design of caching policy for in-network cache is a subtle exercise. In their seminal paper, the authors of CCN suggested to use a *Least Recently Used* (LRU) policy for the cache management of every CR [JST+09]. The main advantage of this policy is that it is simple and easy to implement. But this policy does not make CRs cooperate with each other. In other words, this in-network caching strategy is an aggregation of individual actions. On the other hand, the design of cooperative caching policies is hazardous for the network. We highlight now some key requirements that any cooperative in-network caching policy should meet.

- *the caching policy should stay simple and lightweight.* The CCN routing proto-

col is inspired from today's IP routing. In particular, data forwarding in CCN and in IP networks are both based on finding the longest prefix matching in a routing table. The main motivation behind this choice is twofold: the simplicity and the weak demands it makes on lower layer. Both have been keys in the success of the Internet. This spirit should also guide the design of in-network caching. First, the envisioned cooperative caching on CR should be completely distributed. A CR takes caching decision by using only local information. Second, the cooperative caching policy should be able to run on machines with restricted hardware capabilities. For example, Perino and Varvello [PV11] have shown that current hardware technology is not able to support a CCN deployment at Internet level. Their observation is that no extra-resource is available to sustain the implementation of complex cooperative caching policies.

- *economical aspects should drive the selection of data to cache.* Traditionally, the purpose of in-network caching was to alleviate the load on network links. Nowadays, in-network caching aims principally to reduce the expensive cross-domain traffic. This idea is a direct consequence of the *flattening* of the Internet. The architecture in three tiers between Autonomous Systems (AS) is progressively replaced by a denser and less hierarchical topology [LLJM+10, SdM11, DD10] where the content providers own AS and directly *peer* with the networks of Internet Service Providers (ISP) [IP11]. The major content providers now have settlement-free interconnections, or even charge ISPs for the access to their content in some cases [Got10, LLJM+10]. In the CCN perspective, an operator of AS becomes a small content provider through the CRs it manages. A rationale behavior is to cache as much as possible the most expensive content, *i.e.* the content that is expensive to be accessed from content providers or through other transit networks.

- *the policy should cope with the mismatch between content size and storage capacity.* Studies show that video content will represent more than 90% of the whole Internet traffic in a few years [Cis10]. High-definition video streams with bit-rate in the order of megabits per seconds requires storage capacity in the order of gigabits. In comparison, the storage capacity of CR will probably be small. For an ISP network, the caching capacity in CR is at most 1 terabytes, while it decreases to several hundreds gigabytes in the backbone network environment [PV11]. However, stream delivery applications cannot monopolize the caching space. The idea is rather to partition the caching resource among a selected set of applications [CGP11]. In consequence, the storage space reserved for video stream delivery is unlikely to exceed one hundred gigabytes. If we consider the power consumption of CR, the storage resource constraint becomes even stricter, for example, the total caching capacity is only 36 gigabytes in [LRH10].

5.1.1 Our Proposal: Cooperative In-Network Caching

We propose to replace the LRU policy of CCN by a new cooperative policy, which is especially designed for objects that contain many segments that are accessed in

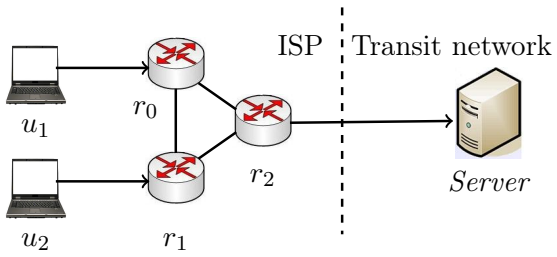


Figure 5.1: Two end-users u_1 and u_2 requesting the same movie

a consecutive manner, *a.k.a.* video. Let us show by an example the limitations of LRU for video diffusion in Figure 5.1. Assume that both end-users u_1 and u_2 watch the same video with a delay corresponding to 21 *segments* (a video segment in CCN is in the order of dozens of seconds), and that each router has a cache capacity of 10 segments for this video service. According to the LRU policy, the cache of r_0 and r_2 contains the last ten segments that u_1 has requested. When u_2 starts the video, the cache r_0 and r_2 store segments ranging from 11 to 20, and the cache of r_1 is empty. All requests from u_2 will thus have to be forwarded to the server. The lack of coordination among the routers results in an inefficient caching strategy with redundant data stored on adjacent routers and unexploited storage of a router.

Our new caching policy is based on a light cooperation among routers: each router does not cache all the segments that it routes, but only a part of them. A router is associated with a *label*, which is a positive integer smaller than a fixed integer k. A router uses the LRU policy only for the segments such that the index of this segment, modulo k, is equal to the said label. In our example of Figure 5.1, let us assume that k is equal to 3, and each router r_i is associated with label i for i in $\{0, 1, 2\}$. With this policy, router r_2 stores the segments $\{2, 5, 8, \ldots, 20\}$, which correspond to the last segments routed by r_2 with a segment index modulo 3 equal to 2. During the video diffusion, r_2 has also forwarded segments $\{0, 3, 6, \ldots, 18\}$ to r_0 because this latter is responsible of all segments with index modulo k is equal to 0. Similarly, r_1 stores segments $\{1, 4, \ldots, 19\}$ now. Therefore, the request from u_2 will not have been forwarded to the server, but directly satisfied by the three cooperating routers. The consequence is a reduction of the number of requests that have to go out the ISP network. The main idea is that we eliminate redundancy among adjacent CRs, therefore we increase the amount of distinct segments that are stored in the in-network cache.

5.1.2 Our Contributions: Algorithms and CCN Protocol

In our work, we introduce this new caching policy, and we carefully analyze it. To our knowledge, this work is the first cooperative policy for handling streams in CCN. Our work demonstrates the interest of designing lightweight cooperative policy for in-network caching. The contributions are as follows:

1. we present an algorithm that allows every CR to determine its label. This is the *initialization stage*. A trivial implementation consists in letting each CR

randomly choose its label. In previous works, we have shown that significant gains can be obtained from a label assignment that takes into account the network linkage among CRs [CLS09b]. However, the optimal assignment has been shown to be NP-complete. In chapter 3, we introduced several heuristic and approximation algorithms. In the present chapter, we present a *distributed* algorithm that allows each CR to determine its label in a way that the overall "*distance*" between CRs are minimized. The notion of distance between CR is detailed in Section 5.2.1. This assignment of labels has proven performance guarantee, it is not worse than $(\frac{3}{2}k - \frac{5}{2})$ of the optimal assignment.

2. we augment the CCN protocol such that our cooperative caching strategy is natively implemented. We describe the new CCN algorithms and its integration into the open-source CCNx prototype. We show in particular that the protocol keeps the simplicity of the original CCN protocol. We present the refinements that are necessary to implement the cooperative caching.

3. we provide theoretical analysis based on simplified network models to highlight the advantages of our cooperative caching policy on a non-cooperative one. This theoretical analysis demonstrates in particular that our cooperative caching policy significantly improves the caching performances for mid-popular videos.

4. we develop the CCNxProSim simulation platform and analyze the performances of our augmented CCNx prototype. We used a network with 40 machines running both the new CCNx and the original prototype with the traditional LRU policy. In order to emulate the behavior of users in VoD and time-shifted services, we generated some traffic from the measurements conducted in [YZZZ06] and the synthetic traces generated in [LS10] respectively. The results demonstrate that the ISP can reduce up to 45% of the cross-domain traffic related to the targeted streaming applications, while increasing by less than 13% the overall number of messages that are exchanged within its network.

Following are the outline of this chapter. In section 5.2, we give the background and describe our network model. Section 5.3 concentrates on label distribution. The detailed information about our cooperative caching protocol is given in section 5.4. We show the theoretical analysis and practical evaluation in section 5.5 and 5.6 respectively. Specifics of the design of the CCNxProSim platform is illustrated in section 5.7. Section 5.8 concludes our work.

5.2 System Model

The principles of CCN have been introduced in section 2.2.3. Please refer to [JST+09] for more details. We start directly the description of our network model and the approaches an ISP can implement to run cooperative caching protocols.

5.2.1 Network Model

We consider a network $\mathcal{N}$ consisting of a set of routers, and a set of bidirectional links between these routers. We note by V the subset of routers that are CR (*i.e.* having

caching capacity). We assume that the ISP is able to compute a static *distance* d_{ij} between two CRs r_i and r_j. This distance reflects the connectivity of two CRs. The metric of distance is generic: for example the length of the shortest path joining r_i to r_j in $\mathcal{N}$, the inverse of the capacity of routers on this path, or the average latency measured between these two routers.

The $k-1$ CRs in V that are the *nearest* from the CR r_i are expected to cooperate with r_i. Here, nearest means having the smallest distance. Our goal is to avoid that these CRs store the same segments. In our special case of video stream delivery, the metric of the distance corresponds to the average latency or round trip time (RTT) from one CR to another. We made this choice for two reasons:

- *we want to reduce the delay.* Video segments that are not under the responsibility of a CR r_i must be retrieved from a remote CRs cooperating with r_i. Therefore, the RTT from r_i to any of its cooperating CR should be minimized in order to guarantee that the request emitted by the user is quickly fulfilled.

- *we do not consider bandwidth issues.* The distance could take into account the capacity on the links. It is true that the bandwidth is the most critical restriction in traditional video services, but the server is the main bottleneck. The distributed and multicast nature of CCN and the high performances of new-generation CRs overcome the bandwidth constraints. Hence, we ignore the bandwidth of CRs when we consider the distance, and we prefer to stick on the latency constraint. Actually, our generic distance formulation enables the implementation of various constraints, with regard to network characteristics.

We note by $N(i)$ the nearest subset of CRs from r_i in V, and, by extension, $N[i]$ is the set $N(i) \cup \{r_i\}$. In the following, we assume that non-CR routers are able to transmit the messages from one CR to another without troubles. The CRs do not experience failures.

5.2.2 Practical Details

We suppose that all the cooperating routers execute the CCN protocol. When an ISP embed our augmented CCN into its routers, the following configuration is set:

- *the name of streams targeted by the cooperative caching.* Each data packet in CCN is entitled by a unique hierarchical URL containing, both the content provider and the related applications. Each CR keeps in its memory a list of streams that can be cooperatively cached. The list is maintained manually by the ISP that controls the CR as it is part of this caching strategy. The ISP can add, modify and delete entries in the list. For example, the ISP can decide to apply the cooperative caching to the catch-up TV stream from a provider, say Hulu. Since Hulu embeds its video streams in Flash, the ISP adds an entry in the list involving two keywords: *hulu.com* and *FLV*. According to the convention given in [JST+09], the first segment of a video offered by Hulu should be named as *hulu.com/watch/channel.flv/_v<timestamp>_s1*. When this segment is received by a CR, its name is extracted and compared with entries in PIT as well as the keywords in the list. As both *hulu.com* and *FLV*

are discovered in the name of the received data, the CR triggers the cooperative caching.

- *the amount of storage space devoted for these streams.* The partitioning of caching space can be either statically divided or dynamically managed. The static division corresponds to the per-stream memory space allocated by each ISP following its own considerations as business concerns or CR capabilities. The performance of this static approach has been analyzed in [CGP11]. It is shown that the static partitioning is able to ensure a minimum hit probability to each of the applications sharing the cache. The authors of [LAS04] proposed an adaptive control scheme to self-tune the size of different cache partitions dynamically. Although this self-adaptive mechanism converges towards a proportional hit rate differentiation among applications, there is no guarantee for the worst performances.

- *the number of different labels k.* An ISP should decide the number of labels based on its intra-AS topology. The value of k must not be too large to yield long RTT between cooperating CRs, and degrade the quality of stream delivery. In the other hand, k should be large enough so that the in-network cache can store as many video segments as possible. Therefore, it is key to address the trade-off between service quality and caching efficiency. Our suggestion is that the ISP first determines a threshold of RTT over which the service might be degraded. Then the ISP gradually enlarges k, and constructs cooperative caching system following the instructions illustrated in section 5.3, until the distance between some CR and its cooperating CR exceeds the threshold.

5.3 Initialization Stage

Once the CRs are deployed and configured, each CR should determine its label. The aim is to ensure that every CR is as close as possible from all the labels that are different than its own label. We note by $F(i)$ the $k-1$ CRs having the $k-1$ other labels and that are collectively the closest from r_i. The sum of distances from a given CR r_i to the CRs in $F(i)$ is called the *rainbow distance* of r_i, and it is noted d_i. Formally, $d_i = \sum_{r_j \in F(i)} d_{ij}$. Determining the optimal assignment of labels, *i.e.* the assignment such that the sum of all rainbow distances is minimal, is NP-hard [CLS09b]. Note that, in an ideal scenario, $F(i) = N(i)$ but this solution is sometimes impossible.

We compute a *fractional optimal solution*, an impossible solution, which gives a lower bound of the optimal solution. We cut each label into k fractional pieces. Then, we form the fractional optimal solution S by assigning to each r_i in V a fraction $\frac{1}{k}$ of all the labels. Therefore, each CR can have an access to any label c by retrieving a fraction of c to itself, and the remaining $\frac{k-1}{k}$ fractions from its $k-1$ nearest nodes in the cooperative caching. It is easy to show that S is a fractional optimal solution because the sum of fractions assigned to each node is equal to 1, and all labels can be accessed from itself and nearest neighbors in the caching network, that is, $F(i) = N(i)$. We denote the rainbow distance of CR r_i in the solution S as $\bar{d}_i$.

5.3.1 Distributed Algorithm

Our distributed algorithm is inspired by Li and Li [HL08]. Details are given in Algorithm 9. The main idea is that every CR tries to optimally assign labels to their nearest CRs and themselves. In order to avoid that two CRs assign a label to the same CR simultaneously, each CR r_i initially exchanges information with all CRs in $N(i)_2$ (lines 1-2), where $N(i)_2$ denotes the union of the $k-1$ nearest CRs of r_i and their respective $k-1$ nearest CRs. Then, these 2-hop neighbors are sorted in increasing optimal rainbow distance $\bar{d}$ order (line 3). Once r_i has received messages from all CRs in $N(i)_2$, it enters in *waiting mode*.

Algorithm 9: Distributed Approximate Algorithm for Label Assignment (CR r_i)

Require: Initially:
1 determine $N(i)_2$, compute optimal rainbow distance $\bar{d}_i$
2 broadcast $\bar{d}_i$ to every CR in $N(i)_2$
3 create a list $L(i)$ containing all CRs in $N(i)_2$ sorted in increasing rainbow distance order
4 enter *waiting mode*

Require: In waiting mode:
5 upon reception of a *release* message from a CR r_j
6 discard r_j from $L(i)$
7 **if** r_i is the first CR in $L(i)$ **then**
8 let $C_{toAssign} = \{0, 1, \ldots, k-1\}$
9 let $N_{toAssign}$ be all CRs in $N[i]$ without assigned label
10 **for** all CRs $r_j \in N[i] \setminus N_{toAssign}$ **do**
11 remove $\mathcal{C}(j)$ from $C_{toAssign}$
12 **if** $|N_{toAssign}| = |C_{toAssign}|$ **then**
13 assign labels in $C_{toAssign}$ to CRs in $N_{toAssign}$ such that no pair of CRs are assigned the same label
14 broadcast *release* message to every CR in $N(i)_2$
15 enter in *final mode*.

Require: In final mode:
16 upon reception of a *release* message from a CR r_j
17 discard r_j from $L(i)$
18 **if** $L(i)$ is empty **and** no label is assigned **then**
19 assign to r_i the label whose closest CR is the farthest among all labels

The CRs with minimal optimal rainbow distance among their 2-hop nearest neighbors execute the algorithm first (lines 5-7). When a CR cannot produce a local optimal assignment, it leaves itself unassigned. Such a configuration occurs when at least two CRs among the $k-1$ nearest neighbors have already been assigned the same label. Otherwise a CR assigns labels to all its unassigned nearest neighbors and itself (lines 8-13). Finally, a CR sends a *release* message and enters in *final* mode.

In the final mode, some CRs may be still unassigned. It means that neither they,

nor their nearest neighbors were able to optimally assign labels. In this case, a CR assigns itself the label that is the farthest (lines 18-19). The maximum complexity of the algorithm is $\mathcal{O}(n \log n)$, where n is the number of CRs in the network.

5.3.2 Correctness and Analysis

Provided that the algorithm runs in a *correct environment*, i.e., there is neither faulty links nor faulty nodes, it returns a solution satisfying the following conditions. First, it runs in finite time. Second, each CR eventually holds a label. Third, there is no missing label in the system.

Theorem 4 *The algorithm gives a valid solution in a correct environment.*

Proof. The last condition is easily satisfied when the first CR (the CR possessing the local minimum rainbow cost) assigns labels to its nearest neighbors and itself. To show that the first and second conditions are also tenable, we just need to prove that r_i will receive all *release* messages from its 2-hop neighbors in a finite time. If the algorithm does not terminate, it must be some nodes r_i and r_j such that r_i never receives a *release* message from r_j, so r_i stays in waiting mode, and never broadcasts the *release* message. Yet, the fractional distance being a unique real number, there is always a CR with a smallest distance, which can enter final mode and broadcast the release message. This also leads to the fact that each CR will execute label assignment. Together with the fact that the distance of each CR is broadcasted only once, we conclude that no CR will be in waiting mode for infinite time. Since the number of nodes is finite, the algorithm terminates in finite time, thereafter each CR holds a label. □

Theorem 5 *For any $k \geq 3$, Algorithm 9 gives an integer solution no more than $\frac{3}{2}k - \frac{5}{2}$ times of the fractional optimal solution for the Label Assignment Problem.*

For the proof of the approximation guarantee, please refer to the proof of theorem 2 in section 3.3.

5.3.3 Implementation in CCN

A trivial way to realize Algorithm 9 is to use some inappropriate message passing mechanisms, (*e.g.*, broadcasting $\bar{d}_i$ and release message), which violates the rationale of CCN. Namely, data packet is transmitted only when the corresponding interest arrives at the CR. As we are not willing to infringe the rules of CCN, we now detail how to apply Algorithm 9 in CCN.

Instead of broadcasting content directly, we leverage on the standard approach for publishing content in CCN. When content becomes available, the CRs connected to the content provider are responsible for flooding the announcement containing the content name to all CRs, so that FIB entries are established to route interests in the future.

In the first step of Algorithm 9, each CR broadcasts via all interfaces an advertisement for a "content" named *ISP_ name/CR_ id*, which indicates that the information

about itself is available. This advertisement does not need to be flooded to all CRs in the network. The destinations of the advertisement are the 2-hop neighbors of the original CR. Thus, the time to live (TTL) of this advertising message is restricted at two hops. Moreover, to calculate the rainbow distance, a timestamp should be attached to the message. After all advertisements from 2-hop neighbors are received, a CR r_i computes $\bar{d}_i$ based on timestamps, and then sends the interest inquiring $\bar{d}_j$ of all r_j in $N(i)_2$. When the CR is in the waiting mode, it periodically sends interests in order to pull release messages from its 2-hop neighbors.

Please note that we have proposed in [LLS11] some centralized algorithms for other purposes such as improving the fairness of distance between CRs. As we are in an intra-ISP environment, where all CRs are under the control of ISP, these algorithms can be implemented as well. It means that the "label" can be computed in a centralized way, then pushed to CR.

5.4 Augmented CCN Protocol

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

5.4.1 New Tables in CCN

Our cooperative caching strategy requires the implementation of two new tables. First, every CR r_i maintains the information of its $k-1$ closest CRs in $N(i)$ in a new table, namely *Cooperative Router Table* (CRT). There are three fields in CRT: the label, the identifier of the cooperative router and the interface. Thus, every CR knows where to redirect an interest or forward a video segment. The second table is the *Cooperative Content Store* (CCS). In CCS, a CR keeps the names and the sequence numbers of all the segments that may be found in its cooperative cache. When an interest arrives, the preference of the four prefix matches is CS match to CCS match to PIT match to FIB match.

5.4.2 Segment Distribution in the Cooperative Cache

When a CR r_i (with label c_i) receives a segment s, the first reaction of r_i is to forward s according to the corresponding PIT entry. Then it has to take a decision (whether to cache it or not) based on c_i, the identifier s of this segment, and the match result. We describe the execution of r_i in Algorithm 10:

- this segment is "handled" by r_i, that is $s \mod k = c_i$. The CR r_i adds s into its cache, and removes the least recently used segment.
- this segment is not handled by r_i, that is $s \mod k \neq c_i$. Firstly, r_i finds in its CRT the router r_j having the label c_j that matches with the segment s, that is $s \mod k = c_j$ (line 8). Then, r_i checks its PIT to see whether the interface to r_j is a matching entry. In the case that no PIT match is found for the interface to r_j, the CR r_i sends a cooperative interest (*c-interest*) with the

name *ISP_name/CR_r_j/content_available/CR_r_i/content_name* out of the interface pointing to r_j according to CRT (line 10).

In response to this c-interest, r_j becomes aware that content should be retrieved from r_i, and it sends back an acknowledgment to consume the interest from r_i as well as an interest with the content name over the interface to r_i. See Algorithm 11 for the treatment of c-interest messages.

Once the segment s is transfered to r_j, the CR r_i adds s in the CCS Table (line 11), so that later interests requiring the same segment will be forwarded to r_j, but no longer according to the FIB.

Finally, r_i cleans up the PIT entry.

Algorithm 10: Action upon Receiving *Content*

Require: r_i **receives s:**

```
1  forward s according to PIT
2  if (s mod k = c_i) then
3      if cache is not full then
4          cache s
5      else
6          replace least recent used segment with s
7  else
8      determine r_j ∈ N(i) such that s mod k = c_j
9      if interface to r_j is not in PIT then
10         send c-interest_s:
           ISP_name/CR_r_j/content_available/CR_r_i/content_name
           to r_j
11     add the identifier of s in CCS
12 delete PIT entry for s
```

Although two extra rounds of message exchange are yielded by the cooperative caching protocol, there is no side-effect on responding delay to users since the extra messages are introduced only when no request from r_j is pended. To prevent broadcast storm, each data packet should carry a random *nonce*. When a duplicated packet with the same nonce is received, it should be immediately discarded.

5.4.3 CCS Consistency

Ideally, the CS of a given CR r_i should always be consistent with all the CCS tables of all CRs that cooperate with r_i. In particular, when a CS entry of r_i is discarded by the replacement policy, the corresponding entry in the CCS of a CR r_j with $r_i \in N(j)$ should also be deleted, otherwise interests for the eliminated content may be lost in the forwarding process. For example, if r_j receives an interest requiring segment s, it finds the CCS match point to r_i. Assume that segment s in r_i has been discarded. The CR r_i forwards the interest following the FIB entry. If r_j is an intermediate CR between r_i and the data source, the interest will be regarded as a duplicated one, and

discarded by r_j. Therefore, the interest for segment s is lost. We should remind that the lost interest can be recognized as a duplicated one because every interest is given a random nonce when it is generated.

To both maintain consistency and minimize the number of control messages, we use piggyback interest (*p-interest*) to carry the control information, where the name of the interest is *ISP_name_/CR_r_i/CCS_inconsistent/content_name* . A CR r_i with label c_i acts as it is illustrated in Algorithm 11 when an interest for segment s is received:

Algorithm 11: Action upon Receiving *Interest*

```
Require: r_i receives interest_s:
 1  if (s mod k = c_i) then
 2      if interest_s is c-interest_s then
 3          send back acknowledgment and interest over the receiving interface
 4      else if CS match then
 5          send back data
 6      else if PIT match then
 7          add receiving interface in PIT
 8      else
 9          interest_s ⇒ p-interest_s:
            ISP_name_/CR_r_i/CCS_inconsistent/content_name
10          multi-cast p-interest_s according to FIB
11  else
12      if interest_s is p-interest_s and r_j ∈ CRT_i then
13          eliminate s in CCS
14          insert request for s in PIT
15          multi-cast p-interest_s according to FIB
16      else
17          if CS match then
18              send back data
19          else if PIT match then
20              add receiving interface in PIT
21          else
22              multi-cast interest_s according to FIB
```

- the requested segment s is handled by r_i, that is $s \mod k = c_i$. The CR r_i first extracts the content name and detects whether its identifier and the keyword *content_available* is in the name. If so it sends an acknowledgment content and the interest with received content name (lines 2 to 3). Otherwise, r_i calculates the CS match. If a CS match is found, it sends back the data directly (lines 4 to 5). Otherwise, if a PIT entry is found, it adds the requiring face into the pending list (lines 6 to 7). If neither CS match nor PIT match is found, r_i changes the interest into a p-interest, it generates a new nonce for the p-interest, and forwards this p-interest according to FIB entry (lines 8 to 10).

- the requested segment s is not handled by r_i, that is $s \mod k \neq c_i$. If the interest is not a concerned p-interest, then the CR r_i just executes the normal CCN process (lines 17 to 22). If the interest is a p-interest, the CR r_i needs first to determine whether the information is relevant. If the CR r_j indicated in the p-interest is in the CRT of r_i, then r_j is a relative cooperative router. The CR r_i should thus eliminate the entry for this segment in the CCS, and adds the requiring interface in its PIT. Finally, r_i forwards the interest according to the FIB, even if PIT already existed. This step ensures that the interest arrives at a provider (lines 12 to 15).

Upon receiving an interest, our cooperative caching may introduce an extra RTT between two cooperating CRs. In the initialization phase, we emphasized that cooperative routers should be picked based on latency. Therefore, the additional delay introduced by our cooperative scheme results in a small RTT value between close routers. The low delay overhead will be revealed in section 5.6.

5.4.4 Implementation into CCNx

We integrated our cooperative caching protocol into the CCNx prototype version 0.3.0 released in 2010. We used the basic C code of the daemon, which is composed by the following main functions:

- *ccnd*, the core function to start a ccnd handle (machine or process running CCNx). It provides caching, forwarding, and packet authentication functionalities. Concerning cache management, a hash-table tracked by an array of pointers is used to store segments.

- *ccndc*, bringing up a link to another ccnd handle. We built our CCN overlay through the command *ccndc add ccnx:/ccnx.org udp machine_ URL.*

- *ccnsendchunks*, injecting one segment of data from *stdin* into CCN.

- *ccncatchchunks*, allows one ccnd handle to retrieve the content published by another remote ccnd handle using *ccncatchchunks ccnx:/ccnx.org/film_ id/segment_ number.*

- *ccndumpnames*, dumping names of available content in local cache and outputting the names to stdout.

The modification of the program locates mainly in the *ccnd* function. Considering the compatibility with future CCNx releases, we preferred to not change the original cache organization. Instead, we create additional data structures, namely: the integer label of each ccnd handle; the integer k indicating the size of cooperative caching group; the table CRT; the list of CCS; and the doubly-linked list pointing to the CS implementing our cooperative caching policy. Moreover, we developed two functions *ccncoput* and *ccncoget* from the function *ccnsendchunks* and *ccncatchchunks* respectively. The two new functions aims at building up CRTs.

5.5 Analysis

We analyze the performances of the cooperative caching policy through some theoretical comparisons with other caching policies. We compare a simplified model of our cooperative CCN caching policy (represented in Figure 5.2) to two caching structures: k CRs running in parallel (see Figure 5.3) and k CRs connected in a series circuit (see Figure 5.4).

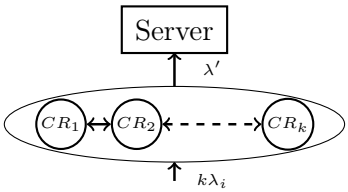


Figure 5.2: Cooperative caching

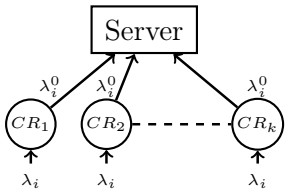


Figure 5.3: Individual parallel caches

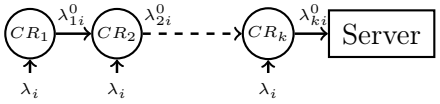


Figure 5.4: Individual caches in series

In Figure 5.3, k homogeneous caches with cache size C are connected to a server. Each cache receives requests for segment i with the same request rate λ_i. A cache miss happens when i is not cached. The cache miss rate for segment i is denoted as λ_i^0. The total cache miss rate of the k caches is $\lambda = k \cdot \lambda_i^0 = k \cdot \lambda_i e^{-\lambda_i \tau_i}$, where τ_i is the maximum inter-arrival time between two adjacent cache hits for segment i [CTW02].

Our cooperative caching is abusively modeled with k homogeneous caches working cooperatively in Figure 5.2. Each cache stores distinct segments and forms a *cooperative group* with cache size $k * C$. Although the caching policy of the cooperative group is not exactly LRU, we approximate the group as a single cache using LRU policy. The cache miss rate generated by the cooperative group is denoted as $\lambda' = k * \lambda_i e^{-k \cdot \lambda_i \tau_i'}$.

Proposition 1 *Cooperative caching achieves at least the same performances as the individual parallel caches.*

Proof. We need to prove that $\lambda \geq \lambda'$ for any $k \in \mathbb{N}^+$ and $C \in \mathbb{N}^+$. We begin our proof from the fact that the function $f(\tau_i) = e^{-\lambda_i \tau_i}$ is continuous and monotonically decreasing. The function $u(\tau_i) = \sum_{j=1, j \neq i}^{N} f(\tau_i) = \sum_{j=1, j \neq i}^{N} e^{-\lambda_j \tau_i}$ is also continuous and monotonically decreasing since $u(\tau_i)$ is the sum of $f(\tau_i)$. According to [CTW02], τ_i and τ_i' can be calculated by the following two equations:

$$\sum_{j=1, j \neq i}^{N} (1 - e^{-\lambda_j \tau_i}) = C \qquad \sum_{j=1, j \neq i}^{N} (1 - e^{-k \cdot \lambda_j \tau_i'}) = kC \tag{5.1}$$

Since $k \in \mathbb{N}^+$ and $C \in \mathbb{N}^+$, we obtain:

$$\sum_{j=1,j\neq i}^{N} e^{-\lambda_j \tau_i} \geq \sum_{j=1,j\neq i}^{N} e^{-k\cdot\lambda_j \tau_i'}$$

As $u(\tau_i)$ is monotone decreasing, we conclude that $\tau_i \leq k\tau_i'$. Because $f(\tau_i)$ is also monotone decreasing, we know that:

$$e^{-\lambda_i \tau_i} \geq e^{-k\lambda_i \tau_i'} \tag{5.2}$$

Multiply both sides of equation (5.2) by $k\lambda_i$, proof follows. □

The second structure is the caching in series as illustrated in Figure 5.4. The request rate from client for i at each cache is still identical and denoted as λ_i. Instead of forwarding the missed stream directly to the server, the missed stream passes to the next hop cache in the direction of the server. Since only the kth cache is connected with the server, the missed stream λ_{ki}^0 of the cache k is the missed stream of the multi-cache system. Therefore, the breakthrough point is to find the expression of λ_{ki}^0. Although the structure is simple, it is not trivial to deduce λ_{ki}^0, since the exact distribution function of the missed stream $f_{ki}^0(t)$ contains infinitely many terms [CTW02]. Consequently, we cannot deduce the exact miss rate because of the computational complexity. However, the incoming request rate at the kth cache is $\lambda_{ki} = l \cdot \lambda_i$, where l is a constant and $1 \leq l \leq k$. Let the miss rate be a function f_{ki}^0 of λ_i, then we have $f_{ki}^0 = \lambda_{ki}^0 = l\lambda_i e^{-l\lambda_i \tau_{ki}}$. Recall that the miss rate of cooperative caching is $f_i' = \lambda' = k\lambda_i e^{-k\lambda_i \tau_i'}$.

Proposition 2 *Cooperative caching achieves at most the same maximum miss rate as the individual caches in series.*

Proof. We need to prove that, for any $k \in \mathbb{N}^+$ and $C \in \mathbb{N}^+$, we have $\max(f_{ki}^0) \geq \max(f_i')$. The value of τ_{ki} can be calculated as follows:

$$\sum_{j=1,j\neq i}^{N} (1 - e^{-l\cdot\lambda_j \tau_{ki}}) = C \tag{5.3}$$

Instead of directly comparing τ_{ki} with τ_i', we use another variable τ_i'' and setup the equation below:

$$\sum_{j=1,j\neq i}^{N} (1 - e^{-k\cdot\lambda_j \tau_i''}) = C \tag{5.4}$$

Since $k \geq l$, combining equations (5.3) and (5.4) we have $\tau_{ki} \leq \tau_i''$. Applying the same method as that in the proof of Proposition 1 on equations (5.4) and (5.1), we obtain that $\tau_i' \geq \tau_i'' \leq \tau_{ki}$. The first deviation of $f_{ki}^0 = 0$ is:

$$le^{-l\lambda_i \tau_{ki}} + l\lambda_i \cdot e^{-l\lambda_i \tau_{ki}} \cdot (-l\tau_{ki}) = 0 \tag{5.5}$$

then we have $\lambda_i = (l\tau_{ki})^{-1}$. The second derivative of f^0_{ki} is less than zero, so $\max(f^0_{ki}) = (e \cdot \tau_{ki})^{-1}$. Similarly, we have $\max(f'_i) = (e \cdot \tau'_i)^{-1}$. Since $\tau_{ki} \leq \tau'_i$, we conclude that $\max(f^0_{ki}) \geq \max(f'_i)$. □

Note that the exponential part of f'_i decreases more rapidly than the same part of f^0_{ki}, which means that our cooperative caching has at least the same performances for highly popular videos, and better performances for middle popular videos. Experimental results confirm this theoretical analysis.

In conclusion, both parallel and series models are less efficient than a cooperative policy. Our approach, which combines cooperative and series approaches, is hence expected to outperform the classic CCN policy.

5.6 Simulations

We now show the results obtained from the implementation of our cooperative caching policy into the CCNx code.

5.6.1 Platform Setup

Our modified CCNx prototype was deployed by the platform CCNxProSimu on 40 machines with dual 2.70 GHz Pentium processor and 4 GB RAM. Details of the platform will be given in section 5.7. Each machine used Ubuntu 10.04 system and was connected to a switch via 100 Mbps ethernet card. The cache size of each ccnd handle is set by the environment variable *CCND_ CAP*. Various Linux shell strips are used to run the CCNx prototype concurrently on different machines.

Some minor changes are inserted into our augmented CCNx prototype in order to analyze the overhead yielded by the cooperative caching: a global counter adds up the number of messages sent by each ccnd handle; a timestamp carried by each interest and data packet to count the RTT of content retrieval. Note that the elapsed time recorded is not the real time but the average link latency offered by the applied network topology.

As our focus is on the performance of the cooperative caching policy, the assignment of label for each CR is pre-computed separately before the launch of ccnd handle. Both the determined label and k become input parameters of the *ccnd* function. As soon as *ccnd* handles have established the network, they use *ccncoput* to publish their labels, then execute *ccncoget* until their CRTs are completed.

Once the CRs that are chosen as point of presents (PoP virtually connecting with content providers) finish the establishment of CRT, they publish all available content using *ccnsendchunks*. Thereafter, edge CRs (virtually connecting with end users) execute the function *ccncatchchunks* according to the user behavior in different applications to generate traffic. More details in the following.

5.6.2 Network and Service Configuration

The topology of the tested ISP network was the European Backbone *Ebone* [SMW02]. Every machine worked as a CR. Among the 40 routers, 20 of them acted as edge routers, with the responsibility to emit the requests from 10,000 end users, and 3

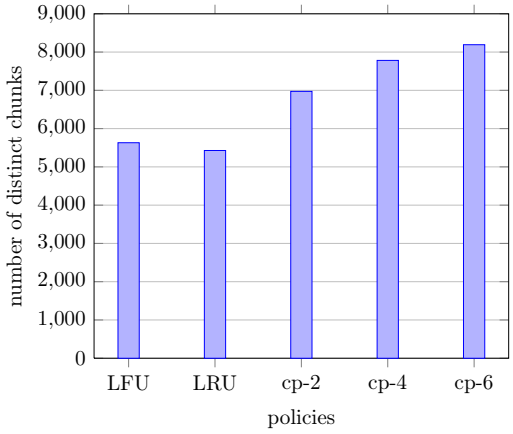


Figure 5.5: Cache diversity for the VoD service

routers were PoPs (virtual servers). The cache capacity of every router was limited up to 1,000 segments.

We evaluated our augmented prototype for two popular applications: VoD and time-shifted TV. The basic data unit of video streaming in both applications was a segment, which contains one minute video playback. We give later some details about the settings we chose for both applications.

We evaluated five caching policies: LRU, Least Frequently Used (LFU), and our cooperative CCN caching where k is 2, 4 and 6. We measured

- the **total caching diversity** by counting the number of distinct segments that are stored in the network. The more distinct segments are stored, the better is the cooperative caching system. The maximal caching diversity is 40,000 segments.
- the **per-video caching diversity** is the percentage of cached segments (including replicas) belonging to each video (or channel).
- the **ISP-friendliness** of the policy by measuring the number of requests that are treated by servers outside the network. The lesser is the number of requests, the friendlier is the caching policy.

5.6.3 VoD Service

Servers initially published all segments for the 5,000 available videos. The size of each video varied uniformly from 60 to 120 segments, so the total number of segments was around 450,000. User behavior (number of users to activate and the daily access pattern) followed the measurement results from [YZZZ06]. Once a client was activated, it chose a video based on a Zipf law with the skew factor equal to 1. The duration for each session was as follows: 50% of sessions ended in 10 minutes, 75% of them stopped in 25 minutes, 90% of them terminated in 50 minutes, and the remaining sessions lasted until the end of the video. We run our simulation for 10,000 minutes, *i.e.* about one week.

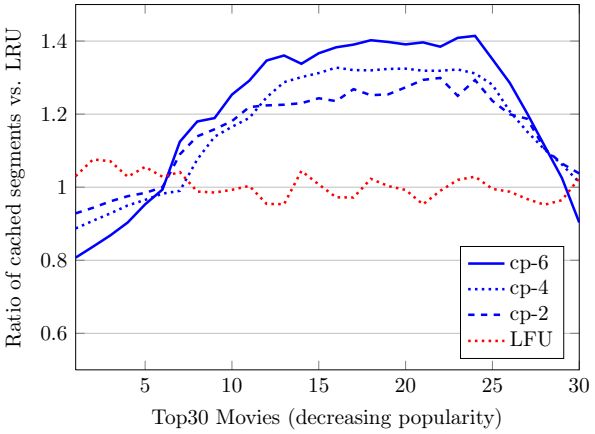


Figure 5.6: Stored segments for the most popular movies of the VoD application.

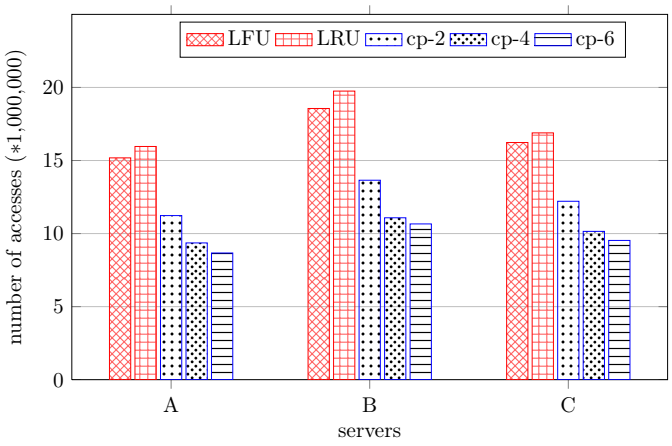


Figure 5.7: Number of times each server is accessed for the VoD service

We first present the caching diversity (measured at the end of the simulation) in Figure 5.5. As we expected, our cooperative scheme outperforms the traditional non-cooperative caching policies. The diversity augments regularly with the increment of k, but the overhead for CCS consistency increases as well. When $k = 6$, the caching diversity reaches $8,200$ segments, that is, the cooperative cache with $k = 6$ is nearly 1.5 times the diversity of the basic LRU policy.

For the per-video caching diversity, we focused on the 30 most popular videos. We compare to the LRU policy, so, for each video, we compute the ratio of the number of cached segments with a policy to the number of cache segments with the LRU policy. When this ratio is more than 1, this policy caches more segments than LRU, and vice versa. As seen in Figure 5.6, LFU caching polices do not differ much from the basic LRU. For the cooperative caching, the number of stored segments from the first and second most popular videos decreases, while it significantly increases from the 10th to the 25th most popular videos. That is, the aforementioned higher diversity focuses on the segments from middle-popular videos. The highest benefit of these films reaches 40% for $k = 6$. These experimental results are consistent with our theoretical analysis.

The most important result is the ISP-friendliness of policies. See Figure 5.7. We observe that the LFU policy performs slightly better than the LRU policy. The improvement is however not large (around 6%). On the contrary, the improvement obtained by our cooperative caching is impressive: in average, servers should upload about $1,750$ segments by minute with LRU, and only 960 with cooperative caching and $k = 6$. In other words, ISP reduces by more than 45% the cross-domain traffic. Even the implementation of the quite basic *cp-2* policy reduces the traffic by more than quarter in comparison to LFU.

5.6.4 Catch-up TV

We used the synthetic model of [LS10] for modeling the behavior of users of catch-up TV. We supposed that 20 channels are supported by an ISP. The channel popularity follows again the Zipf distribution [QGL+09]. A TV stream of each channel is divided into *programs*. The popularity of programs decreased with time. The number of activated clients varied from 20 to 180. Every client get assigned a role: half of the clients are *surfers* (watch a same program during 1 or 2 segments before to switch to another program), 40% of them are *viewers* (switch after a duration uniformly chosen between 2 and 60 minutes), and the rest 10% are *leavers* (stay on a program during more than 60 minutes). The duration of our simulation was also one week, and 200,000 segments were produced during this period.

The caching diversity of our cooperative CCN caching is again far larger than that of non-cooperative policies. In Figure 5.8, the gain reaches 30% for $k = 6$. In comparison to VoD, the gain is slightly lower, due to the higher number of distinct produced segments. We show in Figure 5.9 that the middle-popular channels are also the most cached, with a gain reaching around 20% compared with the basic LRU policy when $k = 6$. Oppositely, the number of stored segments for channels 1 and 20 decreases around 10%. Finally, we demonstrate in Figure 5.10 the ISP-friendliness of our cooperative caching policy with even more impressive gains.

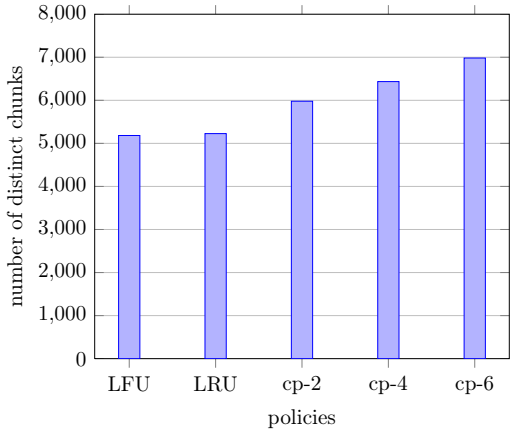


Figure 5.8: Caching diversity for the time-shifted TV service

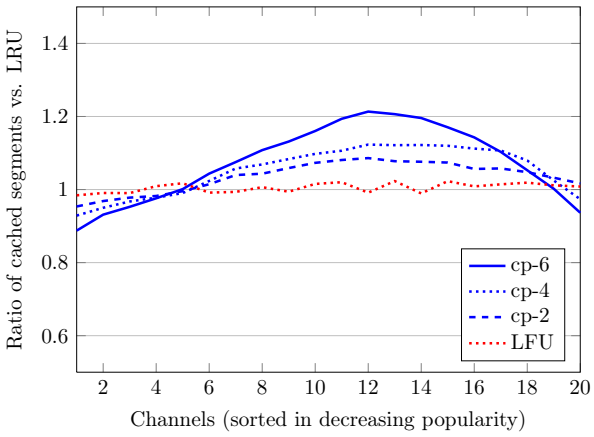


Figure 5.9: Stored segments for the channels of the time-shifted TV service.

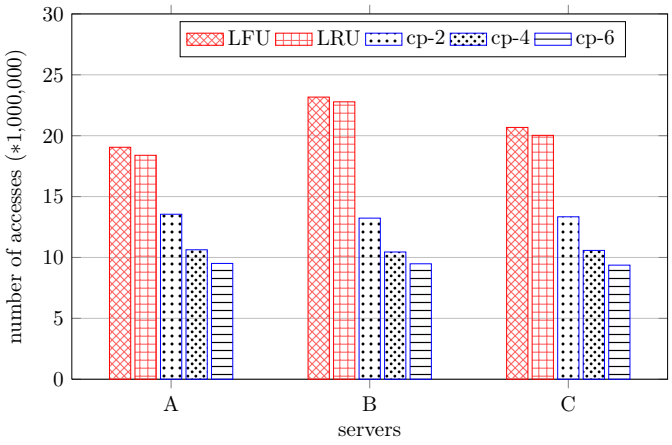


Figure 5.10: Number of times each server is accessed for the time-shifted TV

5.6.5 Overhead

		LRU	*CP-4*	*overhead*
VoD	*Average response time (ms)*	337.1	386.6	+14.5%
	Total number of messages ($\times 10^9$)	3.712	4.183	+12.7%
Time-shifted TV	*Average response time (ms)*	351.5	383.8	+9.1%
	Total number of messages ($\times 10^9$)	4.217	4.508	+6.9%

Table 5.1: Latency and Message Overhead

Our cooperative caching policy can be suspected to produce extra traffic within the ISP network, and to increase latency. We measured the total number of messages in both services, and we compared the average response times for each request. We combine results in Table 5.1. Our cooperative caching causes neither a significant degradation of the Quality of Experience, nor a network flood.

5.7 Design of CCNxProSimu

The task of the CCNxProSimu is to investigate the performance of CCN in a realistic environment. It is developed in *Bash (Linux Shell Script)*. The main idea of the platform is to use a centralized control machine to deploy the compiled CCNx prototype and the network traffic on each CR via *SSH*. Here, one CR is in fact a real machine equipped with the Ubuntu operating system. The whole program consists of five parts:

- *Main program* written in Bash is responsible for the deployment of CCNx prototype and network traffic, the establishment of CCN overlay, the control over the action of CRs and the collection of the simulation result.
- *CCNx prototype* is the packet of executable commands including *ccnd, ccndc, ccnsendchunks, ccncatchunks, etc.*
- *Network traffic* is composed by a set of files indicating the content that should be published or requested by each CR.
- *Overlay topology* contains the network map of the CCN overlay.
- *Output* stores the results retrieved from CRs.

5.7.1 Pre-configuration

To facilitate the execution of CCNxProSimu, some pre-configuration of the operating system and the user's account is necessary. Since the platform is deployed through SSH protocol, we should guarantee the controller can easily access the distributed CRs. Our solution is to use the couple of command *ssh-keygen* and *ssh-copy-id login@IP_CR*. The former one creates the public and private keys in various types, and the latter copies the local-host's public key to the remote-host's *authorized_key* file, so that the controller can establish the SSH connection to CRs without entering

the password each time. All the IP addresses of computers that are accessible for the controller should be listed in the "listMachines.tmp" file. Then the main program is able to choose a set of machines from the list and construct the CCN overlay.

Besides the configuration of underlying network, the input data including the network traffic, overlay topology should be indicated in distinct files. There are three columns of data in the topology map which represent the ids of the two CRs connected and the latency of the corresponding link. The network traffic is made up of a series of *ccnsendchunks* and *ccncatchunks* commands. Concretely, the *ccnsendchunks* commands publishing the available content are contained in the traffic files prepared for core CRs. On the other hand, the scripts for edge CRs comprise the lists of *ccncatchunks* commands that represent users' requests. Other minor configurations such as the user name of the SSH connection, the duration of the simulation, *etc.* are included in a file named "simCCNx.conf".

5.7.2 Scenario Description

The process of the simulation is described in figure 5.11. It is composed by the following steps:

1. The execution of the platform starts from the examination of the configuration file (simCCNx.conf). If the configuration file does not exist, the program stops immediately.

2. After the input parameters are successfully read by the program, it checks the connection of underlying network by *ping* the machines that are listed in listMachines.tmp. The program continues if there are enough machines available to construct the CCN overlay.

3. The main program calls a sub-function to generate the *connection scripts* for each CR according to the machine list and the topology map. A CR will execute the *ccndc* commands in the scripts to connect with its overlay neighbors latter.

4. When the scripts for overlay connection have been prepared, the corresponding configuration files and the executable files are copied to each CR. Then, the *ccnd* handle can be launched in these distributed machines.

5. Once all the *ccnd* handles have started, the controller makes the CRs run the connection scripts, to that the CCN overlay is established.

6. To inject the traffic into the network, core routers should publish available contain at first, then edge routers send their requests following the instructions given by their traffic files.

7. When all traffic are injected, which signifies that the simulation arrives at the end, the controller stops the *ccnd* handles on CRs. The result is output in two ways. The list of content that are cached in each CR is obtained by the *ccndumpnames* command. Other statistics are given by the standard output of C program, when the handle is shutting down.

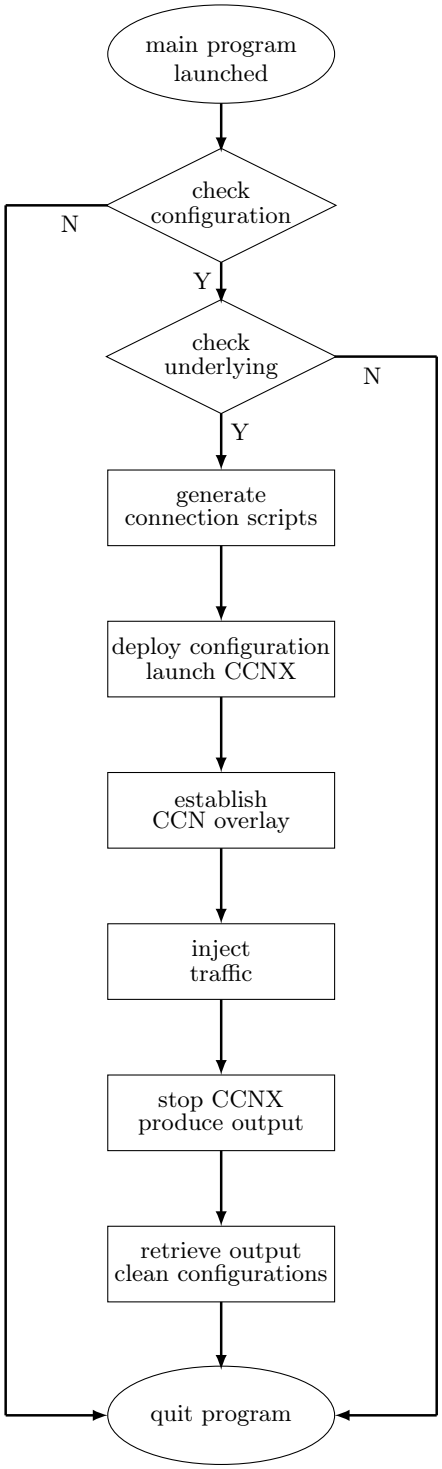


Figure 5.11: Execution Process of CCNxProSimu

8. The last step of the process is to retrieve the results from remote machines using *scp* command and clean the files that are copied at the beginning of the execution.

5.7.3 Input and Output of CCNxProSimu

simCCNx.conf

```
♯ Specify the path of input files and user login of the distant machines.
♯ MACHINE_LIST (list of machines to ping)
♯ NETWORK_NODES (network cartography,containing source and destination nodes)
♯ TRAFFIC_PATH (directory to store traffic configuration files, which have
♯ commands as ccnsendchunks and ccncatchunks )
♯ USER_LOGIN (login for ssh session)
♯ ARCHIVES_CCNx (CCNx bin directory)
♯ LIBCRYTO (library in case of incompatible version)
♯ ROUTER_CONFIG (script to create router configuration files)
♯ CCNx_DIR (working directory in distant machines)
♯ OUTPUT (directory to story result fils,in ./output by default)
♯ TRAFFIC_SENDER (routers who send chunks,eg:"0 3 4 19")
♯ ROUTER_CONF_TIME (waiting time for router configuration)
♯ TRAFFIC_CONF_TIME (waiting time for traffic injection time)
♯ KEY_DIR (directory to story key)
♯ This is an example of configuration file. Better use absolute path.

MACHINE_LIST=../listMachines.tmp
NETWORK_NODES=../carte.tmp
USER_LOGIN=yxu
TRAFFIC_PATH=../traffic_test
ARCHIVES_CCNx=../ccnx_0.6.0.tar.gz
LIBCRYTO=../libcrypto.so.0.9.8
ROUTER_CONFIG=../router_config/routercreate.sh
TRAFFIC_SENDER="0"
ROUTER_CONF_TIME=4
TRAFFIC_CONF_TIME=10
KEY_DIR=../.ccnx
```

Figure 5.12: Example of configuration file

In CCNxProSimu, system parameters are configured in the file simCCNx.conf. It contains the user name of SSH connection, the length of each part of the simulation, the location of other configuration files and sub-programs. An example is given in figure 5.12.

After the configuration file is successfully loaded, the CCNxProSimu starts to

carte.tmp

```
♯ Source Destination Delay
0 1 29
0 2 3
0 3 2
0 4 5
1 8 18
⋮
```

Figure 5.13: Example of network map

traffic-router-1

```
./ccncatchunks ccnx:/ccnx.org/film_35/chunk_1
./ccncatchunks ccnx:/ccnx.org/film_35/chunk_2
  sleep 5
./ccncatchunks ccnx:/ccnx.org/film_35/chunk_3
./ccncatchunks ccnx:/ccnx.org/film_1/chunk_1
./ccncatchunks ccnx:/ccnx.org/film_1/chunk_2
⋮
```

Figure 5.14: Example of traffic

res_ccnx.log_1

```
ccnx:/ccnx.org/film_23/chunk_1
ccnx:/ccnx.org/film_11/chunk_2
ccnx:/ccnx.org/film_36/chunk_3
ccnx:/ccnx.org/film_1/chunk_1
ccnx:/ccnx.org/film_1/chunk_5
⋮
```

Figure 5.15: Segments stored in CS

access_number_0.txt

♯ Access number	Average delay
196854	83.72

Figure 5.16: Server load & RTT

deploy the CCNx prototype and exchange messages following the network map and the traffic scripts. The examples of these categories of inputs are shown in figure 5.13 and 5.14. Here CR_1 is a edge router so that its traffic is a set of *ccncatchunks* commands representing the received requests from clients. The *sleep* command is used to set the synchronization between CRs.

As we mentioned before, the result of the simulation is obtained by the *ccndump-names* command and the standard output of the C program. Examples of the two output files are displayed in figure 5.15 and 5.16. The former one lists the video segments that cached by CR_1 and the latter one gives the number of requests that arrive at $server_0$ and the average latency of these arrived interest.

5.8 Summary and Conclusion

A key aspect of Content-Centric Network is that each router is endowed with caching capability, which permits the storage of content in multiple small equipments. While the current caching policy is not optimal, we introduced in this work the first lightweight cooperative caching for CCN. The policy is specially designed for large scale video stream delivery in an intra-domain environment. Since the cooperation occurs among a small group of CRs, only local knowledge is necessary for the execution, which guarantees its scalability. We detailed its deployment and the augmented CCN caching

protocol. The advantage of the cooperative caching is confirmed by theoretical analysis on simple network models. Particularly, we implemented the cooperative caching in the CCNx prototype, and performed extensive experiments under realistic network conditions. The experimental results highlight the remarkable benefit of the proposition that yields acceptable extra overhead.

Our results demonstrate the benefits one can expect from slightly smarter policies than LRU. As it is the first study about cooperative in-network caching, we see a lot of interesting challenges here. We presented here one cooperative policy, but various other strategies can be envisioned (despite we pointed out some critical requirements related to the characteristics of CCN). In particular, one can expect to leverage on the topology of ISP to produce dedicated more efficient policies. The implementation of our strategy in a real router, with the consequent issues of hardware management, is also in our future works.

Chapter 6

Stochastic Model for In-Network Caching Policies

This chapter is the conjunctive work of chapter 5. While the last chapter treats the cache decision problem of CCN in-network caching, the current chapter devotes the effort in ameliorating the replacement decision policy. Concretely, we first introduce a replacement decision policy called Least Recently/Frequently Used (LRFU), which takes both the recency and the frequency of the incoming requests into account. We present a set of simulation based studies about the policy. Then, we derive an approximation model as an efficient tool to investigate the LRFU cache. According to the observation of the approximation result, we propose a new multi-γ caching replacement policy that improves about 16% the overall in-network caching hit-ratio.

6.1 Introduction

Recent works on Information Centric Networking enable the exploitation of the caching resources in the new generation of routers Content Routers (CR). The research in this area has recently flourished [PCL+11, CGP11, CGMP11, WGMK11, LS11]. Unfortunately, only a basic Least Recently Used (LRU) strategy implemented on every CR has been proposed. More generally, the research community lacks methods for analyzing and evaluating caching policies (other than LRU) in generic multi-cache topologies.

Our contributions are twofold. First, we present an analytical tool that approximates in generic in-network caching the performances of the caching policies that are based on the analysis of both recency and frequency of requests. To validate this analytical tool, we compare the theoretical performances of CRs to the ones we obtained from simulations.

Second, we present a *multi-policy* in-network caching, where every CR implements its own caching policy according to its position in the network. We derive from the results that we obtained with our analytical tool a simple method to determine the proper caching policies for the CRs. We demonstrate the interest of our multi-policy approach by implementing a network of CCNx nodes in the context of a VoD application. As presented in [CGMP11], we assume that the operator has reserved

a part of the caching resources of CRs for this application. Compared to LRU, the performances of every single CR increase by up to 16%, which results in a globally better in-network cache.

We depict rest of the chapter in the following order. We present our simulation based study of LRFU caching performance in section 6.2. The approximation model for multi-cache LRFU policies is derived and validated in section 6.3. Thereafter, in section 6.4 we highlight the usefulness of our approximation. Section 6.5 extends our basic LRFU approximation so that it can approximate the caching performance with changing object popularity. Finally, the work is concluded in section 6.6.

6.2 Simulation Based Study of LRFU

In this section, we recall the principles of the LRFU caching policy in section 6.2.1. Then, we detail its performance on both single cache and multi-cache cases.

6.2.1 Introduction to LRFU Policy

A cache policy is intended to keep copies of the objects that are more likely to be requested in the future. While the LRU caches the objects that have been more recently requested, the LFU stores the objects that have been more frequently requested. Both LRU and LFU policies suffer from the sidedness of their analysis of the more recent accesses. The LRFU policy [LCK+01] aims combining LRU and LFU. Each object is associated with a *Combined Recency and Frequency* (CRF) value, which aims at characterizing the likelihood that it will be accessed in the near future. The computation of CRF uses a function $f(x) = e^{-\gamma x}$ to weigh the importance of the most recent occurrences for an object. The *weighing parameter* γ, which takes its value from 0 to 1, allows a trade-off between recency and frequency such that the more recent accesses become prevalent when γ increases. The computation of the CRF of an object b at time t_b is given below:

$$C_{t_b} = \begin{cases} f(0) + f(t_b - t'_b) * C_{t'_b}, \text{if } b \text{ is known} \\ f(0), \text{if } b \text{ is unknown} \end{cases} \tag{6.1}$$

where t'_b is the timestamp of the latest access to b [LCK+01]. Eq.6.1 gives more weight to the more recent requests since $f(x)$ is monotonically nonincreasing. On the other hand, $C_{t'_b}$ takes into account the previous accesses so that successive requests contribute to the CRF value. The extreme values 0 and 1 lead to the classical LFU and LRU caching policies respectively.

6.2.2 Single Cache Performance

We first simulated the behavior of *one* cache implementing the LRFU caching policies. There were 45,000 available objects in the system. Typically objects are one-minute video segments in a VoD service with 500 movies. The popularity of objects followed a *Zipf distribution* with a skew factor equal to one. Such distribution is an accurate model of movie requests in a typical VoD service [YZZZ06]. The size of the cache varied from 100 to 1,000, which seems reasonable values, with respect to the limited

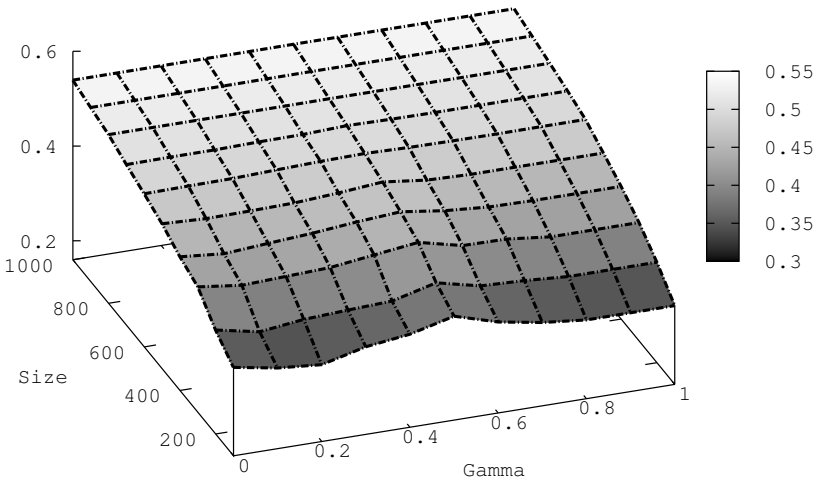


Figure 6.1: Single-cache simulation with static object popularity (On the y-axis, absolute hit-ratio produced by simulation.)

storage of CRs and the fact that only a part of the storage is reserved for the application. Moreover, we do not keep the CRF value for all appeared objects in the cache. Once an object is evicted from the cache, its CRF value is also removed. Therefore, the cache is not a "perfect" LRFU cache. The consideration is that in a real world application maintaining CRF values for all appeared objects is impossible.

In Fig. 6.1 we display the absolute hit-ratios produced by the simulation. The cache with the size of 100 objects is able to satisfy up to 30% of requests asking for 45,000 objects, depending on the value for γ. When the cache size augments to 1,000 objects, the cache hit-ratio reaches 54%.

Actually, Fig. 6.1 reveals that for a single cache directly receiving requests from clients, the best configuration is to set γ around 0.5. When the cache capacity is quite limited, *i.e.*, 100 objects, the LRFU with $\gamma = 0.5$ outperforms the worst case $\gamma = 0.2$ about 17.6%. Under the same condition, the configuration $\gamma = 0.5$ works 9.5% and 16.2% better than basic LFU and LRU respectively. On the other hand, we find that for larger caches, the impact of the variation of γ turns to be small. All different configurations of γ yield similar average cache hit-ratio, although $\gamma = 0.5$ still performs slightly better than other configurations. The benefit is about 0.1% comparing with the worst γ configuration.

6.2.3 Multi-Cache Performance

We emulated the multi-cache network according to a real backbone topology (European Backbone *Ebone* [SMW02]). We chose 40 nodes interconnected with 108 bidirectional links from the original ISP map. Three nodes are servers storing all

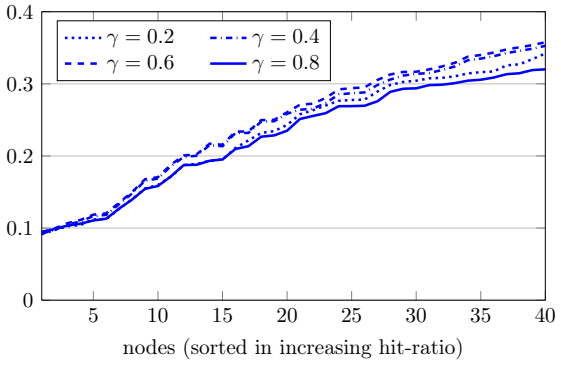


Figure 6.2: Multi-cache simulation with static object popularity.

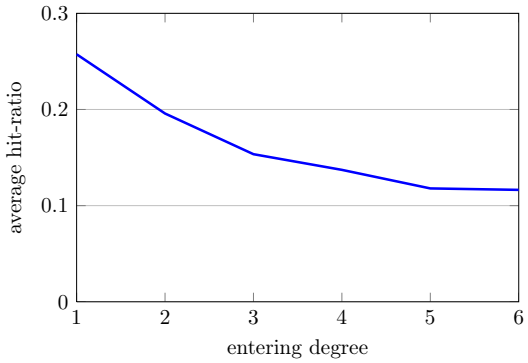


Figure 6.3: Impact of the number of incoming request streams.

objects. In order to emulate the traffic generated by end-users, every CR generated κ requests by round, κ being randomly chosen between 2 and 10. Missed requests were redirected along the shortest path from the origin node to the nearest server. The simulation is 10,000 rounds. Every CR had a cache capacity of 100 objects, which is the worst case according to the results on single cache.

The absolute hit-ratio of the multi-cache simulation is exhibited in Fig. 6.2. For all configurations of γ, we observe the same prominent disparity between the best and the worst hit-ratio, which ranges from 0.1 to 0.35. But the caching policy with γ at 0.4 and 0.6 performs better than their counterparts with more extreme values of γ for the routers that have a better cache hits. On the contrary, these latter perform slightly better for routers with low cache hits. In section 6.4, we will take advantage of this characteristic to improve the overall in-network caching hit-ratio.

In order to better understand why CRs experience widely different hit ratios, we conjecture that this is due to their "position" on the path between the client and the server. We first define the notion of *entering degree*: given the fact that when a CR cannot meet a request, this request is forwarded to the next CR r on the shortest path to its server, the entering degree of r is the number of shortest paths connecting front-end CRs with servers via r. We consider that all requests from clients to one edge CR as one request flow and we classify CRs by their *entering degree*. Front-end

CRs have an entering degree of 1 because they only receive traffic from end-users. A CR with a entering degree larger than 1 treats requests that have been missed by at least one previous CR. In our network configuration, the entering degrees range from 1 (for the Front-end CRs) to 6 (for the well-connected back-end CRs near the servers).

Then, in order to validate our conjecture, we analyze the impact of the entering degree on the hit ratio performance of the CRs in Fig. 6.3 (only LRU ($\gamma = 1$) policy is shown here although similar results have been obtained for other policies). This figure does indicate that the larger is the entering degree, the lower is the hit-ratio. CRs with a large entering degree that receive requests for less popular objects in the long-tail of the popularity distribution demonstrate the worst average hit-ratio. This indeed validates our conjecture.

6.3 Generalized Multi-cache Approximation

We now derive an analytical approximation of a single LRFU cache in section 6.3.1. More precisely, we approximate the stationary buffer hit probability for each object in the system, *i.e.*, the probability that each object is in the cache. The approximation is based on two assumptions: the input stream conforms to the *Independent Reference Model*, and the incoming requests for objects follow a *Poisson process*. Both assumptions are common in previous works related to cache approximation. We show in section 6.3.2 that these assumptions do not affect the quality of the approximation.

6.3.1 Derivation of the analytical approximation

An object is a generic piece of information. For simplicity, we assume that all objects have the same size (for example, an object can represent a video chunk in a VoD application). Let B be the total number of distinct objects and let S be the number of objects a cache can store. We consider an *Independent Reference Model* (IRM), so the successive requests form a sequence of *i.i.d* random variables, each distributed according to the access probability. We note α_b the access request probability of object b. See Table 6.1 for the notations.

γ	weighing parameter for LRFU ($\gamma = 0$ for LFU, $\gamma = 1$ for LRU)
C_{t_b}	CRF value of an object b a time t
$\overline{C(b)}$	steady value of the CRF of an object b
B, S	number of objects, capacity of the single cache
α_b, λ_b	access request probability and access rate of b
$P_{bb'}$	probability that b has larger CRF value than b'
P_{br}	probability that the last request was for object b
$\rho_b(s)$	probability that b is at the s-*th* position of the cache
$P_b(s)$	probability that b is within the s first positions of the cache

Table 6.1: Notations

We assume that the incoming requests to all objects form a Poisson process. The access rate for object b is defined as $\lambda_b = \lambda \cdot \alpha_b$, where λ is the total request rate.

Therefore, the time interval between two successive requests for a given object b follows an exponential distribution with expectation $\frac{1}{\lambda_b}$. Using the classical notation, the distribution for these time intervals relative to object b is given by:

$$g(x;\lambda_b) = \lambda_b e^{-\lambda_b x} \tag{6.2}$$

Keep in mind that our objective is to derive the probability that a requested object is in the cache. Our approximate derivation relies on three successive steps: i) We approximate the steady CRF value of each object; ii) From the steady CRF value, we derive the probability that an object b is before another object b' in the cache when a request arrives; iii) Then we use *fixed point method* to approximate the probability that each object is in the cache.

The first step of our derivation consists in obtaining an approximate value for the CRF value of an object b. Because we are considering the steady state value of CRF, we claim that we can approximate it by its expectation:

$$\begin{aligned}\overline{C(b)} \approx E(C_{t_b}) &= E\left(f(0) + C_{t'_b} \cdot e^{-\gamma(t_b - t'_b)}\right) \\ &= 1 + E\left(C_{t'_b} \cdot e^{-\gamma(t_b - t'_b)}\right)\end{aligned}$$

Since the time interval of two successive requests $(t_b - t'_b)$ for b is independent of $C_{t'_b}$, we have:

$$E(C_{t_b}) = 1 + E\left(C_{t'_b}\right) \cdot E\left(e^{-\gamma(t_b - t'_b)}\right) \tag{6.3}$$

As we assume that C_{t_b} and $C_{t'_b}$ are the steady CRF values of b, we conclude that $E(C_{t_b}) = E(C_{t'_b})$. From Eq.6.3 we obtain:

$$E(C_{t_b}) = \frac{1}{1 - E(e^{-\gamma(t_b - t'_b)})} \tag{6.4}$$

On the other side, the expectation of $e^{-\gamma(t_b - t'_b)}$ deduced from Eq.6.2 is:

$$E(e^{-\gamma(t_b - t'_b)}) = \frac{\lambda_b}{\lambda_b + \gamma} \tag{6.5}$$

Substituting $E(e^{-\gamma(t_b - t'_b)})$ in Eq.6.4 by Eq.6.5, we naturally obtain the approximation of $\overline{C(b)}$:

$$\overline{C(b)} \approx 1 + \frac{\lambda_b}{\gamma} \tag{6.6}$$

During the second step of our derivation, we compute the probability that, due to a demand for object b, object b' (which is currently in the cache) is pushed down. This probability is approximated by the probability that the new $C_{(t+x)_b}$ is larger than $C_{(t+x)_{b'}}$, where $C_{(t+x)_b} = 1 + e^{-\gamma x} * (1 + \frac{\lambda_b}{\gamma})$ and $C_{(t+x)_{b'}} = 1 + \frac{\lambda_{b'}}{\gamma}$:

$$\begin{aligned}1 + e^{-\gamma x} * (1 + \frac{\lambda_b}{\gamma}) &> 1 + \frac{\lambda_{b'}}{\gamma} \\ x &< \frac{\ln(\frac{\gamma + \lambda_b}{\lambda_{b'}})}{\gamma}\end{aligned} \tag{6.7}$$

Then the probability for b to possess the larger CRF value than b' is:

$$P_{bb'} = P(x < T) = 1 - e^{-\lambda_b T} \tag{6.8}$$

where

$$T = \left(\frac{\ln\left(\frac{\gamma+\lambda_b}{\lambda_{b'}}\right)}{\gamma} \right)^+,$$

and

$$(v)^+ = \begin{cases} v & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases}$$

The third step of our derivation consists in deriving the steady state distribution of the cache's content. Using the previous result, we derive the probability that b is inserted within the first s positions of the cache upon the arrival of a new request by iteratively calculating the following conditional probability:

$$\begin{aligned} P_b^n(s) =& \alpha_b \cdot \left(P_b^{(n-1)}(s) + (1 - P_b^{(n-1)}(s)) \cdot \sum_{b' \in B, b' \neq b} \frac{\rho_{b'}^{(n-1)}(s)}{1 - \rho_b^{(n-1)}(s)} \cdot P_{bb'} \right) \\ &+ \sum_{b' \in B, b' \neq b} \alpha_{b'} \cdot \left(P_b^{(n-1)}(s-1) + \rho_b^{(n-1)}(s)(1 - P_{b'b}) \right) \end{aligned} \tag{6.9}$$

and the equation:

$$\rho_b^n(s) = P_b^n(s) - P_b^n(s-1), s = 2, 3, \ldots, S, \tag{6.10}$$

where n is the number of iteration. We do not know $P_b^n(s)$, but starting our iterative computation from an arbitrary cache distribution, the above equations allow to converge towards the steady state distribution of the cache's contents.

In other words, we obtain the probability that each object is in the cache heap by iteratively solving the equations Eq.6.8, Eq.6.9 and Eq.6.10 for each $b \in B$, assuming an arbitrary cache distribution at the first iteration:

$$\begin{aligned} P_{bb'} =& 1 - e^{-\lambda_b T} \\ P_b^n(s) =& \alpha_b \cdot \left(P_b^{(n-1)}(s) + (1 - P_b^{(n-1)}(s)) \cdot \sum_{b' \in B, b' \neq b} \frac{\rho_{b'}^{(n-1)}(s)}{1 - \rho_b^{(n-1)}(s)} \cdot P_{bb'} \right) \\ &+ \sum_{b' \in B, b' \neq b} \alpha_{b'} \cdot \left(P_b^{(n-1)}(s-1) + \rho_b^{(n-1)}(s)(1 - P_{b'b}) \right) \\ \rho_b^n(s) =& P_b^n(s) - P_b^n(s-1), s = 2, 3, \ldots, S, \end{aligned}$$

The iteration is stopped when the probability distribution meets a predefined precision threshold. In our implementation, we take the mean square error of the probabilities calculated by two consecutive iterations as the halting criterion. When the mean square error is less than the predefined value ϵ, we stop the iteration. This

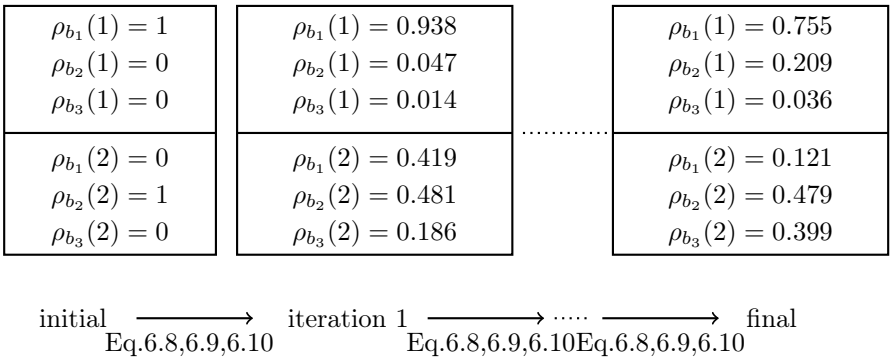


Figure 6.4: Example of cooperative cache

procedure has converged quickly in all the tests we made, although this is not a formal proof for convergence.

The iterative computation process is illustrated by an small example with three objects and a 2-objects cache in Fig.6.4. The identifier of an object is given according to its access rate (*i.e.*, b_1 is the object with the largest access rate). The initial state is given below:

$$\rho_b^{(0)}(s) = \begin{cases} 1 & \text{if } b \text{ has the } s\text{-th largest access rate} \\ 0 & \text{otherwise.} \end{cases}$$

Please note that the CRF value of objects cannot be computed when $\gamma = 0$, that is, when the LRFU policy becomes LFU. This drawback is not surprising as the CRF value degenerates to the total number of accesses when $\gamma = 0$ and $f(x) = 1$. We propose to set the CRF of an object b as $\overline{C(b)} = P_{br}$ when the caching policy is LFU. Indeed, P_{br} corresponds to the ratio of requests for b among all requests.

This approximation also presents limitations to treat some extreme case. For example, when access rates for all the objects in the cache system are identical, this approximation is unable to decide which object has the higher probability to reside at the first position of the cache. This extreme case may theoretically occur, however, the tie is unlikely to exist in reality. Hence, the shortcoming does not impact the reliability of the approximation when it is used to approximate the performance of a practical system.

This approximation model for a single LRFU cache policy, which computes the probability of existence of each object in the cache, is compatible with the multi-cache model proposed in [RKT10]. Therefore, the algorithms proposed in this latter model can be directly used to obtain an approximation of any LRFU policy in any multi-cache topology.

6.3.2 Validation of the approximate model

We show here that the hit-ratio obtained from our approximate model matches the hit-ratio obtained by simulation.

Fig. 6.5 shows the ratio of the hit-ratio of the approximation to the hit-ratio of the simulation in the case of a single cache. The closer to 1, the better is the

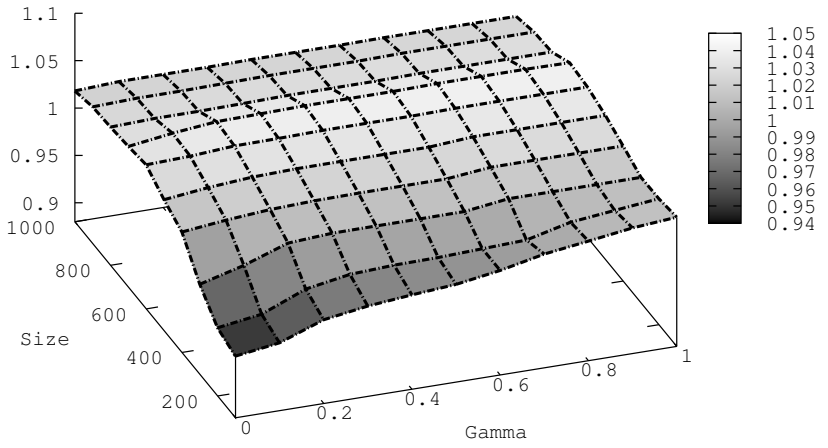


Figure 6.5: Comparison of approximation and simulation for a single cache with static object popularity (On the y-axis, ratio of hit-ratio approximation to hit-ratio simulation. The same in Fig. 6.6 and Fig. 6.11)

approximation. Fig. 6.5 shows than the approximation model closely matches the simulation results. The error is larger than 5% in only one situation: when LRFU is close to LFU and the cache size is small. Otherwise, the error is less than 4%. Except for the smallest cache size, where ratio varied from -5.5% for $\gamma = 0$ to $+1.6\%$ for $\gamma = 1$, the impact of γ on the approximation ratio was negligible.

We think that the approximation model overestimates the single cache performance because the probability of each object is larger than 0 in the approximation, whereas most of the unpopular objects experience a null hit-ratio in the simulation.

We now validate the multi-cache LRFU approximation. The network topology and the tested parameters are the same described in section 6.2.3. Fig. 6.6 shows the ratio of the hit-ratio of the approximation to the hit-ratio of the simulation for all routers and four different values of γ, ranging from 0.2 to 0.8. Despite we were in the worst scenario for cache size (100), the error due to approximation was never more than 10% for a given CR. The error was below 5% for all CRs when $\gamma = 0.2$ and $\gamma = 0.4$, and for 36 out of 40 CRs when $\gamma = 0.6$. Moreover, for a given γ, the approximation performed uniformly on all routers. The CRs with the lowest hit-ratio were affected by a larger ratio degradation than the CRs with the highest hit-ratio, but this degradation difference stayed reasonable (less than 5%).

6.4 Optimizing LRFU in Multi-cache Network

To our best knowledge, the global optimization of a multi-cache network is still an open problem. Although the routers which directly receive requests from end-users

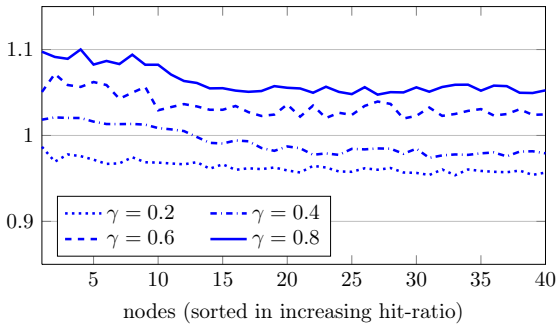


Figure 6.6: Multi-cache approximation performance with static object popularity.

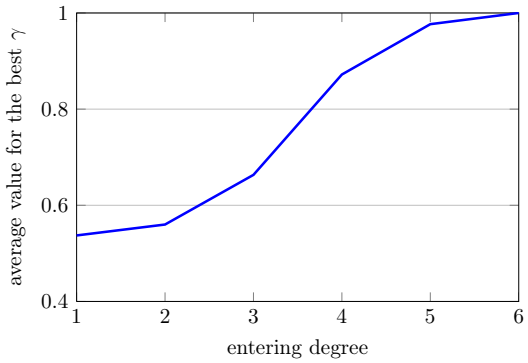


Figure 6.7: Best configuration of γ for CRs with different entering degree

("front-end" CRs) work as classic caches, the behavior of "back-end" CRs, which are between the front-end CRs and the server, have to serve an incoming traffic that is expunged from the most popular requests, which makes their behaviour hard to assess and control. Usually, all routers apply the same caching policy, irrespective of their location in the network. In the present section, we assess the potential gain of varying the caching policy according to the location of the CRs. In our context, considering different caching policies for different CRs consists in selecting different values of γ for CRs with various entering degree.

6.4.1 Optimizing cache policies on the Ebone example

We assess the efficiency of various policies through our approximate analytical model of cache behaviour. Fig. 6.7 gives the value of the parameter γ for which a CR with a given entering degree experiences the best hit-ratio in the Ebone network topology considered previously. The front-end CRs reach their best hit-ratio for $\gamma = 0.53$, which is consistent with the results originally presented in [LCK+01]. When the entering degree grows, the best γ increases and ultimately, the best caching policy is LRU ($\gamma = 1$) for CRs with entering degree of 6.

This shows indeed that CRs with different entering degree reach their best performance with various values of γ. This suggests that, instead of running the same policy on all CRs, one should carefully select the value of γ for CRs at distinct positions in

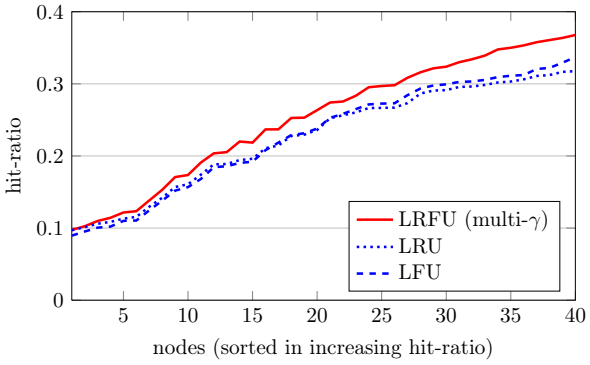


Figure 6.8: Hit ratio for different cache policies

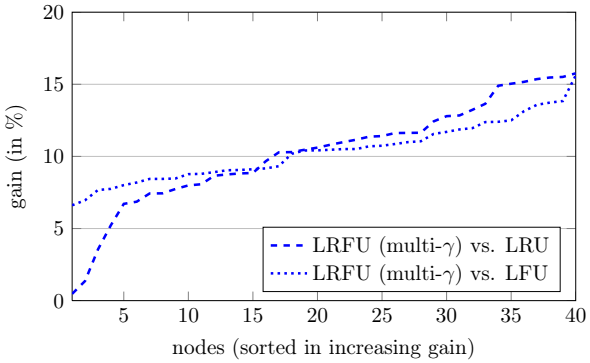


Figure 6.9: Gain in terms of hit ratio of LRFU (multi-γ) versus LRU and LFU

the network, so that the LRFU with suitable γ value can maximize the performance of each single cache and thus improve the performance of the multi-cache network.

6.4.2 Relaxing the Independence assumption

Keep in mind that the the analytical approximation relies on assuming that successive requests are independent and occur as a Poisson process. In a realistic network, this is not the case. In the present section, we shall simulate a network (the Ebone [SMW02]) that implements in-network caching, and is used to support a Video-on-demand (VoD) service.

Servers initially publish all segments for 500 available videos. The size of each video varies uniformly from 60 to 120 segments, so the total number of segments is around 45,000. User behavior (number of users to activate and the daily access pattern) follows measurement results from [YZZZ06]. Once a client is activated, it chooses a video based on a Zipf law with the skew factor equal to one. The duration for each session is as follows: 50% of sessions end last less than 10 minutes, 75% less than 25 minute, 90% less than 50 minutes, and the remaining sessions last till the end of the video.

The LRFU policy is implemented using the open source CCNx demo. Our mod-

ified CCNx prototype is deployed on 40 machines with dual 2.70 GHz Pentium processor and 4 GB RAM. Each machine uses a Ubuntu 10.04 system and is connected to a switch via 100 Mb/s ethernet card. Every machine works as a router. Among the 40 routers, 20 of them act as edge routers, with the responsibility to emit the requests from 1,000 end users, and 3 routers are servers. The cache capacity of every router is limited to 100 segments.

The γ value is selected according to the results given in Fig. 6.7 and the topology of the network. For the 20 edge routers, γ is set to 0.53, and the 3 CRs connected with servers used directly the LRU policy. Then, the other 17 intermediate CRs are classified according to their entering degree. Each group of CRs is assigned respectively 4 different γ values: 0.56, 0.66, 0.87, 0.98. This policy is denoted "LRFU multi-γ". We run our simulation for 10,000 minutes, *i.e.* about one week.

We consider two aspects of the multi-cache's performance: (i) **The single cache hit-ratio** which measures how often the requested object is within a cache. (ii) **The number of access to servers** which measures the load offered to the servers (keep in mind that using in-network caching aims at reducing this load).

Fig. 6.8 compares the hit-ratio for three policies (LRU, LFU and LRFU multi-γ). As expected, front-end CRs achieve a higher hit-ratio than back-end CRs for all policies. For edge routers (CRs 21-40) LFU outperforms LRU whereas LRU is slightly better than LFU for the back-end routers. However, LRFU multi-γ is significantly better than both LRU and LFU, especially for front-end CRs. The gain, relative to hit ratio, yielded by LRFU multi-γ is explicitly displayed in Fig. 6.9. The highest gain in terms of hit ratio reaches 16% comparing with both LRU and LFU policies.

The amount of servers' access is shown in Fig. 6.10. LFU is slightly more efficient than LRU for the 3 servers, but is significantly less efficient than LRFU multi-γ. In-network caching system using LRFU multi-γ outperforms LRU by at least 15% in terms of servers' access.

This simulation study confirms the results obtained in section6.4.1, which rely on independant requests. In this more realistic scenario where successive demands are not independent since they correspond to successive chunks of the same video, it is still beneficial to finely tune the γ value according to the location of the CRs relative to users and servers.

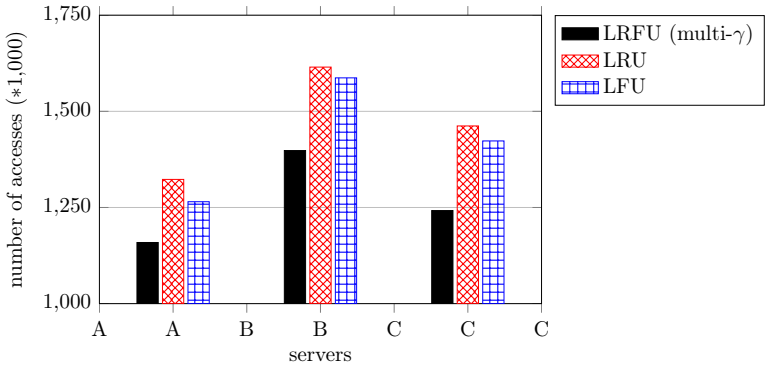


Figure 6.10: Number of times each server is accessed

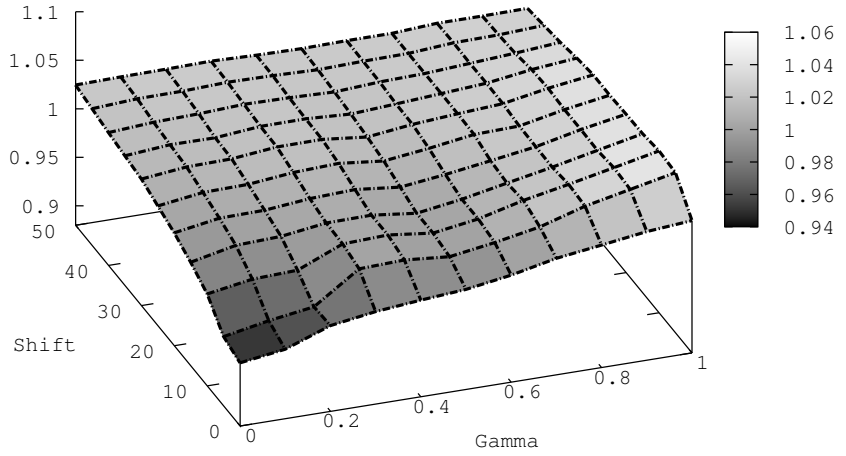


Figure 6.11: Comparison of approximation and simulation for a single cache with dynamic object popularity (On the y-axis, ratio of hit-ratio approximation to hit-ratio simulation).

6.5 Approximation Under Dynamic Popularity

On-demand streaming applications, especially catch-up TV, exhibit frequent changes in the popularity of objects. Our approximate model, as all other studies related to caching approximation consider a static popularity distribution.

In the present section, we slightly modify our approximate model in order to take into account the dynamic popularity changes. We assume that the popularity for the objects periodically change, but that the period duration is long enough for the system to reach a new equilibrium before a new popularity change.

The modified single cache approximation model is obtained as follows: the steady CRF value $\overline{C(b)}$ of an object b, which records the history of b's access pattern, is computed as in section 6.3.1 for a given set of popularity values whereas $P_{bb'}$, probability b pushes down b' is computed using the modified value for the popularity of b.

$$P_{bb'} = P(x < T) = 1 - e^{-\mu_b T}, \tag{6.11}$$

where μ_b is the new value for the access rate of b, whereas λ_b, the old value for the access rate of b is used to compute T.

We now validate our approximation under dynamic popularity context. The model for modifying the objects' popularity is the following: we first sort all objects in ascending order of their access rates, then we shift all access rates from objects to their immediate predecessor objects in the sorted list. Let consider that object b_i has the ith largest access rate λ_{b_i}. After the shift, μ_{b_i} will be equal to $\lambda_{b_{i+1}}$, which was the previous access rate for b_{i+1}. We iterate this a number of times that we express

through a percentage j of the number of objects B. That is, a j%-shift corresponds to a popularity change where object i has taken the access rate of object $i+j*B \mod B$. In other words, j% of the most unpopular objects have become the most popular ones. This evolution is believed to be particularly suitable for catch-up TV, where the most recent video segments are the most popular, and all other segments have a strictly decreasing popularity [LS10]. Moreover, since our approximation model requires a fixed number of objects, this algorithm allows to mimick the creation of objects by transforming objects with null popularity in popular objects.

Fig. 6.11 dhows the ratio of hit-ratio of approximation to hit-ratio of simulation. The error was again below 6% for all configurations except the extreme values $\gamma = 0$ and $\gamma = 1$ when the traffic was static. We observe that our approximation model performs especially well when the traffic changes are more brutal, a 50%-shift being a major disruption in the popularity of objects. In such configuration, the error is less than 3% whatever the value of γ.

6.6 Summary and Conclusion

Research on Information-Centric Networks currently lacks of tools for the analysis of network performance and delivered QoS. In this chapter, we address the evaluation of caching policies for in-network caching, which we believe is a major topic. We provide an analytical tool for the approximation of cache content distribution and hit-ratio for any multi-cache topology and for any LRFU caching policies based on both recency and frequency. This approximation model allows the analysis of a range of caching policies that fit ICN requirements.

Then, we show the advantage of fine-tuning the cache management policy according to the location of CRs relative to users and servers. This advantage is measured in terms of both hit ratio (which improves latency) and servers's load (which reduces network's costs). We thus demonstrate that finding suitable policy for each single cache improves the overall performance of the in-network cache system.

This work is a positive step in the analysis and the understanding of network of caches, with the promises of better network resource usage.

Future works include an in-depth analysis of the trade-off between caching and bandwidth usage, including the optimization of the caches size, location and management policies. This analysis should be carried out for both static and dynamic objects popularity.

Chapter 7

Conclusion

Recent Internet evolution, particularly the exploding HD video streaming traffic, calls for shifting current content delivery mechanisms. It necessitates the deployment of massive content servers deeper inside network operators' infrastructures so as to avoid congestion on peering links, reduce traffic across backbones, *etc.* Unwilling to allow CDNs to abuse their infrastructures, network operators gain the initiative by hatching their own content distribution service named *Telco-CDN*. Managing both the content delivery overlay and the underlying network enables ISPs to significantly improve the service quality. However, intelligent resource placement strategies are mandatory to reach the purpose. In this thesis, we sketch a practical Telco-CDN architecture based on explicit business incentive and concentrate on the optimal resource placement in this context. The objective is to provide valuable references, especially in the aspect of optimal resource placement, for Telco-CDN constructors and operators.

7.1 Synopsis

Aiming at optimizing the resource placement in Telco-CDN, we contribute in pushing various resource placement schemes to their limitations.

- **Application components allocation.** Chapter 3 investigates various schemes to allocate partitioned large-scale application in Telco-CDN system. We address the problem from an optimization and algorithmic point of view so as to achieve better cost efficiency for system operator. The resulting optimization problem, which we refer to as the k-Component Multi-Site Placement Problem (k-CMSP), applies to service distribution in a wide range of communication networking scenarios including Telco-CDN system. We provide a theoretical analysis of the problem's computational complexity, and develop an integer programming model that provides a reference results for performance benchmarking. On the algorithmic side, we present four approaches: an algorithm with approximation guarantee and three heuristics algorithms. The first heuristic is derived from graph theory on domatic partition. The second heuristic, built on intuition, admits distributed computation. The third heuristic emphasizes on fairness in cost distribution among the sites. We report simulation results for sets of networks where cost is represented by round-trip time (RTT) originating from real

measurements. For small networks, the integer model is used to study algorithm performance in terms of optimality. Large networks are used to compare the algorithms relatively to each other. Among the algorithms, the heuristic based on intuition has close-to-optimal performance, and the fairness heuristic achieves a good balance between single-site cost and the overall one. In addition, the experiments demonstrate the significance of optimization for cost reduction in comparison to a random allocation strategy.

- **Optimal video placement.** Since the network operator in Telco-CDN controls both the infrastructure and the content delivery overlay, it is in position to engineer Telco-CDN so that networking resources are optimally utilized. In chapter 4, we show that 1. it is possible to implement a quasi-optimal algorithm for the placement of video chunks into a telco-CDN. We present an algorithm, which is based on a genetic algorithm implemented on the MapReduce framework. We show that, for a national VoD service, computing a quasi-optimal placement is possible. 2. such push strategy makes sense because it allows to actually take into account fine-grain traffic management strategies on the underlying infrastructure. Our proposal re-opens the debate about the relevance of such "push" approach (where the manager of Telco-CDN proactively pushes video content into servers) versus the traditional caching approach (where the content is pull to the servers from requests of clients). Our proposal of a quasi-optimal tracker enable fair comparisons between both approaches for most traffic engineering policies. We illustrate the interest of our proposal in the context of a major European Telco-CDN with real traces from a popular Video-on-Demand (VoD) service. Our experimental results show that, given a perfect algorithm for predicting user preferences, our placement algorithm is able to keep aligned with LRU caching in terms of the traditional hit-ratio, but the impact of a push strategy on the infrastructure is almost half the one of a caching strategy.

- **Cooperative caching protocol in ICN.** As a revolutionary design of Telco-CDN, ICN can potentially reinforce the benefit by enabling network layer content oriented routing. In ICN, content router is in particular a key component that improves the system performance by caching popular objects. In chapter 5, we refer to the management of caches of a network of CRs as *in-network caching*. Since only a basic LRU policy implemented on every CRs has been proposed, our contribution is the proposition of a cooperative caching strategy in the context of Content-Centric-Network (CCN), a typical model of ICN. Same as before, the cooperative caching is designed for the treatment of large video streams with on-demand access. This caching strategy combines the traditional hash-based and directory-based cooperative caching schemes, and addresses the need of ISP by halving the cross-domain traffic. We illustrate first the changes that have to be brought to the CCN protocol in order to implement this strategy. Thereafter, we prove the advantages of this cooperative policy over standard non-cooperative policies in simple network structures. Finally, we describe an augmented version of the CCNx protocol implementing this policy, and we present a set of simulations, which have been conducted on an experimental platform for CCNx. Our cooperative caching protocol provides a prominent

benefit in term of the reduction of inter-domain traffic.

- **Analytical model for evaluating multi-policies.** Chapter 6 proceeds with the work of chapter 5 to further investigate the behavior of various caching policies other than LRU. Unfortunately, previous works related to the analysis of in-network caching in CCN have considered only the LRU. More generally, the research community lacks methods for analyzing and evaluating caching policies in generic multi-cache topologies. So we contribute in two aspects. First, we present an analytical tool that approximates in generic in-network caching systems the performance of the caching policies that are based on the analysis of both recency and frequency of requests. To validate this analytical tool, we compare the theoretical performance of CRs to the one estimated from simulations. Second, we present a *multi-policy* in-network caching, where every CR implements its own caching policy according to its location in the network. The results obtained with our analytical tool yield a simple method to determine the optimum caching policies for the CRs. We demonstrate the interest of our multi-policy in-network caching approach by implementing a network of CCNx nodes in the context of a VoD application. Compared to the single LRU policy, the multi-policy approach visibly increases the performance in terms of hit-ratio of the in-network caching system.

7.2 Future directions

In this thesis, we dealt with the optimal resource placement in Telco-CDN design. Our studies stretch both push-based and pull-based approaches to their limitations, and improve the system performance in aspects of reducing operation cost and delivering better service quality. Potential research that keeps up our work contains following directions.

Reinforcing the solutions for application allocation includes the design of approximation algorithms with better performance ratio; the identification of classes of cost functions where exact solutions are within reach in polynomial time; and allocating multiple components on one site and fault tolerance.

Improving the efficiency of optimal video placement requires the examination of various settings of genetic algorithm operators; the implementation on different video related applications and larger data sets; and the improvement on the MapReduce scheduling scheme so as to accelerate the computation.

Ameliorating the hit-ratio of in-network caching necessitates in-depth analysis of the trade-off between caching and bandwidth usage; the optimization of the caches size, location and management policies; and the realization of our strategies in real routers.

Bibliography

[AGKP98] V. Arya, N. Garg, R. Khandekar, and V. Pandit. Local search heuristic for k-median and facility location problems. In *Proc. of ACM-SIAM SODA'98*, pages 1–10, 1998. 17

[AGM03] Shingo Ata, Yoshihiro Gotoh, and Masayuki Murata. Replication strategies in peer-to-peer services over power-law overlay networks. In *Proc. of the 7th Asia-Pacific Network Operations and Management Symposium*, 2003. 17

[Bac02] Brent Baccala. Data-oriented networking. internet draft, ietf, 2002. 15

[Bar96] Yair Bartal. Probabilistic approximation of metric space and its algorithmic applications. In *In Proceedings of 37th Annual IEEE Symposium on Fuoundations of Computer Science*, 1996. 13

[BCF+99] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *IEEE INFOCOM*, 1999. 20

[BSB+13] G. Bertrand, E. Stephan, T. Burbridge, P. Eardley, K. Ma, and G. Watson. Use cases for content delivery network interconnection. IETF draft, Feb. 2013. draft-ietf-cdni-use-cases-10. xv, 3

[CG95] Boris V. Cherkassky and Andrew V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Integer Programming and Combinatorial Optimization*, 920:157–171, 1995. 51

[CG00] David.R. Cheriton and Mark. Gritter. Triad: A new next-generation internet architecture, 2000. 15

[CGMP11] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, and Diego Perino. Modeling data transfer in content-centric networking. Technical report, Orange Labs/Alcatel-Lucent, 2011. xix, 6, 93

[CGP11] Giovanna Carofiglio, Vinicius Gehlen, and Diego Perino. Experimental evaluation of storage management in content-centric networking. In *IEEE ICC*, 2011. 68, 72, 93

[CGST99] M. Charikar, S. Guha, D. B. Shmoys, and E. Tardos. A constant factor approximation algorithm for the k-median problem. In *Proc. of ACM STOC '99*, pages 1–10, 1999. 17

[CHR08] Jin Li Cheng Huang, Angela Wang and Keith W. Ross. Understanding hybrid cdn-p2p: Why limelight needs its own red swoosh. In *Proceedings of NOSSDAV*, 2008. 14

[CHW03] Chen-Lung Chan, Shih-Yu Huang, and Jia-Shung Wang. Cooperative cache framework for video streaming applications. In *IEEE ICME*, 2003. 21

[CIS] CISCO. Cisco ip next-generation network carrier ethernet design solutions overview. http://www.cisco.com/en/US/solutions/collateral/ns341/ns524/ns562/ns577/brochure_c02-496508.html. 15

[Cis10] Cisco. Visual Networking (VNI) Forecast 2009-2014. Technical report, Cisco Company, June 2010. 68

[CJL+11] Kideok Cho, Hakyung Jung, Munyoung Lee, Diko Ko, T.T. Kwon, and Yanghee Choi. How can an ISP merge with a CDN? *IEEE Communications Magazine*, 49(10):156 –162, oct. 2011. xvi, 4

[CLP+12] Kideok Cho, Munyoung Lee, Kunwoo Park, Ted Taekyoung Kwon, Yanghee Choi, and Sangheon Pack. Wave: Popularity-based and collaborative in-network caching for content-oriented networks. In *IEEE INFOCOM NOMEN Workshop*, 2012. 21

[CLS09a] Yiping Chen, Jimmy Leblet, and Gwendal Simon. On Reducing the Cross-Domain Traffic of Box-Powered CDN. In *ICCCN: 18th IEEE International Conference On Computer Communications and Networks*, August 2009. 36

[CLS09b] Yiping Chen, Jimmy Leblet, and Gwendal Simon. On reducing the cross-domain traffic of box-powered cdn. In *IEEE ICCCN*, 2009. 70, 72

[CS98] F. A. Chudak and D. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. Unpublished manuscript, 1998. 17

[CSJ+08] Bin Cheng, Lex Stein, Hai Jin, Xiaofei Liao, and Zheng Zhang. Gridcast: Improving peer sharing for p2p vod. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(4):1–31, 2008. 11

[CTW02] Hao Che, Ye Tung, and Zhijun Wang. Hierarchical web caching systems: Modeling design and experimental results. *IEEE Journal on Selected Areas in Communications*, 20(7):1305–1314, 2002. 19, 79, 80

[Dan08] György Dan. Cooperative caching and relaying strategies for peer-to-peer content delivery. In *Proc. of Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2008. 31

[DD10] Amogh Dhamdhere and Constantine Dovrolis. The internet is flat: Modeling the transition from a transit hierarchy to a peering mesh. In *ACM CoNEXT*, 2010. 68

[DD11] Amogh Dhamdhere and Constantine Dovrolis. Twelve years in the evolution of the internet ecosystem. *IEEE/ACM Trans. Netw.*, 19(5):1420–1433, 2011. xiv, 2

[DHT08] Tai T. Do, Kien A. Hua, and Mounir A. Tantaoui. Robust video-on-demand streaming in peer-to-peer environments. *Comput. Commun.*, 31(3):506–519, 2008. 11

[DLL+11] Jie Dai, Bo Li, Fangming Liu, Baochun Li, and Hai Jin. On the efficiency of collaborative caching in isp-aware p2p networks. In *IEEE INFOCOM*, 2011. 21

[DLLL12] Jie Dai, Fangming Liu, Baochun Li, and Jiangchuan Liu. Collaborative caching in wireless video streaming through resource auctions. *IEEE Jounral on Selected Areas in Communications (JSAC) Special issue on Cooperative Networking Challenges and Applications*, 30(2):458–466, 2012. 21

[DN08] Sachin Deshpande and Jeonghun Noh. P2tss: Time-shifted and live streaming of video in peer-to-peer systems. In *IEEE International Conference on Multimedia and Expo*, Honolulu, Hawaii, USA, June 2008. 11

[DT90] Asit Dan and Don Towsley. An approximate analysis of the LRU and FIFO buffer replacement schemes. In *ACM SIGMETRICS*, 1990. 19

[dU07] Özgür Ulusoy. Research issues in peer-to-peer data management. In *Proc. 22nd international symposium on Computer and Information Sciences*, 2007. 17

[DXC06] Catherine Rosenberg Dongyan Xu, Sunil Suresh Kulkarni and Heung-Keung Chai. Analysis of a cdn-p2p hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11(4), 2006. 14

[EPY97] David Eppstein, Michael S. Paterson, and Frances F. Yao. On nearest neighbor graphs. *Discrete & Computational Geometry*, 17(3):263–282, April 1997. 30, 32

[FCAB98] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *ACM SIGCOMM*, 1998. 20

[FCB+08] Peyman Faratin, David D. Clark, Steven Bauer, William Lehr, Patrick W. Gilmore, and Arthur Berger. Complexity of internet interconnections. *Communications & Strategies*, (72):51, 2008. xiv, 2

[FHKS02] Uriel Feige, Magnús M. Halldórsson, Guy Kortsarz, and Aravind Srinivasan. Approximating the domatic number. *SIAM J. Comput.*, 32(1):172–195, 2002. 31

[FRR12] Christine Fricker, PHilippe Robert, and James Roberts. A versatile and accurate approximation for lru cache performance. *arXiv.org*, Feb. 2012. 19

[Ful12] Scott M. Fulton. Vomcast on xbox: Does a cdn, by nature, threaten the internet? `http://www.readwriteweb.com/archives/comcast_on_xbox_does_a_cdn_by_nature_threaten_the.php`, March 2012. xv, 3

[GB06] Sushant Goel and Rajkumar Buyya. Data replication strategies in wide area distributed systems. *Enterprise Service Computing: From Concept to Deployment*, 2006. 17

[GDS+03] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *ACM SOSP*, 2003. 20

[GJ79] Michael R. Garey and David. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman & Co., 1979. 25

[GK99] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithmm*, 31:228–248, 1999. 17

[Gol10] Daniel Golding. The real story behind the comcast-level 3 battle. *Giga Om*, Dec. 2010. xv, 3

[Got10] Greg Goth. New internet economics might not make it to the edge. *IEEE Internet Computing*, 14, Jan 2010. 68

[GS09] Shahram Ghandeharizadeh and Shahin Shayandeh. A comparison of block-based and clip-based cooperative caching techniques for streaming media in wireless home networks. In *WASA*, 2009. 20

[Gur] Gurobi. Gurobi optimization. Gurobi Optimizer, http://www.gurobi.com/. 37

[GYH10] Suixiang Gao, Wenguo Yang, and Fei Huang. Replica placement algorithms in content distribution networks. In *Information Engineering and Electronic Commerce (IEEC)*, 2010. 13

[Har] Harvest. http://harvest.transarc.com/. 20

[HBM+08] Fabio Victora Hecht, Thomas Bocek, Cristian Morariu, David Hausheer, and Burkhard Stiller. LiveShift: Peer-to-Peer Live Streaming with Distributed Time-Shifting. In *Proc. of 8th Int. P2P Conf.*, pages 187–188, 2008. 11

[HCP11] Diliang He, Wei K. Chai, and George Pavlou. Leveraging in-network caching for efficient content delivery in content-centric network. In *LCS*, 2011. 21

[HL08] Huei-Chuen Huang and Rongheng Li. A k-product uncapacitated facility location problem. *European Journal of Operational Research*, 185(2):552 – 562, 2008. 17, 73

[HMPV00] Michel Habib, Ross M. McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.*, 234(1-2):59–84, 2000. 32

[HSXG09] Yifeng He, Guobin Shen, Yongqiang Xiong, and Ling Guan. Optimal prefetching scheme in p2p vod applications with guided seeks. *IEEE Transactions on Multimedia*, 11(1):138–151, 2009. xiii, 1

[IP11] Sunghwan Ihm and Vivek S. Pai. Towards understanding modern web traffic. In *IMC'11*, 2011. 68

[JJJ+00] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. On the placement of internet instrumentation. In *In Proceedings of IEEE INFOCOM 2000*, 2000. 13

[JJK+01] Sugih Jamin, Cheng Jin, Anthony R. Kurc, Danny Raz, and Yuval Shavitt. Constrained mirror placement on the internet. In *IEEE INFOCOM*, 2001. 13

[JKvS04] M. Jelasity, A.M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Proc. of ACM/IFIP/USENIX Int. Conf on Middleware*, pages 79–98, 2004. 12

[JMS02] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *ACM Symposium on Theory of Computing (STOC)*, 2002. 17

[JST+09] Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs, and Rebecca Braynard. Networking named content. In *ACM CoNEXT*, 2009. xviii, 6, 15, 67, 70, 71

[JV99] K. Jain and V. V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proc. of IEEE FOCS'99*, 1999. 17

[JVB08] Chao Jin, Christian Vecchiola, and Rajkumar Buyya. Mrpga: An extension of mapreduce for parallelizing genetic algorithms. In *IEEE eScience*, 2008. 46

[JZSRC09] Wenjie Jiang, Rui Zhang-Shen, Jennifer Rexford, and Mung Chiang. Cooperative content distribution and traffic engineering in an isp network. In *ACM SIGMETRICS*, 2009. xiii, 1

[KCC+07] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *SIGCOMM*, 2007. 15

[KK10] D. S. Knysh and V. M. Kureichik. Parallel genetic algorithms: a survey and problem state of the art. *Journal of Computer and Systems Sciences Internationl*, 49(4):579–589, 2010. 54

[KKGZ11] Dilip Kumar Krishnappa, Samamon Khemmarat, Lixin Gao, and Michael Zink. On the feasibility of prefetching and caching for online tv services: A measurement study on hulu. *Passive and Active Measurement, Lecture Notes in Computer Science*, 6579:72–80, 2011. xiii, 1

[KL87] J. G. Klincewicz and H. Luss. A dual based algorithm for multiproduct uncapacitated facility location. *Transportation Science*, 21:198–206, 1987. 17

[KLR86] J. G. Klincewicz, H. Luss, and E. Rosenberg. Optimal and heuristic algorithms for multiproduct uncapacitated facility location. *European Journal of Operational Research*, 26:251–258, 1986. 17

[Kon12] Leonidas I. Kontothanasis. Content delivery considerations for web video. `http://www.mmsys.org/?q=node/64`, 2012. xiii, 1

[KPR98] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proc. of ACM-SIAM SODA'98*, pages 1–10, 1998. 17

[KRS00] P. Krishnan, Danny Raz, and Yuval Shavitt. The Cache Location Problem. *IEEE/ACM Transactions on Networking*, 8(5):568 –582, oct. 2000. 13

[KSB+99] David Karger, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Bian Kim, Luke Mathkins, and Yoav Yerushalmi. Web caching with consistent hashing. *Computer Networks*, 31, May 1999. 20

[LAS04] Ying Lu, Tarek F. Abdelzaher, and Avneesh Saxena. Design, implementation, and evaluation of differentiated caching services. *IEEE Transactions on Parallel and Distributed Systems*, 15(5), May 2004. 72

[LCC+02] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. 16th International Conference of Supercomputing*, 2002. 17

[LCK+01] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, and Chong Sang Kim. LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies. *IEEE Trans. on Computers*, 50(12):1352–1361, 2001. 19, 94, 102

[LD10] Jimmy Lin and Chris Dyer. *Data-Intensive Text Processing with MapReduce*. Morgan & Claypool Publishers, 2010. 46

[LDD12] Aemen Lodhi, Amogh Dhamdhere, and Constantine Dovrolis. Peering strategy adoption by transit providers in the internet: a game theoretic approach? *SIGMETRICS Performance Evaluation Review*, 40(2):38–41, 2012. xiv, 2

[LDP06] Elias Leontiadis, Vassilios V. Dimakopoulos, and Evaggelia Pitoura. Creating and maintaining replicas in unstructured peer-to-peer systems. Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2006. 17

[Lei09a] Tom Leighton. Improving Performance on the Internet. *Communications of the ACM*, February 2009. 10

[Lei09b] Tom Leighton. Improving performance on the internet. *Communications of the ACM*, 52(2):44–51, 2009. 67

[Liu11] Yaning Liu. *Advanced Features for Peer-to-Peer Video Streaming Systems*. PhD thesis, Institut Mine Telecom - Telecom Bretagne, 2011. 11

[LL08] B. Li and R. Li. Approximation algorithm for k-puflpn. *Engineering and Applications (in Chinese)*, 44:97–99, 2008. 17, 18, 24

[LLJM+10] Craig Labovitz, Scott Lekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet inter-domain traffic. In *ACM SIGCOMM*, 2010. 68

[LLS11] Jimmy Leblet, Zhe Li, and Gwendal Simon. Optimal network locality in distributed virtualized data-centers. *Computer Communications*, 34(16), Oct 2011. 75

[LQK+08] Bo Li, Yang Qu, Yik Keung, Susu Xie, Chuang Lin, Jiangchuan Liu, and Xinyan Zhang. Inside the new coolstreaming: Principles, measurements and performance implications. In *INFOCOM 2008: Proc. of 27th IEEE Int. Conf. on Computer Communications.*, April. 2008. 11

[LRH10] Uichin Lee, Ivica Rimac, and Volker Hilt. Greening the internet with content-centric networking. In *ACM e-Energy Conference*, 2010. xviii, 6, 68

[LS10] Yaning Liu and Gwendal Simon. Distributed Delivery System for Time-Shifted Streaming System. In *IEEE LCN*, 2010. 70, 84, 106

[LS11] Zhe Li and Gwendal Simon. Time-Shited TV in Content Centric Networks: the Case for Cooperative In-Network Caching. In *IEEE ICC*, 2011. 48, 93

[LSHA+12] Zhe Li, Mohamed Karim Sbai, Yassine Hadjadj-Aoul, Annie Gravey, Damien Alliez, Jeremie Garnier, Gerard Madec, Gwendal Simon, and Kamal Singh. Network friendly video distribution. In *International Conference on the Network of the Future*, 2012. xiii, 1

[LSO+07] Nikolaos Laoutaris, Georgios Smaragdakis, Konstantinos Oikonomou, Ioannis Stavrakakis, and Azer Bestavros. Distributed placement of service facilities in large-scale networks. In *Proc. of IEEE INFOCOM*, 2007. 17

[LZZ10] Kaijun Liu, Yonghong Zhou, and Zigang Zhang. Capacitated location model with online demand pooling in a multi-channel supply chain. *European Journal of Operational Research*, 207(1):218–231, 2010. 18

[MCL+11] Richard T. B. Ma, Dah Ming Chiu, John C. S. Lui, Vishal Misra, and Dan Rubenstein. On cooperative settlement between content, transit, and eyeball internet service providers. *IEEE/ACM Trans. Netw.*, 19(3):802–815, 2011. xiv, 2

[MCLTC10] Han-suk Sohn Ming-Che Lai, Tzu-Liang Tseng, and Chunkuan Chiang. A hybrid algorithm for capacitated plant location problem. *Expert Systems with Applications*, 37(12):8599–8605, 2010. 18

[Mit96] Melanie Mitchell. *An introduction to genetic algorithms*. The MIT Press, 1996. 46

[MKH+09] Tatsuya Mori, Noriake Kamiyama, Shigeaki Harada, Haruhisa Hasegawa, and Ryoichi Kawahara. Improving deployability of peer-assisted cdn platform with incentive. In *Globecom*, 2009. 14

[MNR+98] Scott Michel, Khoi Nguyen, Adam Rosenstein, LIxia Zhang, Sally Floyd, and Van Jacobson. Adaptive web caching: towards a new global caching architecture. *Computer Networks and ISDN Systems*, 30, June 1998. 20

[MRKR99] C. Gerg Plaxton Madhukar R. Korupolu and Rajmohan Rajaraman. Placement algorithms for hierarchical cooperative caching. In *Proceedings of 10th ACM-SIAM SODA*, 1999. 13

[MW05] Thomas Moscibroda and Roger Wattenhofer. Maximizing the Lifetime of Dominating Sets. In *Proc. of IEEE Int Parallel and Distributed Processing Symposium (IPDPS)*, pages 242b–242b, 2005. 31

[MYZ02] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 2462, pages 229–242, 2002. 17

[NMM08] Abdelhamid Nafaa, Sean Murphy, and Liam Murphy. Analysis of a Large-Scale VOD Architecture for Broadband Operators: A P2P-Based Solution. *IEEE Communications Magazine*, December 2008. 36

[NSS10] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010. 67

[OD98] S. H. Owen and M. S. Daskin. Strategic facility location: A review. *Euro. Journal of Operational Research.*, 111:423–447, 1998. 17

[ope12] Netflix Open Connect Peering Guidelines, 2012. `https://signup.netflix.com/openconnect`. xv, 3

[Ora01] Andy Oram. Peer-to-peer: harnessing the power of distuptive technologies. *O'Reilly*, 2001. 17

[ora12] Decision relating to practices concerning reciprocal interconnection services in the area of internet connectivity. Decision 12-D-18, Sep. 2012. French Competition Authority. xv, 3

[PB07] Al-Mukaddim Khan Pathan and Rajkumar Buyya. A taxonomy and survey of content delivery networks. Technical report, Univ. of Melbourne, 2007. xiii, 1

[PCL+11] Ioannis Psaras, Richard G. Clegg, Raul Landa, Wei K. Chai, and George Pavlou. Modelling and Evaluation of CCN-caching Trees. In *IFIP Networking*, 2011. 19, 21, 93

[PJ98] Hasan Pirkul and Vaidyanathan Jayaraman. A multi-commodity, multi-plant capacitated facility location problem: formulation and efficient heuristic solution. *Computer and Operations Research*, 25(10):869–878, 1998. 18

[PP06] Sriram V. Pemmaraju and Imran A. Pirwani. Energy conservation via domatic partitions. In *Proc. of ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 143–154, 2006. 25, 31

[PPl] PPlive. `http://www.pplive.com`. 11

[PPS] PPStream. `http://www.ppstream.com`. 11

[Proa] 4WARD Project. Final architectureal framework. `http://www.sail-project.eu`. 15

[Prob] COMET Project. Content mediator architecture for content-aware networks (comet). `http://www.comet-project.org`. 15

[Proc] SAIL Project. Scalable and adaptive internet solutions (sail). `http://www.sail-project.eu`. 15

[PV06] George Pallis and Athena Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006. xiii, 1

[PV11] Diego Perino and Matteo Varvello. A reality check for content centric networking. In *ICN'11*, 2011. 68

[QGL+09] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Qi Zhao, and Jun Xu. Modeling channel popularity dynamics in a large iptv system. In *ACM SIGMETRICS/Performance*, 2009. 84

[QNM01] Lili Qiu, Venkata N.Padmanabhan, and Geoffrey M.Voelker. On the placement of web server replicas. In *INFOCOM*, 2001. 13

[QPV01] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the placement of web server replicas. In *In Proceedings of IEEE INFOCOM 2001*, 2001. 13

[Ray09] Dan Rayburn. More isps not letting cdn place servers inside their network, doing it themselves. `http://blog.streamingmedia.com/the_business_of_online_vi/2009/04/more-isps-not-letting-cdn-place-servers-inside-their-\network-doing-it-themselves.html`, April 2009. xv, 3

[Ray11] Dan Rayburn. Three new white papers released on the telco cdn space. http://bit.ly/GUDrUZ, 2011. xiii, 1

[RCG98] Micheal Rabinovich, Jeff Chase, and Syam Gadde. Not all hits are created equal: Cooperative proxy caching over a wide-area network. *Computer Networks and ISDN Systems*, 30, June 1998. 20

[RH12] Carsten Rossenhoevel and David W. Hsieh. Optimizing cloud-based video delivery, 2012. `http://www.lightreading.com/webinar.asp?webinar_id=29982`. 9

[RIF02] Kavitha Ranganathan, Adriana Iamnitchi, and Ian Foster. Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In *Proc. of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002. 18

[RKT10] Elisha J. Rosensweig, Jim Kurose, and Don Towsley. Approximate models for general cache networks. In *IEEE INFOCOM*, 2010. 19, 21, 100

[RR89] A. Srinavasa Rao and C. Pandu Rangan. Linear algorithm for domatic number problem on interval graphs. *Information Processing Letters*, 33(1):29 – 33, 1989. 31, 32

[RRL+06] S Rajasekhar, B Rong, K Y Lai, I Khalil, and Z Tari. Load sharing in peer-to-peer networks using dynamic replication. In *proc. of the 20th International Conference on Advanced Information Networking and Applications*, volume 1, 2006. 18

[SdM11] Oliver Spatscheck and Jacobus Van der Merwe. The unpubilicized sea change in the internet. *IEEE Internet Computing*, 15, Jan 2011. 68

[Smi12] Mat Smith. Youtube hits 4 billion views per day, deals with 60 hours of uploaded content every minute (update: Count it in nyans), 2012. `http://www.engadget.com/2012/01/23/youtube-hit-4-billion-views-per-day-deals-with-60-hours-of-uplo/`. 9

[SMW02] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP Topologies with Rocketfuel. In *ACM SIGCOMM*, 2002. 81, 95, 103

[SPVD09] Konstantinos Stamos, George Pallis, Athena Vakali, and Marios D. Dikaiakos. Evaluating the utility of content delivery networks. In *UPGRADE-CN*, pages 11–20, 2009. xiii, 1

[Squ] Squid. Squid-cache. `http://www.squid-cache.org/`. 20

[SRP+09] Stefano Secci, Jean-Louis Rougier, Achille Pattavina, Fioravante Patrone, and Guido Maier. ClbuMED: Coordinated Multi-Exit Discriminator Strategies for Peering Carriers. In *Next Generation Internet Networks (NGI'09)*, 2009. 10

[ST09] Karol Suchan and Ioan Todinca. Minimal interval completion through graph exploration. *Theor. Comput. Sci.*, 410(1):35–43, 2009. 32

[STA97] D. B. Shmoys, E. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proc of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997. 17

[SVS12] Abhigyan Sharma, Arun Venkataramani, and Ramesh Sitaraman. Distributing content simplifies isp traffic engineering. `http://arxiv.org/abs/1209.5715`, 2012. xiii, 1

[TSHP05] Yi-Cheng Tu, Jianzhong Sun, Mohamed Hefeeda, and Sunil Prabhakar. An analytical study of peer-to-peer media streaming system. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 1(4), 2005. 14

[Ver] Verizon. Hbo for fios customers. `http://bit.ly/JQ2dn8`. xv, 3

[VLM+09] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez. Greening the Internet with Nano Data Centers. In *Proc. of ACM CoNEXT*, 2009. 36

[VR98] Vinod Valloppillil and Keith W. Ross. Cache array routing protocol v1.0, 1998. `http://icp.ircache.net/carp.txt`. 20

[Wan99] Jia Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29, October 1999. 19

[WGMK11] Walter Wong, Marcus Giraldi, Maurício F. Magalhães, and Jaakko Kangasharju. Content routers: Fetching data on network path. In *IEEE ICC*, 2011. 93

[WL09] Jiahua Wu and Baochun Li. Keep cache replacement simple in peer-assisted vod systems. In *IEEE INFOCOM*, 2009. xiii, 1

[WL11] Weijie Wu and John C.S. Lui. Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. In *IEEE INFOCOM*, 2011. xiii, xvi, 1, 4

[WMM02] Naoki Wakamiya, Masayuki Murata, and Hideo Miyahara. On proxy-caching mechanisms for cooperative video streaming in heterogeneous environments. In *IFIP MMNS*, 2002. 21

[WVS+99] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web proxy caching. In *SOSP*, 1999. 20

[WW12] Shuming Wang and Junzo Watada. A hybrid modified pso approach to var-based facility location problems with variable capacity in fuzzy random uncertainty. *Information Sciences*, 192(1):3–18, 2012. 18

[XKSY08] Haiyong Xie, Arvind Krishnamurthy, Avi Silberschatz, and Y.Richard Yang. P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers. *P4PWG Whitepaper, May*, 2008. 19

[You12] YouTube. Youtube statistics, 2012. `http://www.youtube.com/t/press_statistics/`. 9

[YZZZ06] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. Understanding user behavior in large-scale video-on-demand systems. *SIGOPS Oper. Syst. Rev.*, 40(4):333–344, 2006. 70, 82, 94, 103

[ZCS10] Zhanguo Zhu, Feng Chu, and Linyan Sun. The capacitated plant location problem with customers and suppliers matching. *Transportation Research Part E: Logistics and Transportation Review*, 46(3):469–480, 2010. 18

Presentations and Publications

Conferences

[1] Jimmy LEBLET, Zhe LI, Gwendal SIMON, " Domatic Partition of a Distributed Service: Analysis of the Interval Completion Approach ", in *ALGOTEL'2009*, 16 - 19 June, Carry-Le-Rouet, France, 2009. (Poster)

[2] Zhe LI, Gwendal SIMON, " Time-Shifted TV in Content Centric Networks: the Case for Cooperative In-Network Caching ", in *ICC'2011: IEEE International Conference on Communications*, 05 - 09 June, Kyoto, Japan, 2011.

[3] Zhe LI, Gwendal SIMON, " Caching Policies for In-Network Caching ", in *ICCCN'2012: IEEE International Conference on Computer Communications Networks*, 30 july – 2 August, Munich, German, 2012.

[4] Zhe LI, Gwendal SIMON, Annie GRAVEY, " Caching Policies for CCN Content Store ", in *CCNxCon'2012: CCNx Community Meeting*, 12 - 13 September, Sophia Antipolis, France, 2012. (Presentation)

[5] Zhe LI, Mohamed Karim SBAÏ, Yassine HADJADJ-AOUL, Annie GRAVEY, Damien ALLIEZ, Jeremie GARNIER, Gwendal SIMON, Kamal SINGH, Gérard MADEC, " Netwrok Friendly Video Distribution ", in *NoF'2012: IEEE&IFIP Network of the Future*, 21 - 23 November, Tunisia, Tunis, 2012.

Journals

[1] Jimmy LEBLET, Zhe LI, Gwendal SIMON, Di YUAN, " Optimal Network Locality in Distributed Virtualized Data-Centers ", in *Computer communications*, October 2011, vol. 34, n.16, pp. 1968-1979.

[2] Yiping CHEN, Erwan LE MERRER, Zhe LI, Yaning LIU, Gwendal SIMON, " OAZE: a Network-Friendly Distributed Zapping System for Peer-to-Peer IPTV ", in *Computer Networks - The international journal of computer and telecommunications networking*, January 2012, vol. 56, n.1, pp. 365-377.

Working Papers

[1] Zhe LI, Gwendal SIMON, " Cooperative Caching in Content Centric Network for Video Stream Delivery ", submitted to *Springer: Journal of Network and Systems*

Management.

[2] Zhe LI, Gwendal SIMON, " In a Telco-CDN, Pushing Content Makes Sense ", submitted to *ACM Sigmetrics'2013*.

Glossary

–A–	
AS	Autonomous System
–B–	
BGP	Border Gateway Protocol
–C–	
CAGR	Compound Annual Growth Rate
CCN	Content Centric Network
CCS	Cooperative Content Store
CDN	Content Delivery Network
COMET	COntent Mediator architecture for content-aware nETworks
CPU	Central Processing Unit
CR	Content Router
CRF	Combined Recency and Frequency
CRT	Cooperative Router Table
CS	Content Store
–D–	
DASH	Dynamic Adaptive Streaming over HTTP
dCDN	distributed Content Delivery Network
DHT	Distributed Hash Table
DNS	Domain Name System
DONA	Data-Oriented Network Architecture
–E–	
Ebone	European backbone
–F–	
FIB	Forwarding Information Base
FLP	Facility Location Problem
–G–	
GA	Genetic Algorithm

–H–	
HD	High Definition
HDFS	HaDoop File System
HTTP	Hypertext Transfer Protocol
–I–	
ICN	Information Centric Network
ICP	Internet Caching Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IRM	Independent Reference Model
IS-IS	Intermediate System To Intermediate System
ISP	Internet Service Provider
ITP	Internet Transit Provider
–K–	
k-CMSP	k-Component Multi-Site Placement
κ-MP	κ-Median Problem
k-PCFLP	k-Product Capacitated Facility Location Problem
k-PUFLP	k-Product Uncapacitated Facility Location Problem
–L–	
Lex-BFS	Lexicographic Breadth First Search
LFU	Least Frequently Used
LRU	Least Recently Used
LRFU	Least Recently/Frequently Used
LSA	Link State Advertisement
–M–	
MCMF	Minimum Cost Maximum Flow
MR	MapReduce
–N–	
NAT	Network Address Translation
–O–	
OSPF	Open Shortest Path First
–P–	
PGA	Parallel Genetic Algorithm
PIT	Pending Interest Table
PoP	Point of Presence
P2P	Peer-to-Peer
–Q–	
QoS	Quality of Service

–R–

RAM	Random-Access Memory
RTP	Real-time Transport Protocol
RTT	Round-Trip Time

–S–

SAIL	Scalable and Adaptive Internet Solutions
SDN	Software Defined Networking
SSH	Secure Shell

–T–

TCP	Transmission Control Protocol
TRIAD	Translating Relaying Internet Architecture integrating Active Directories
TV	Television

–U–

UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

–V–

VCR	Videocassette Recorder
VoD	Video on Demand