UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

Ecole Doctorale N° 601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Hamza BEN AMMAR

On Models for Performance Evaluation and Cache Resources Placement in Multi-Cache Networks

Thèse présentée et soutenue le 19 Mars 2019
Unité de recherche : IRISA UMR 6074

Rapporteurs avant soutenance :

BARAKAT Chadi	Directeur de Recherche, INRIA Sophia Antipolis
BEYLOT André-Luc	Professeur, ENSEEIHT

Composition du jury :

Président :	BEYLOT André-Luc	Professeur, ENSEEIHT
Examinateurs :	AIT-CHELLOUCHE Soraya	Maître de Conférences, Université de Rennes 1
	BARAKAT Chadi	Directeur de Recherche, INRIA Sophia Antipolis
	FRICKER Christine	Chargée de Recherche, INRIA, Paris
Dir. de thèse :	HADJADJ-AOUL Yassine	Maître de Conférences, Université de Rennes 1
Co-dir. de thèse :	RUBINO Gerardo	Directeur de Recherche, INRIA, Rennes

REMERCIEMENTS

Je tiens tout d'abord à remercier mes directeurs de thèse et mes encadrants Yassine HADJADJ AOUL, Gerardo RUBINO et Soraya AIT CHELLOUCHE, de m'avoir donné la chance d'effectuer cette thèse. Leur suivi et leurs conseils avisés m'ont apporté beaucoup d'aide et m'ont permis d'aboutir aux travaux présentés dans ce manuscrit.

Je veux également exprimer toute ma gratitude envers les membres du jury de soutenance. Je remercie Chadi BARAKAT et André-Luc BEYLOT d'avoir accepté d'être rapporteurs de cette thèse et de m'avoir fourni des conseils et des retours très intéressants sur mes travaux. Je remercie aussi Christine FRICKER pour l'intérêt qu'elle a bien voulu porter à mon travail en acceptant de prendre part à ce jury et pour ses remarques enrichissantes.

Merci aussi à tous les membres des équipes CAIRN et GRANIT pour les moments agréables qu'on a passé durant ces dernières années que ce soit au labo ou en dehors du cadre du travail comme par exemple pendant nos réunions de CST/CSID/CSI, à ne pas confondre avec le Comité de Suivi de Thèse, devenu Comité de Suivi Individuel du Doctorant et qui se nomme maintenant Comité de Suivi Individuel (Really ?). Leur bonne humeur et leurs blagues pas racistes (officiellement) m'ont permis de survivre aux BdP/BdR/BdL subis durant toute la thèse (seul les vrais savent !). En particulier, je remercie Cédric et Antoine que je considère comme mes encadrants de thèse (TMTC) pour leur soutien et leurs conseils qui ont permis d'enrichir mes travaux de recherche et merci aussi à Gabriel et Christophe de m'avoir soutenu tout le long de ma thèse, surtout durant les moments les plus critiques (BWAAAAAAAH !).

Pour finir, un grand merci à mon oncle Charif, à toute ma famille et surtout à mes parents Hamadi et Rafika qui m'ont soutenu tout au long de mes études et qui m'ont aidé à aller au bout de mes envies.

CONTENTS

List of figures **11**

List of tables **13**

Acronyms **15**

Résumé **19**

Abstract **21**

0 Résumé étendu **23**
1 L'évolution d'Internet et les motivations d'Information-Centric Networking 24
2 Content-Centric Networking 25
2.1 Modèle d'un nœud CCN 27
2.2 Le routage des paquets d'intérêts/de données dans les CCN . . 28
3 Contributions 29

Introduction **33**
1 Context of the work 33
2 Motivations and contributions 35
3 Thesis outline 37

1 Background **39**
1 Information-Centric Networking 40
1.1 Motivations 40
1.2 ICN description 40
1.3 ICN proposals 42
1.3.1 Publish Subscribe Internet Routing Paradigm 42
1.3.2 Network of Information 43
1.3.3 Data Oriented Networking Architecture 45
1.3.4 Content-Centric Network 46

2 Content-Centric Networking 46
2.1 CCN naming and message exchange 46
2.2 CCN node model 48
2.3 The Interest/Data packets routing in CCN 49
2.4 Discussion 51
3 Cache modeling in multi-cache networks 51
3.1 Related work 51
3.2 Che approximation 53
3.3 Discussion 55
4 Cache allocation in multi-cache networks 55
4.1 Related work 55
4.2 Discussion 57
5 Conclusion 58

2 Modeling multi-cache networks using Markov chains 59
1 Motivations 60
2 System description 61
3 A single LRU cache model under LCE 62
3.1 A comprehensive analysis of LRU caches 62
3.2 Markov chain-based approximation of an LRU cache 65
4 A general Markov chain model of a single LRU cache 68
4.1 A comprehensive analysis of generic LRU caches 68
4.2 A generic model of a single LRU cache 68
4.3 Single cache model for 2Q 72
4.4 Single cache model for LCD 74
5 Multiple caches systems 75
6 Model Evaluation 77
6.1 Tests environment 77
6.2 Model results and analysis 79
6.3 Caching algorithms comparison 85
7 Conclusion 91

3 Efficiently allocating distributed caching resources in future networks 93
1 Motivations 94
2 Multi-objective cache placement strategy 95

2.1 System assumptions 95
2.2 Problem formulation 96
2.3 Solving cache allocation problem using GRASP 98
2.3.1 Mono-objective GRASP 98
2.3.2 Multi-objective GRASP 101
3 Performance evaluation 104
3.1 Model configuration 104
3.2 Model results and analysis 105
4 Conclusion 111

Conclusion **113**
1 Overview 113
2 Perspectives 115
2.1 Dynamic content popularity 115
2.2 Caching from an economic point of view 116
2.3 Caching in wireless networks 116

Publications **119**

Bibliography **121**

LIST OF FIGURES

1 Le nommage du contenu en CCN. 25
2 Modèle d'un nœud CCN. 26
3 Traitement des paquets dans CCN. 29
4 Cisco Visual Networking Index: Global IP traffic evolution and forecast, 2015–2022. 33
5 Network architecture with the in-network caching capability. 34

1.1 ICN networking Overview. 41
1.2 Overview of PSIRP [22]. 42
1.3 Routing scheme of NetInf [23]. 43
1.4 Routing scheme of DONA [24]. 44
1.5 Overview of CCN [12]. 45
1.6 CCN packet types [12]. 46
1.7 The naming of contents in CCN [12]. 47
1.8 The node model in CCN [12]. 48
1.9 Packets processing in CCN [25]. 50

2.1 A Markov chain model of an LRU cache where the LCE strategy is adopted with $R = 3$ and $N = 2$. 64
2.2 Markov chain model for a content c_r in an LRU cache. 66
2.3 General Markov chain model for a content c_r in an LRU cache. 69
2.4 Cache hit ratio of contents with various popularities vs iterations number of the 2Q fixed-point solution under a single node (catalog = 20000, cache size = 200). 73
2.5 Cache hit ratio of contents with various popularities vs iterations number of the LCD fixed-point solution under a single node (catalog = 20000, cache size = 200). 74
2.6 Comparison of MACS and Che approximation in the case of a single cache. 79

2.7 Average hit rate vs content popularity with different parameters and using the LCE scheme. 81
2.8 Average hit rate vs content popularity with different parameters and using the 2Q scheme. 81
2.9 Average hit rate vs content popularity with different parameters and using the LCD scheme. 82
2.10 Total hit rate of the network vs different configurations using the LCE scheme. 82
2.11 Total hit rate of the network vs different configurations using the 2Q scheme. 83
2.12 Total hit rate of the network vs different configurations using the LCD scheme. 84
2.13 Cache hit ratio at different layers of the network using the LCE scheme. 84
2.14 Cache hit ratio at different layers of the network using the 2Q scheme. . 85
2.15 Cache hit ratio at different layers of the network using the LCD scheme. 86
2.16 Total hit rate of the network with different parameters and various caching strategies. 87
2.17 Total hit rate at different layers of the network with different parameters and various caching strategies. 88
2.18 Cache hit ratio vs content popularity at different layers of the network with different parameters and various caching strategies. 89
2.19 Average distance reduction ratio of the network with different parameters and various caching strategies. 90
2.20 Content provider load of the network with different parameters and various caching strategies. 91

3.1 An example of a network architecture for content delivery with cache-enabled nodes. 104
3.2 Comparison between the Pareto front and GRASP solutions under the LCE scheme. 105
3.3 Comparison between the Pareto front and GRASP solutions under the 2Q scheme. 106
3.4 Performance optimization vs total cache size using GRASP compared to the optimal values under the LCE scheme. 107

3.5 Performance optimization vs total cache size using GRASP compared to the optimal values under the 2Q scheme. 107

LIST OF TABLES

1 Un exemple d'une table PIT. 27
2 Un exemple d'une table FIB. 28

1.1 An example of a CCN PIT. 49
1.2 An example of a CCN FIB. 49

2.1 Summary of the notations. 61
2.2 Hit ratio accuracy comparison (in %) between different models of a 2Q single cache under the IRM model with a Zipf's distribution parameter $\alpha = 0.8$. Simulations are based on 10 runs of $10^3 \times R$ requests with a warm-up period of 33 %. 80

3.1 Cache allocation solutions using GRASP with two separate evaluation functions when LCE is used. 108
3.2 Cache allocation solutions using GRASP with two separate evaluation functions when 2Q is used. 109
3.3 Cache allocation solutions using GRASP with weighted combined evaluation function when LCE is used. 109
3.4 Cache allocation solutions using GRASP with weighted combined evaluation function when 2Q is used. 110

ACRONYMS

ICN	Information-Centric Networking
PSIRP	Publish Subscribe Internet Routing Paradigm
NetInf	Network of Information
DONA	Data Oriented Networking Architecture
CCN	Content-Centric Network
NDO	Named Data Object
RN	RendezVous Node
TN	Topology Node
BN	Branching Node
FN	Forwarding Node
NRS	Name Resolution Service
NBR	Name Based Routing
RH	Resolution Handler
CS	Content Store
PIT	Pending Interest Table

FIB	Forwarding Information Base
FIFO	First In First Out
LRU	Least Recently Used
TTL	Time-To-Live
LCE	Leave Copy Everywhere
LCD	Leave Copy Down
CDN	Content Delivery Network
NFV	Network Function Virtualization
QoS	Quality of Service
ILP	Integer Linear Programming
CDNaaS	CDN as a Service
VNF	Virtual Network Function
QoE	Quality of Experience
MACS	Markov chain-based Approximation of Caching Systems
ISP	Internet Service Provider
CP	Content Provider
ARPU	Average Revenues Per User

IRM	Independent Reference Model
RR	Random Replacement
LFU	Least Frequently Used
2Q	Two Queue
UGC	User Generated Content
VoD	Video on Demand
GRASP	Greedy Randomized Adaptive Search Procedure
MEC	Multi-access Edge Computing
PoP	Point of Presence
RCL	Restricted Candidate List

RÉSUMÉ

Au cours des dernières années, les fournisseurs de contenu ont connu une forte augmentation des demandes de contenus vidéo et de services riches en média. Compte tenu des limites de la mise à l'échelle du réseau et au-delà des réseaux de diffusion de contenu, les fournisseurs de services Internet développent leurs propres systèmes de mise en cache pour améliorer la performance du réseau. En effet, la diffusion de caches dans l'infrastructure permet non seulement d'absorber la congestion du réseau, mais aussi de rapprocher les contenus aux utilisateurs, ce qui contribue à améliorer leur qualité d'expérience. Ces facteurs expliquent l'enthousiasme de l'industrie et des académiques à l'égard du concept de réseau centré sur le contenu (Content-Centric Network) et de sa fonction de mise en cache en réseau. La quantification analytique de la performance de la mise en cache n'est toutefois pas suffisamment explorée dans la littérature. De plus, la mise en place d'un système de *caching* efficace au sein d'une infrastructure réseau est très complexe et demeure une problématique ouverte.

Pour traiter ces questions, nous présentons d'abord dans cette thèse un modèle générique de nœuds de *caching* nommé MACS (Markov chain-based Approximation of Caching Systems) qui peut être adapté très facilement pour représenter différents schémas de mise en cache et qui peut être utilisé pour calculer différentes mesures de performance des réseaux multi-cache, y compris le *cache hit* moyen. Les essais effectués ont démontré la précision de notre modèle et plusieurs conclusions ont été tirées sur l'efficacité et les limites de la mise en cache. Nous avons ensuite abordé le problème de l'allocation des ressources de cache dans les réseaux avec capacité de *caching*. Moyennant notre outil analytique MACS, nous avons proposé une approche permettant de résoudre le compromis entre différentes mesures de performance en utilisant l'optimisation multi-objectif. Ensuite, une adaptation de la métaheuristic GRASP (Greedy Randomized Adaptive Search Procedure) pour résoudre le problème a été présentée et nous avons pu en savoir plus sur le placement optimal des ressources de cache distribuées.

Mots clés : Réseau centré sur le contenu, Mise en cache, Chaînes de Markov, Allocation de caches, Optimization multi-objectif, GRASP.

ABSTRACT

In the last few years, Content Providers (CPs) have experienced a high increase in number of requests for video contents and rich media services. In view of the network scaling limitations and beyond Content Delivery Networks (CDNs), Internet Service Providers (ISPs) are developing their own caching systems to improve the network performance. Indeed, disseminating caches in the infrastructure not only helps in absorbing the network's congestion, but in addition, brings content closer to users, which helps improving their Quality of Experience (QoE). These factors explain the enthusiasm of industry and academics around the Content-Centric Networking (CCN) concept and its in-network caching feature. The analytical quantification of caching performance is, however, not sufficiently explored in the literature. Moreover, setting up an efficient caching system within a network infrastructure is very complex and remains an open problem.

To address these issues, we provide first in this thesis a fairly generic model of caching nodes named MACS (Markov chain-based Approximation of Caching Systems) that can be adapted very easily to represent different caching schemes and which can be used to compute different performance metrics of multi-cache networks, including the average cache hit. The conducted tests have shown the accuracy of our model and several conclusions were drawn on the efficiency and limits of caching. We tackled after that the problem of cache resources allocation in cache-enabled networks. By means of our analytical tool MACS, an approach solving the trade-off between different performance metrics was proposed using multi-objective optimization. Then, an adaptation of the metaheuristic GRASP (Greedy Randomized Adaptive Search Procedure) to solve the optimization problem was presented, and we were able to gain more insights on the optimal placement of distributed caching resources.

Keywords: Content-Centric Networking, Caching, Markov chains, Cache allocation, Multi-objective optimization, GRASP.

CHAPTER 0
RÉSUMÉ ÉTENDU

Contents

1 L'évolution d'Internet et les motivations d'Information-Centric Networking 24

2 Content-Centric Networking 25

2.1 Modèle d'un nœud CCN 27

2.2 Le routage des paquets d'intérêts/de données dans les CCN . 28

3 Contributions 29

1 L'évolution d'Internet et les motivations d'Information-Centric Networking

L'usage de l'Internet a radicalement changé depuis ses débuts. Conçu pour établir des communications point-à-point entre hôtes, l'usage qu'on en fait aujourd'hui, est de plus en plus centré sur l'accès à des contenus et services, indépendamment des hôtes qui les fournissent. Ce décalage entre le modèle conversationnel sur lequel il est conçu et l'usage qui en est fait, résulte en de nombreuses limites et contraintes, en termes de capacité, de performance et de fiabilité.

Des solutions, sous forme de patchs, ont été proposées et déployées de manière incrémentale, afin de pallier à ces limites. Comme exemple de ces solutions, on peut citer les réseaux de diffusion de contenu (Content Delivery Networks ou CDNs) et les réseaux pair-à-pair déployés pour répondre aux problèmes liés au passage à l'échelle de la distribution des contenus. Cependant, ces solutions ne prennent souvent pas en considération les réseaux de transport sous-jacents.

Les réseaux centrés sur l'information (Information-Centric Networks ou ICNs) représentent une alternative à l'architecture actuelle de l'Internet. Ces réseaux ont été proposés pour répondre à plusieurs problématiques liées aux différentes tendances actuelles dans l'usage qui en est fait. Le réseau centré sur les contenus (Content Centric Network ou CCN), proposé par le Palo Alto Research Center, représente une des architectures ICN qui a suscité un vif intérêt de la part de la communauté de recherche. Basée, d'une part, sur le déploiement de caches au sein même du réseau (on parle de "in-network caching") et d'autre part, sur la séparation du nommage de la localisation, cette architecture permet de fournir, de par sa nature, un meilleur support pour la mobilité, le multicast, la gestion des ressource, la sécurité, etc.

L'intégration des caches aux équipements réseau est une fonctionnalité inhérente aux réseaux centrés sur les contenus. En effet, ce paradigme consiste à placer des caches dans un plus grand nombre d'équipements réseau tels que des routeurs ou des boxes. Ces équipements sont certes plus nombreux et plus proches des utilisateurs, mais ils sont également plus limités en ressources, comparés aux caches utilisés dans les architectures plus traditionnelles telles que les CDNs. De nouvelles stratégies de gestion de caches prenant en compte ce nouveau contexte de déploiement sont donc nécessaires afin d'arriver à un compromis entre une meilleure disponibilité des contenus et une utilisation optimale des ressources réseau.

2 Content-Centric Networking

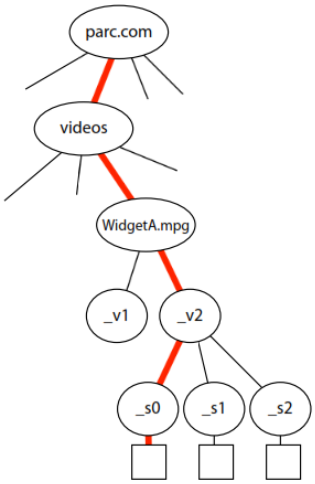


FIGURE 1 – Le nommage du contenu en CCN.

Content-Centric Networking a été proposé en 2009 par le Palo Alto Research Center (PARC). CCN propose un modèle complet de mise en réseau basé sur le contenu (nommage du contenu, routage basé sur le contenu, etc.) où un contenu est accédé par son nom au lieu de l'adresse IP de l'hôte qui le possède. Pour assurer la communication entre les différents nœuds d'un réseau CCN, deux types de paquets sont utilisés :

— Paquet d'intérêt ou "Interest" : utilisé pour transporter les demandes de l'utilisateur. Il porte les éléments suivants :
  ► Nom du contenu.
  ► Sélecteurs : ce sont des éléments optionnels utilisés pour découvrir et sélectionner les données qui correspondent aux besoins de l'application.
  ► Nonce : il s'agit d'un nombre aléatoire utilisé par un routeur pour détecter si le paquet d'intérêt est dupliqué ou non.

— Paquet de données : il contient les champs suivants :
  ► Nom du contenu.
  ► Signature et infos signées : selon les informations signées, les utilisateurs peuvent obtenir la clé publique de l'éditeur de contenu pour vérifier les données reçues avec la règle spécifiée dans le champ signature et décider d'accepter ou non les données.

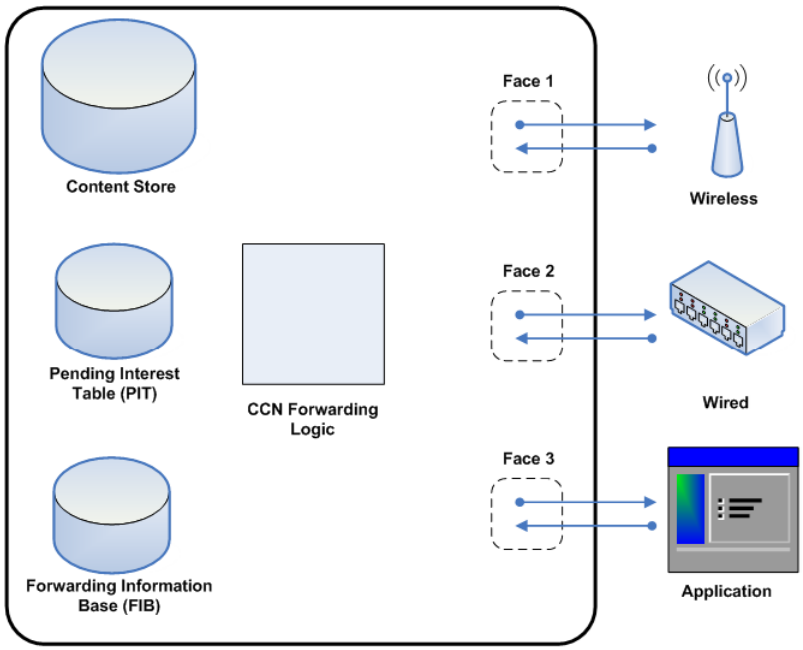


FIGURE 2 – Modèle d'un nœud CCN.

- Données : ce sont les données qui correspondent au nom du contenu dans le paquet d'intérêt reçu.

Un utilisateur demande un contenu spécifique en diffusant son intérêt dans le réseau. Tout nœud qui reçoit l'intérêt et a les données sollicitées peut répondre avec un paquet de données et dans ce cas, l'intérêt sera consommé. Comme nous l'avons déjà mentionné, le nom du contenu dans CCN est le seul identificateur des données et les échanges d'informations qui reposent sur l'établissement de canaux de communication sont abandonnés. Les noms sont structurés hiérarchiquement comme le nom de chemin d'un système de fichiers (voir Figure 1).

La structure arborescente permet d'agréger les contenus et donc de faciliter la découverte du contenu. Par exemple, comme le montre la Figure 1, tout le contenu fourni par "parc" peut partager le même préfixe "parc.com/". Un même objet de contenu dans CCN peut avoir plusieurs versions et il peut être divisé en plusieurs fragments appelés "chunks" (ou morceaux) afin d'ajuster la couche de transport. Pour simplifier la recherche et le transfert de contenu, le nom se termine généralement par la version et les informations sur les morceaux.

TABLE 1 – Un exemple d'une table PIT.

Noms des contenus	Faces entrantes
ccn :/spotify.com/music1.mp3	301
ccn :/irisa.com/documents/doc1.docx	201,203
ccn :/youtube.com/videos/video1.mp4	102,105
...	...

2.1 Modèle d'un nœud CCN

Dans un réseau CCN, chaque nœud est essentiellement composé de trois structures de données (voir Figure 2) :

— Le Content Store (CS) : le CS est une mémoire tampon qui fournit la fonction de mise en cache dans le réseau en sauvegardant des copies des données qui circulent afin d'optimiser les performances du réseau (minimiser la latence, réduire la surcharge du réseau, etc.).

— La table PIT (Pending Interest Table) : elle contient deux types d'informations (voir Table 1) : les noms des contenus dans les messages d'intérêt reçus et les *faces* entrantes associées (les *faces* jouent un rôle similaire aux interfaces dans le réseau IP). Une table PIT a deux rôles. Le premier consiste à garder la trace des intérêts transférés en attente d'un paquet de données retourné, en enregistrant le nom du contenu demandé et la face associée. Le deuxième rôle est d'éviter de transmettre plusieurs intérêts qui demandent le même contenu ; lorsqu'un nœud CCN reçoit de la part de différentes interfaces plusieurs messages d'intérêt qui demandent le même contenu, ces paquets d'intérêt seront regroupés en une seule entrée dans le PIT et seul le premier sera acheminé.

— La table FIB (Forwarding Information Base) : la FIB est utilisée pour guider les paquets "Interest" vers les sources potentielles des données demandées en sauvegardant dans un tableau deux types d'informations (voir Table 2) : le nom du contenu (ou les préfixes agrégés des noms) et les faces sortantes associées. Un contenu ayant plusieurs interfaces sortantes signifie qu'une donnée peut avoir plusieurs sources.

TABLE 2 – Un exemple d'une table FIB.

Préfixes	Faces sortantes
ccn :/spotify.com/	101
ccn :/irisa.com/documents/	204,206
ccn :/youtube.com/videos/	301,302
...	...

2.2 Le routage des paquets d'intérêts/de données dans les CCN

Dans les réseaux CCN, seuls les paquets d'intérêts sont acheminés et lorsqu'ils remontent vers des sources de données potentielles, ils laissent des traces dans les nœuds par lesquelles ils sont passés afin que les paquets de données puissent suivre pour atteindre leurs destinations.

Le traitement et le transfert de paquets dans les nœuds CCN s'effectuent comme suit :

1. Tout d'abord, un nœud CCN reçoit un paquet d'intérêt où le nom du contenu porté par ce paquet est vérifié dans le *Content Store*.
2. S'il y a un paquet de données dans le CS qui correspond à la demande, il est alors renvoyé par l'interface entrante de l'intérêt traité. Sinon, le nom du contenu est vérifié dans le PIT.
3. S'il y a une entrée correspondante dans le PIT, alors la requête est rejetée et sa face entrante est ajoutée à l'entrée PIT existante (agrégation des paquets Intérêt). Si ce n'est pas le cas, une nouvelle entrée est créée et les intérêts sont transmis sur la base des informations de la table FIB.
4. Lorsque le contenu de la requête est trouvé, le paquet de données suivra les traces laissées par les paquets d'intérêt afin d'atteindre la destination.
5. Une fois qu'un noeud reçoit un paquet de données, son nom de contenu est vérifié dans le CS.
6. S'il y a une correspondance, le paquet de données doit être écarté puisqu'il existe déjà dans le CS. Sinon, le nom du contenu est recherché dans le PIT.

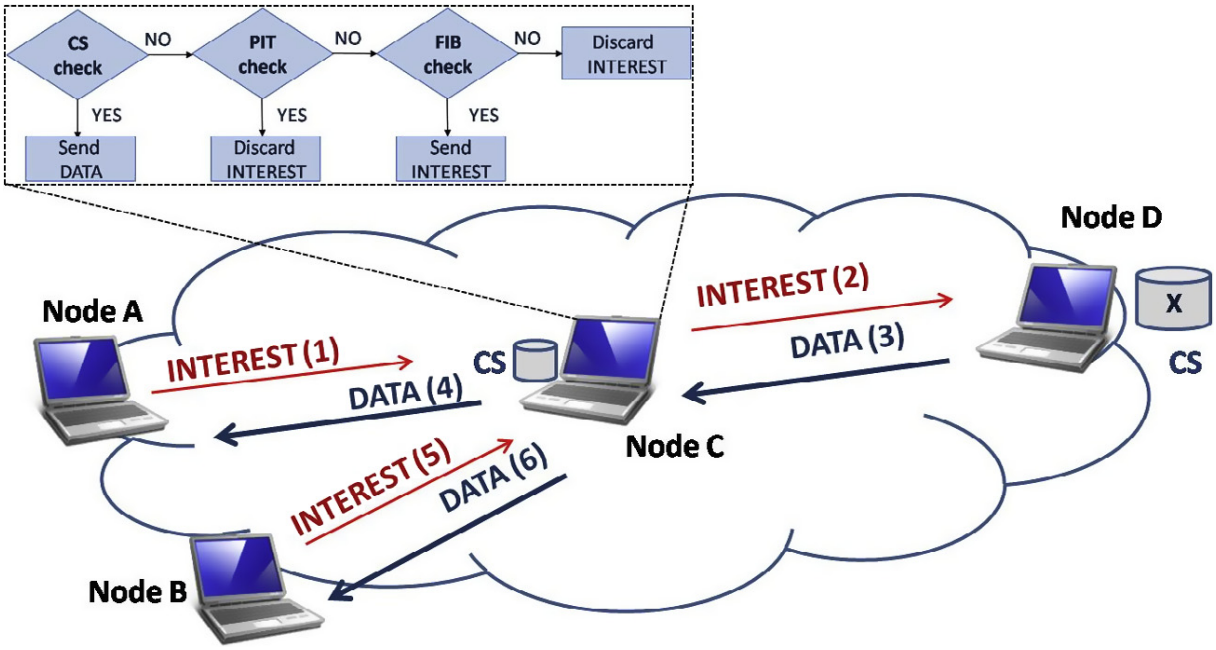


FIGURE 3 – Traitement des paquets dans CCN.

7. Si une entrée correspondante est trouvée dans le PIT, le paquet de données est envoyé par les faces où l'intérêt a été reçu. L'entrée PIT correspondante est alors supprimée et en fonction de la stratégie de mise en cache, le nœud peut conserver une copie du contenu dans le CS.

Un exemple de traitement et de transfert de paquets dans un CCN est présenté dans la Figure 3. Une fois que le paquet d'intérêt est reçu du nœud A (qui demande l'objet x), le nœud C qui ne trouve pas de correspondance dans son cache ni dans la table des intérêts, transmet le paquet au nœud source D. Lorsque le nœud C reçoit le paquet de données de la part de D, il le renvoie au client attaché à A et il peut ensuite avec sa copie en cache servir la demande provenant du nœud B pour le même contenu x.

3 Contributions

Comme mentionné précédemment, la fonctionnalité de *caching* est l'une des caractéristiques les plus importantes de CCN. Elle joue un rôle majeur dans l'amélioration des performances du réseau. C'est pourquoi il est crucial d'utiliser efficacement les ressources de caches afin d'avoir un meilleur accès aux contenus. L'optimisation de la

fonctionnalité de caching peut être traitée à différents niveaux :

— Algorithme de remplacement de cache : c'est l'algorithme responsable de la gestion du cache. Lorsque le cache est plein, un élément est choisi par l'algorithme pour être rejeté. Nous pouvons citer les algorithmes de mise en cache First In First Out (FIFO), Least Recently Used (LRU), etc.
— Politique de décision de caching : elle détermine quel contenu doit être mis en cache et dans quel nœud. En d'autres termes, un nœud doit décider quand il reçoit un contenu et selon la stratégie appliquée, s'il doit mettre les données en cache ou pas.
— Placement des ressources de caches : la capacité du cache pouvant être allouée à un nœud CCN étant limitée, le placement de cache consiste à étudier la répartition optimale des ressources de stockage entre les nœuds du réseau (ex : adopter une répartition équitable, donner plus d'espace de stockage aux routeurs du *edge*, etc).

Les travaux de cette thèse se sont concentrés sur les problèmes liés à la modélisation du cache et à l'allocation des ressources du cache dans les réseaux CCN et les réseaux multi-cache en général. Les contributions réalisées et validées peuvent être résumées comme suit :

— Un modèle analytique basé sur les chaînes de Markov nommé MACS (Markov chain-based Approximation of Caching Systems) a été proposé. Il permet de modéliser la gestion des caches dans les réseaux CCN et d'estimer ainsi les performances du système. Ce travail couvre la technique de mise en cache LCE (Leave Copy Everywhere) et l'algorithme de remplacement de données dans le cache LRU.
— Ensuite, une extension du modèle MACS a été développée. Elle nous permet de modéliser le mécanisme de caching dans le cas général afin de pouvoir analyser plusieurs stratégies de mise en cache intéressantes. Le modèle a été validé théoriquement et par simulation. Grâce à cet outil, nous avons pu modéliser et analyser deux techniques de caching très intéressantes nommées 2Q (Two Queue) et LCD (Leave Copy Down) connues pour avoir un bon rapport efficacité/complexité. Les différents résultats générés ont montré la précision du modèle proposé sous différentes configurations du réseau et ils nous ont permis d'en savoir plus sur les stratégies de caching qu'il faudra utiliser pour avoir les meilleures performances.

— Après avoir étudié comment placer et utiliser de façon efficace les ressources limitées de mise en cache de données dans un réseau distribué, nous avons proposé une modélisation du problème sous forme d'optimisation combinatoire, où notre outil MACS est utilisé pour évaluer les fonctions objectifs. Une application de la métaheuristique GRASP (Greedy randomized adaptive search procedure) sur notre modèle d'optimisation a été ensuite proposée et une analyse a été faite sur les différents résultats que nous avons obtenus. Cela nous a permis de mieux comprendre l'impact du placement des ressources du cache sur les performances du réseau et de voir comment notre solution peut s'adapter aux différentes contraintes et objectifs établies dans un système de multi-cache.

INTRODUCTION

1 Context of the work

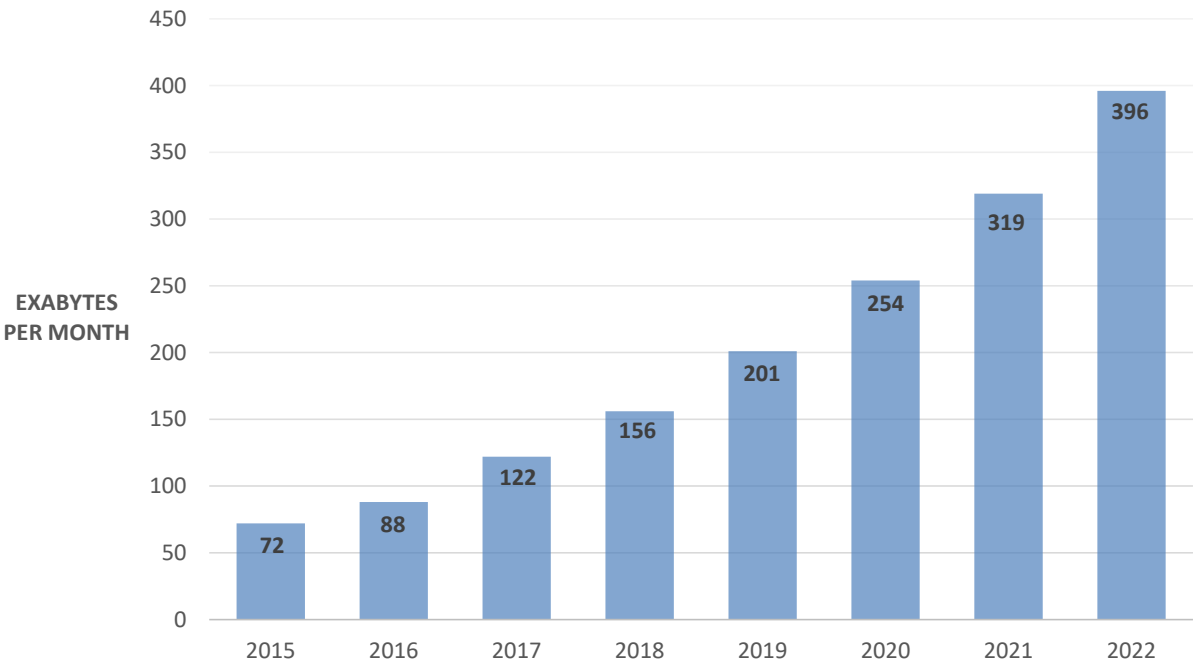


Figure 4 – Cisco Visual Networking Index: Global IP traffic evolution and forecast, 2015–2022.

The last few years witnessed a shift on the Internet usage that switched from a host-centric model to a content-centric approach, especially when dealing with content retrieval and data dissemination [1]. This evolution is mostly driven by the rapid growth of media-enriched services, e.g., Peer-to Peer file sharing, Video on Demand, video/audio streaming and social networks, which has significantly changed the way that people experience the Internet, making media traffic, and especially video traffic, one of the main drivers of the Internet Economy. According to the Cisco Visual Networking Index [2]-[3], the global IP traffic will reach 396 exabytes (EB) per month in 2022, up from 72 EB per month in 2015 (see Figure 4). Besides, video services and

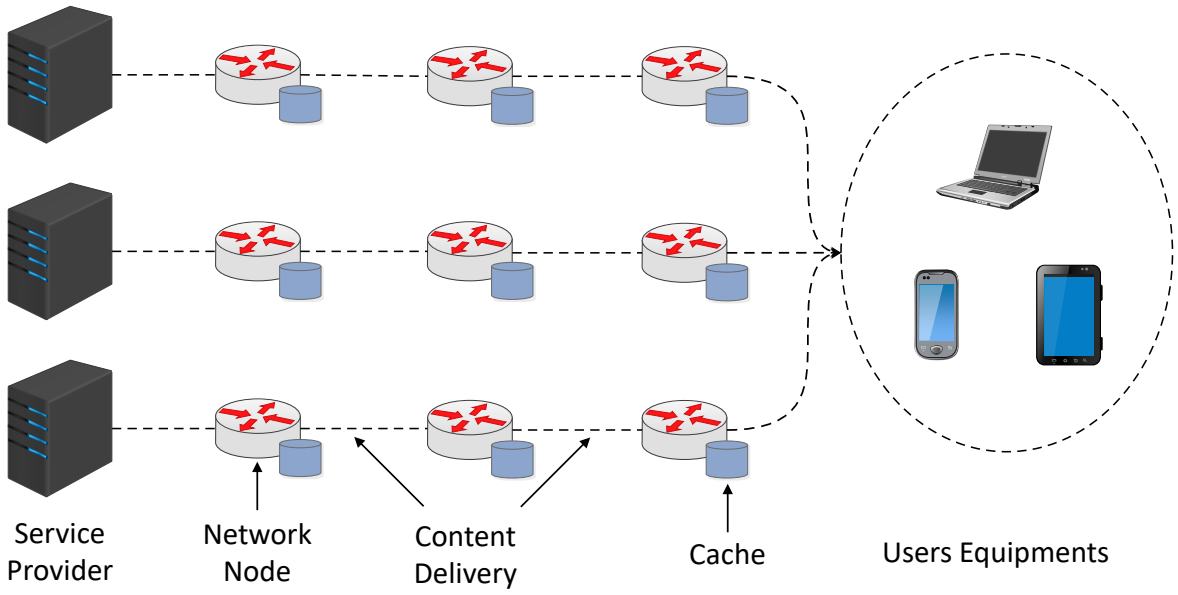


Figure 5 – Network architecture with the in-network caching capability.

content continue to be the dominant leader compared with all other applications. Internet video will account for about 80% of global Internet traffic by 2020, up from 63 percent in 2015. Across the globe, the number of devices connected to IP networks will be more than three times the global population by 2022. There will be 3.6 networked devices per capita by 2022, up from 2.4 networked devices per capita in 2017 and a total of 28.5 billion networked devices by 2022, up from 18 billion in 2017 [3].

To deal with these new trends of the Internet usage and the important challenges that have raised with it [4], it is obvious that current IP-based infrastructures should evolve in a way to optimize the content delivery. Such data consumption context allowed CDNs (Content Delivery Networks) [5] to be at the center of the content distribution value chain. Internet Service Providers (ISPs), for their part, are struggling to benefit from this traffic increase [6]. On the contrary, this trend incurs large expenditures to meet the increasing demand and satisfy subscribers. Besides, given the adopted flat rate business models and even though that ISPs are payed by CDNs for hosting their data centers, the Average Revenues Per User (ARPU) is getting lower. ISPs are, thus, investigating the possibilities to extend their infrastructure with caching capabilities [7] as a way to be part of this content distribution value chain (see Figure 5).

The term *cache* [8] was first employed in computing systems to describe a data storing technique that provides the ability to access data or files at a higher speed.

Caches can be implemented both in hardware and software and they are used to serve as an intermediary component between the primary storage appliance and the recipient hardware or software device to reduce the latency in data access. The concept of caching was later adopted by the Internet [9]. By storing popular contents from previous requests, web caching solutions can help enhancing the content access time significantly. The rapid growth of Internet traffic has led to the emerging of Content Delivery Networks, that have become an important layer in the Internet ecosystem. CDNs are used to enhance the delivery of content by replicating commonly requested files across a globally distributed set of caching servers. The aim is to reduce the load on origin servers and to improve the experience of the clients by delivering a local copy of the content from a nearby cache edge.

More recently, there has been a growing emphasis on the Information-Centric Networking paradigm (ICN) [10]-[11]-[4], which is emerging as one of the most interesting alternatives to the existing network architecture. The ICN paradigm consists in redesigning the future Internet architecture by focusing on named data instead of the end-to-end principle and the host-centric paradigm. One major feature considered in ICN to achieve service scalability and performance, is the in-network caching, where each node equipped with a content store module has the ability to cache the content that passes by it. The advent of ICNs represents, thus, a real opportunity for ISPs [12]. In fact, ICN networks enable focusing on the content itself and not on its location, which allows to overcome the limitations of the current Internet network. Thus, end-users' requests that are routed toward the Content Providers' servers (CPs), can be satisfied by intermediate caching nodes, which allows reducing the network traffic and the servers' loads while shortening the latency for end-users.

2 Motivations and contributions

When content caching is possible, a significant improvement can be achieved, as shown in several studies [13]-[14]. The analytical quantification of caching performance is, however, not sufficiently explored in the literature. In fact, several issues need to be addressed in order to understand the behavior of such a caching network. Indeed, in addition to the cache hit probability, other metrics deserve a deeper analysis, such as the average distance to reach a particular content, the content provider load or the impact of cache size and traffic pattern on performance.

Moreover, setting up an optimal caching system within a network infrastructure, up to the edge of the latter at the Multi-access Edge Computing (MEC) level [15], is very complex and remains an open issue in the literature [16]. In fact, if we want to optimize the use of caches, we will tend to put everything in the same place at the exit of the network (at Point of Presence level), where users requests will be aggregated. This allows, indeed, to maximize the hit rate, but has the drawback of overloading the network. However, if one wants to maximize user satisfaction while at the same time reducing network load, the ideal would be to put everything on the edge. The problem here is that there will be more redundancy at the edge because of the multitude of access networks and the origin server will be more solicited. The performance of distributed caches is also strongly related to the popularity profile of content that generally follows a Zipf law [16][17]. In such a distribution, the popularity of a content of rank r is proportional to $1/r^{\alpha}$ for some α. The larger the factor α, the smaller the number of very popular files, with therefore less caching needs. The smaller it is, the larger is the number of very popular files, with consequently a much greater caching requirement.

The contributions of this thesis work focus on these aforementioned issues related to caching optimization in multi-cache networks in general and more specifically in Content-Centric Networks and they can be summarized as follows. We start first by introducing an analytical model based on Markov chains named MACS (Markov chain-based Approximation of Caching Systems). This model allows us to estimate the hit probability of an LRU cache (Least Recently Used) operating under the default caching scheme where the contents are always saved when received. Then, an extension of MACS was presented by proposing a methodology that allows modeling the caching decision process independently from the strategy of caching that will be adopted. We have shown how the versatility of our cache modeling tool enables to mimic efficient caching strategies from the state-of-the-art and it can be easily adapted to represent an interconnection of caches under different schemes. We tackled after that the problem of cache resources allocation, which consists on studying how limited storage capacities should be distributed across the network's nodes in order to ensure an effective use of caches. The proposed approach solves the trade-off between minimizing the hit rate in the origin server and minimizing the distance between clients and the requested contents. To do so, the problem was modeled as a multi-objective integer nonlinear program and we proposed the use of the Greedy Randomized Adaptive Search Procedure (GRASP) due to the NP-hardness of the problem.

3 Thesis outline

The rest of the thesis is organized as follows:

— In chapter one, we start first by giving an overview of the Information-Centric Networking paradigm to focus after that on Content-Centric Network, which represents a promising ICN solution for the future Internet. Then, we analyze the existing works related to the caching feature of ICNs and of multi-cache systems in general. More specifically, we discuss the issues related to cache modeling and cache resources allocation in multi-cache networks.

— In chapter two, we present an analytical model named MACS capable of computing the hit ratio of an LRU cache [18]. By proposing a methodology that allows modeling the caching decision process in the general case, we were able to extend MACS so that it can be used to analyze different caching algorithms [19]. We aim with our proposal to gain more insights on the efficiency and limits of CCN caching strategies, with a focus on the schemes having a good performance/overhead trade-off through the analysis of different network performance metrics. We have shown how the versatility of our cache modeling tool enables to mimic existing efficient caching strategies and it can be easily adapted to represent an interconnection of caches under different schemes [20]. Even though our proposal was first used in the context of CCN networks, it can of course be applied to multi-cache networks in general. In addition to the cache hit, MACS can be used to compute different performance metrics like the content provider load and the distance reduction ratio. The conducted tests have shown the accuracy of our model in estimating the cache hit rate of a multi-cache system under different caching schemes and several conclusions were drawn on the efficiency and limits of caching that we hope it will help designing optimized caching strategies.

— In chapter three, the optimal placement of caching resources is investigated. The cache allocation issue is one of the most important problems to address when studying multi-cache networks due to the expensive cost of deploying distributed storage resources along the network. In this context, we propose an approach that solves the trade-off between minimizing the hit rate in the origin server and minimizing the distance to retrieve contents. To do so, we modeled the cache allocation problem as a multi-objective integer nonlinear program where our an-

alytic tool MACS is used to evaluate the objective functions. Our formulation of the problem turned out to be an NP-hard problem. Thus, we proposed an adaptation of the metaheuristic GRASP to solve the problem using different evaluation functions to generate the solutions [21]. The conducted tests have showed the closeness of the metaheuristic's outcomes to the optimal ones. The aim of the work presented in this chapter is to give a tool capable of efficiently allocating distributed caching resources that takes into account more than one performance metric and versatile enough to adapt to specific desired network performance and constraints.

— Finally, this dissertation closes by summarizing our thesis work and the different conclusions that were drawn and perspectives on future work are outlined for the overall thesis.

CHAPTER 1
BACKGROUND

Contents

1 Information-Centric Networking 40
1.1 Motivations 40
1.2 ICN description 40
1.3 ICN proposals 42
1.3.1 Publish Subscribe Internet Routing Paradigm 42
1.3.2 Network of Information 43
1.3.3 Data Oriented Networking Architecture 45
1.3.4 Content-Centric Network 46
2 Content-Centric Networking 46
2.1 CCN naming and message exchange 46
2.2 CCN node model 48
2.3 The Interest/Data packets routing in CCN 49
2.4 Discussion 51
3 Cache modeling in multi-cache networks 51
3.1 Related work 51
3.2 Che approximation 53
3.3 Discussion 55
4 Cache allocation in multi-cache networks 55
4.1 Related work 55
4.2 Discussion 57
5 Conclusion 58

Introduction

The aim of this chapter is to give first an overview of the Information-Centric Networking paradigm. We will start by giving a brief description of ICN and some of its important features. Then, we will focus on the legacy and state-of-the-art proposals related to the caching feature of ICNs and of multi-cache systems in general, which is the core of this thesis work. More specifically, we will discuss the issues related to cache modeling and cache resources allocation in multi-cache networks.

1 Information-Centric Networking

1.1 Motivations

In the last years, accessing content has become the most important usage of the Internet. This evolution is mostly driven by the increased popularity of content-oriented services, e.g., Peer-to Peer file sharing, Video on Demand, video/audio streaming and social networks where users focus more on contents and not on the physical locations from which contents can be retrieved. Unlike its usage, the Internet was designed for host-to-host communications. To overcome this mismatch between usage and design, many projects have been carried on in order to propose a network infrastructure capable of providing services better suited to today's applications requirements. These studies led to a new paradigm named Information-Centric Networking [10]-[11]-[4].

1.2 ICN description

ICN represents a promising new paradigm for the Internet of the future, where the communications are no longer depending on named hosts (host-to-host communications) but relies on named data (see Figure 1.1). Contrarily to IP network where users before accessing any data or starting any activity should tell the network which location they want to communicate with (IP addresses), end-users in ICN express only their interests for contents or information objects (IO) that are identified only by their names [4]. The network entities are in charge of routing the clients requests (based only on the content names) towards the machines that host an authorized copy of the demanded items. Once found, the requested contents will be delivered through the reverse paths

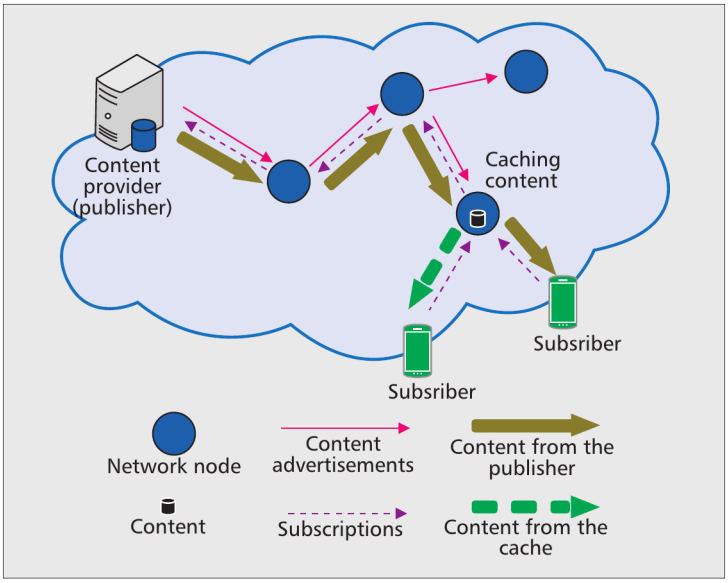


Figure 1.1 – ICN networking Overview.

to the clients, who have no need to know their locations.

ICN aims with this design to simplify the content delivery experience and to optimize the network performance by including natively into the networking design the following features:

— Multicast: it consists on aggregating in one ICN node the requests coming from different end-users who are demanding the same content. When the content is found, each requester will get a copy of it.

— Multipath: in case where a content object is available in more than one location within the network, an ICN node can send a request for a specific data through multiple interfaces in order for example to find out the best container for the requested content.

— Content security: ICN adopt a content-oriented security model. Unlike IP network where safeness is achieved via securing the communication channels, the contents in ICN are protected with the content provider's signatures and only the authorized users can verify their validity by resolving the signature.

— In-network caching: in order to enhance the network resource usage and data availability, the ICN architecture exploits the possibility of the in-network caching. The network nodes, equipped with a local cache store, can save contents or pieces of them that are flowing in the network and thus, the next requests of the cached data will be served more rapidly.

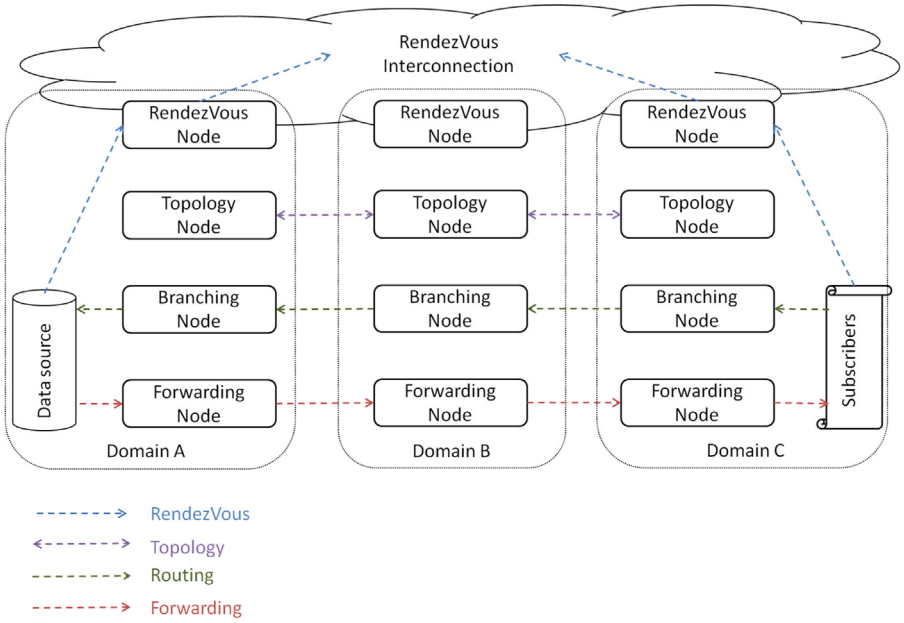


Figure 1.2 – Overview of PSIRP [22].

In the last years, several projects have been put in work by several research teams with the objective to carry out an ICN concrete solution. Among the most important of those works results, we can quote:

— Publish Subscribe Internet Routing Paradigm or PSIRP [22].
— Network of Information or NetInf [23].
— Data Oriented Networking Architecture or DONA [24].
— Content Centric Network or CCN [12].

1.3 ICN proposals

1.3.1 Publish Subscribe Internet Routing Paradigm

This approach is based on the publish/subscribe concept where hosts can publish data or Named Data Object (NDO) in the network and subscribe to any published data. Publishing content in the network does not imply a data transfer, this transfer only take place when a node subscribes to an NDO. The network, in this case, looks for the publication and establishes a delivery path from the publisher to the subscriber [22].

The PSIRP network architecture is basically composed of four modules (Figure 1.2):

— The RendezVous module: which consists on a distributed database that maps the requested data to the subscriber.

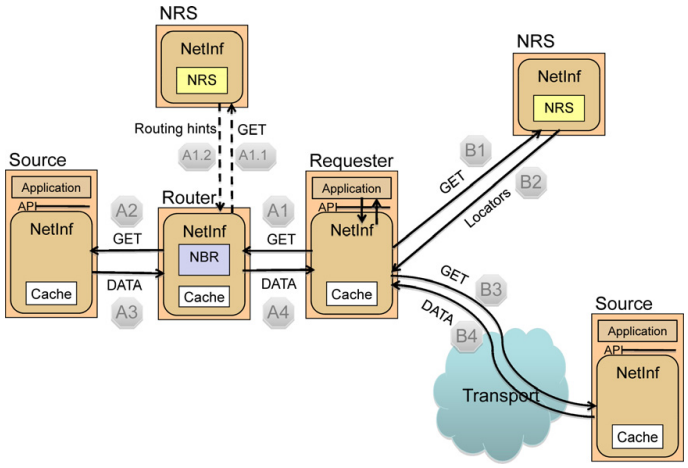


Figure 1.3 – Routing scheme of NetInf [23].

— The forwarding module: it handles the data delivery from one location to another based on the labels of the packets.
— The topology module: it manages the forwarding traffic accomplished by the forwarding module by creating and maintaining delivery trees.
— The Branching module: it provides a routing map for routing the subscriber requests toward the inter-domain or intra-domain content locations by using the topology maintained by the topology module.

The PSIRP network is divided into Domains and each domain has one RendezVous Node (RN), one Topology Node (TN), one Branching Node (BN) and one or several Forwarding Nodes (FN). A subscriber can express its subscription of a content to the local RN of its domain to get the content container. Then, the BN uses the networking topology managed by the TN to forward the subscription to the content location. The FNs finally ensure the returning of the required content to the subscriber.

1.3.2 Network of Information

The Network of Information provides access to Named Data Objects (NDO) that are independent of the objects' locations in the network topology. To access and distribute data in NetInf, three pairs of messages are used:

— Search/Search-Resp: the requester uses the Search message to send a set of queries. A node receiving the Search message will respond to the requester using the Search-Resp message if it has the NDO demanded in its cache. Otherwise, it will forward the Search message.

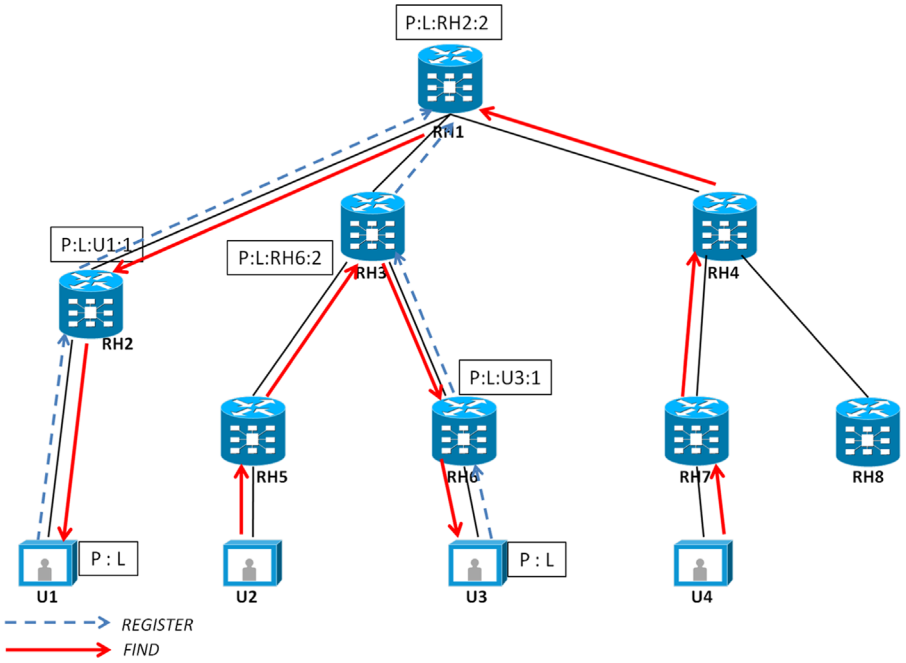


Figure 1.4 – Routing scheme of DONA [24].

— Publish/Publish-Resp: the publisher can post an NDO with a Publish message using a Name Resolution Service (NRS). The node that receives the Publish message will either choose, depending in the local policy adopted and the availability of resources, to cache the data and returns a Publish-Resp message or to forward the Publish message to other nodes.
— Get/Get-Resp: a requester, demanding an NDO from the network, will approach the NRS using the Get message, which will lead him to the information publisher. A node would send a Get-Resp message to respond to the Get request.

Figure 1.3 shows an example of a routing scheme in NetInf. The name-based routing (NBR) (steps from A1 to A4) forwards a GET request between the NetInf nodes until a cached copy of the NDO or a server having the requested data is found. In case where the router does not have enough routing information to perform an NBR (step A2), it can perform a name resolution step (steps from A1.1 to A1.2) before forwarding the request (step A2) based on the retrieved routing hints. The object can be cached on the return path in intermediate nodes for subsequent demands. Alternatively, the initial client can query an NRS (steps from B1 to B4) via a GET message to resolve the object name into a set of routing hints. Subsequently, the routing hints are used to retrieve the object via the underlying transport network.

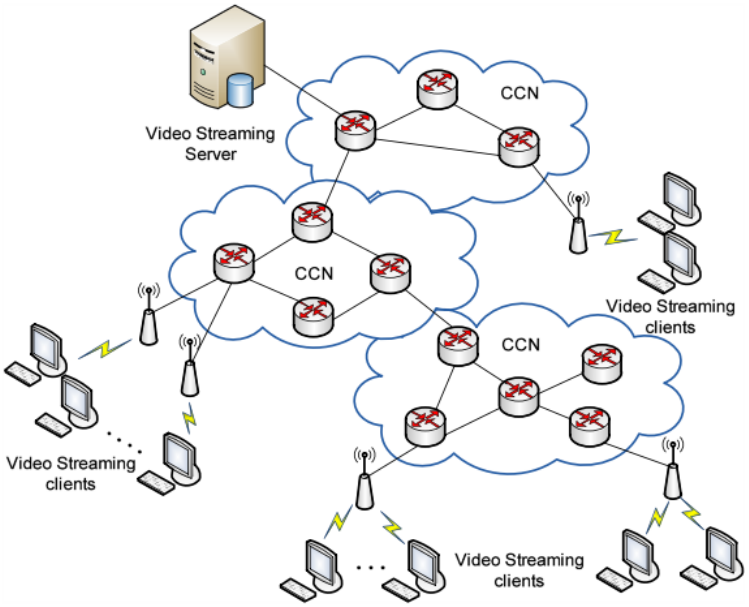


Figure 1.5 – Overview of CCN [12].

1.3.3 Data Oriented Networking Architecture

The DONA relies on a new class of network entities called Resolution Handlers (RHs). Name resolution is conducted through employing two basic functions:

- Register: Register packets are used by nodes that are approved to act as data sources to register their Named Data Objects (NDO) within the RHs.
- Find: to locate and access to specific data (identified by NDO), clients use Find packets.

An example of the routing mechanisms in DONA is showed in Figure 1.4. Each RHs keeps up a registration table that contains three tuples:

- The content name “P : L”, where P and L represent respectively the content provider and the content label chosen by its provider.
- The next hop from where the node receives the content name advertisement.
- The distance.

A client, seeking access to certain data, issues a Find packet to the local RHs. Once receiving the client request and depending on the information provided by the registration table, the Find packet will be routed hierarchically in order to find the demanded content that is closest to the user. When a copy of the requested data is found, it will be returned over the reverse RHs path.

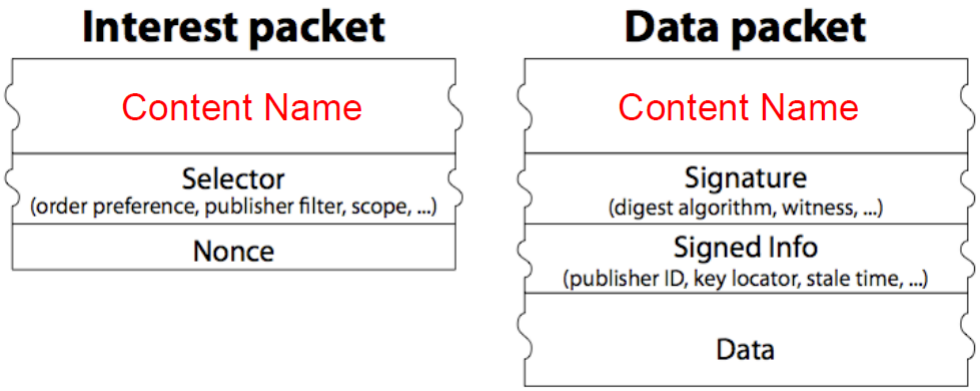


Figure 1.6 – CCN packet types [12].

1.3.4 Content-Centric Network

To request and retrieve data in CCN (Figure 1.5), two types of packets are used:

— Interest packet: it contains the content name requested by a client.

— Data packet: it is used to deliver data to its destination.

Clients can ask for specific data objects by sending Interest packets, which are forwarded towards data sources. Then, using the Data packets, data objects will be routed back to the clients on the reverse path and any node along this path can store a copy of the data to eventually answer to future requests for the same content.

Among these ICN solutions, we choose for our study the Content Centric Networking, which has quickly gained consensus in the scientific community and represents a promising architecture in ICN networks for the future Internet.

2 Content-Centric Networking

2.1 CCN naming and message exchange

Content-Centric Networking [12] was proposed in 2009 by Palo Alto Research Center (PARC). CCN proposes a complete content-based networking model (content naming, content-based routing, etc.) where a content is accessed by its name instead of the IP address of the host possessing it.

The communication and data retrieval in CCN [12] are managed using two types of packets (Figure 1.6):

— Interest packet: it is used to send the user's requests. It carries:

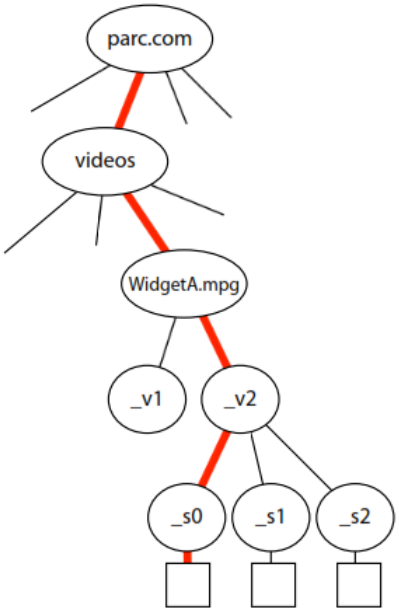


Figure 1.7 – The naming of contents in CCN [12].

- ▶ The content name.
- ▶ Selectors: which are optional elements used for discovering and selecting the data that matches best to what the application wants.
- ▶ Nonce: it is a sequence of random number by which a router detects whether the interest is duplicated or not.

— Data packet: it contains the following fields:

- ▶ The content name.
- ▶ Signature and signed info: according to the signed info, users can get the public key of the content publisher to verify the received data with the rule specified in the signature field and decide to accept the data or not.
- ▶ Data: which is the data that match the content name in the interest packet.

A user asks for a specific content by broadcasting its interest in the network. Any node that receives the interest and has the solicited data can respond with a Data packet and in that case, the interest will be consumed.

As mentioned before, the content name in CCN is the only identifier of data and the information exchanges that are based on establishing communication channels is abandoned. Names are hierarchically structured like a path-name of a file system (see Figure 1.7).

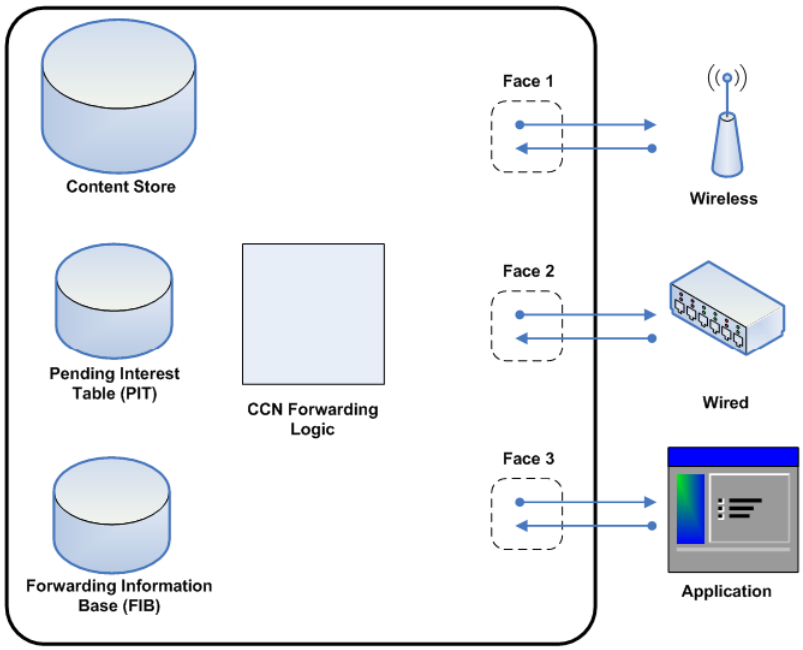


Figure 1.8 – The node model in CCN [12].

The tree-like structure enables the aggregations of contents and thereby facilitates the content discovery. For example, as shown in the Figure 1.7, all the content supplied by “parc” can share the same prefix “parc.com/”. A same content object in CCN can have many versions, and it can be divided into multiple fragments called “chunks” in order to adjust the transport layer. To simplify content discovery and forwarding, the content name generally finishes with the version and the chunk information.

2.2 CCN node model

In a CCN network, each node is basically composed by three data structures (Figure 1.8):

— Content Store (CS): the CS is a buffer set that provides the in-network caching feature by caching the data that passes by in order to optimize the network performance (minimizing the latency, reducing network overload, etc.).

— Pending Interest Table (PIT): it consists on a table that contains two types of information (see Table 1.1): the contents names within the received Interest messages and the associated incoming faces (the faces play a similar role of interfaces in IP network). The PIT has two roles. The first one consists on keeping

Table 1.1 – An example of a CCN PIT.

CONTENT NAMES	IN FACES
ccn:/spotify.com/music1.mp3	301
ccn:/irisa.com/documents/doc1.docx	201,203
ccn:/youtube.com/videos/video1.mp4	102,105
...	...

Table 1.2 – An example of a CCN FIB.

PREFIXES	OUT FACES
ccn:/spotify.com/	101
ccn:/irisa.com/documents/	204,206
ccn:/youtube.com/videos/	301,302
...	...

track of the forwarded interests that are awaiting for a returned Data packet by saving the content name of the requested content and the associated face. The second role is to avoid forwarding multiple Interests that request the same content; when a CCN node receives from different faces several Interest messages that are demanding the same content, these Interest packets will be aggregated in one entry in the PIT and only the first one is routed.

— Forwarding Information Base (FIB): The FIB is used to guide Interest packets toward potential sources of the requested data by saving in a table two types of information (see Table 1.2): the content name (or the aggregated prefixes of the names) and the associated outgoing faces. A content having a list of outgoing faces means that a data can have multiple sources.

2.3 The Interest/Data packets routing in CCN

In CCN networks, only Interest packets are routed and as they head upstream at potential data sources, they leave a track of “bread crumbs” so that Data packets can follow to get to the requester(s) [12].

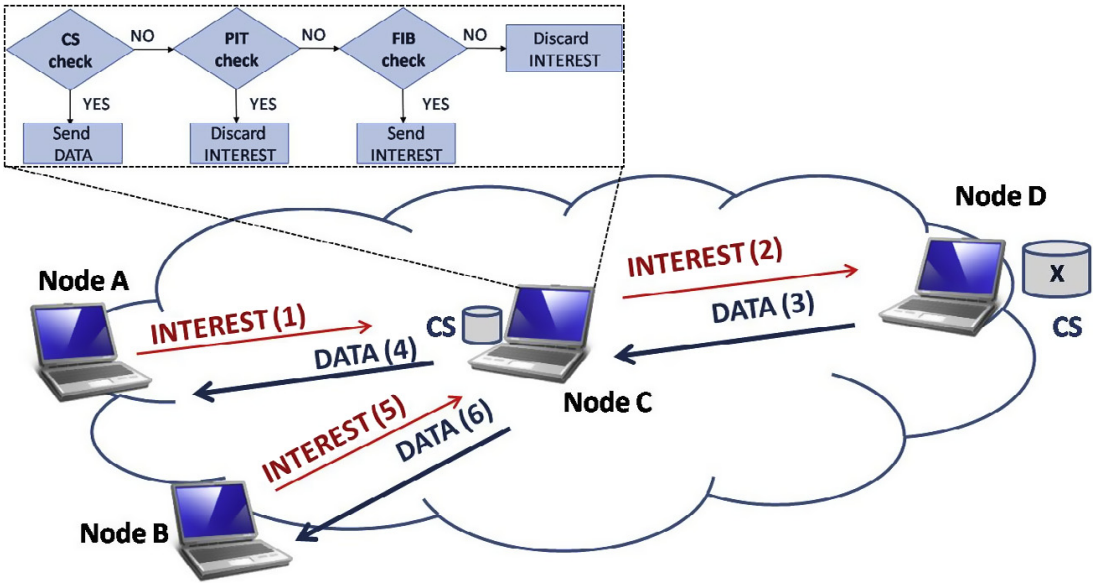


Figure 1.9 – Packets processing in CCN [25].

The packet processing and forwarding in CCN nodes is performed as follows:

1. First, a CCN node receives an Interest packet where the content name carried by this packet is checked in the content store.
2. If there is a Data packet in the CS that matches the request, it is then sent back through the incoming face of the processed Interest. Otherwise, the content name is checked in the PIT.
3. If there is a matching entry in the PIT, then the Interest packet is discarded and his incoming face is added to the existing PIT entry (aggregation of Interest packets). If not, a new entry is created and the Interest is forwarded based on the information of the FIB table.
4. When the request content is found, the Data packet will track the traces left by the Interest packets in order to reach its requester(s).
5. Once a node receives a Data packet, its content name is checked in the CS.
6. If there is a match, the Data packet should be discarded since it already exists in the CS. Otherwise, the content name is looked up in the PIT.
7. If a matching entry in the PIT is found, then the Data packet is sent out through the faces where the Interest was received. The corresponding PIT entry is then deleted and based on the caching strategy, the node can keep a copy of the content in the CS.

An example of the packets processing and forwarding in CCN is shown in Figure 1.9. Once receiving the Interest Packet from node A (who requests the object x), the node C which doesn't find a match in its content store nor in the pending interest table, forwards the packet to the source node D. When the node C receives the Data packet from D, it forwards it back to the requester A and can thereafter with its cached copy serve the request coming from node B for the content x.

2.4 Discussion

As mentioned before, the on-path caching functionality is one of the most important features provided by CCN. It plays a major role on increasing the network performance. That is why it is crucial to use efficiently the cache resources in order to realize a suitable content retrieval. The caching optimization can be treated at different levels:

— Cache replacement algorithm: it is the algorithm responsible of managing the cache. When the cache is full, an item is chosen by the algorithm to be discarded. We can cite from the existing algorithms First In First Out (FIFO), Least Recently Used (LRU), etc.

— Cache decision policy: it determines which content should be cached and at which node. In other words, a node should decide when receiving a content and depending on the applied strategy whether to cache or not the data.

— Cache resources placement: since the cache capacity that can be allowed to a CCN node is limited, cache placement consists on studying the optimal distribution of storage resources between the network's nodes (e.g.: adopting a fair distribution, giving more storage spaces to edge routers, etc.).

The focus on this thesis was on problems related to cache modeling and cache resources allocation in CCN and multi-cache networks in general. A state-of-the-art pertaining to these subjects will be presented in the next two sections.

3 Cache modeling in multi-cache networks

3.1 Related work

Many studies have been conducted these last years to deal with the performance analysis of a single cache and a network of caches using mathematical modeling [26]

[27] [28]. We limit ourselves to the papers most relevant to our work.

Several works modeled and analyzed the behaviour of a caching system using Markov Process. The study in [29] was probably the first attempt to evaluate and model caching systems. The author proposed an exact model for predicting the buffer hit probability under the LRU and FIFO replacement policies using Markov chains. Flajolet and co-authors [30] derived a simpler alternate expression of the cache hit rate to the one presented in [29] using a different approach. Unfortunately and in both cases, the computational complexity grows exponentially with the cache size N and the number of data items R. In [31], Dan and Towsley proposed an approximate analysis of the cache behaviour by exploiting the stack properties of an LRU cache using Markov chains. They derived an iterative algorithm with a complexity of $O(NR)$ to predict the hit probability for a cache of size N using the hit rates of a cache of size $N+1$ under the LRU replacement policy. The proposed solution is, however, limited to the case of a single cache and the tests were done only with very low values of cache size and catalog capacity. In [32], the authors extended the algorithm of [31] to the case of a network of caches. Psaras et al. proposed, in [33], a Markov chain-based model to estimate the proportion of time a given piece of content is cached in the case of a single router, which then has been extended to cover the case of multiple caching nodes. However, the assumptions and approximations made by the authors in their cache model can lead to high error rate in terms of hit ratio accuracy.

One of the most interesting proposals related to cache modeling was presented in [34] by Che and co-authors. Using a different method compared to the previous mentioned works, Che et al. propose a simpler and more efficient approach to estimate the hit rate of an LRU cache. Their solution, known as the “Che approximation”, allows identifying a characteristic time approximation for each item in the cache, which was used to estimate the cache hit rate per content. Their proposal has proved to be very accurate, with a low complexity ($O(R)$). Their work was further investigated in [35], where the authors provided a mathematical explanation for the success of the Che approximation even in configurations that are more general than those presented in the original work. In [36], the authors extended the work of [34] and proposed Time-To-Live (TTL) based caching model that assigns a timer to each content stored in the cache and redraws it every time the content is requested. Nevertheless, the aforementioned proposals are applicable only when the Leave Copy Everywhere caching strategy (LCE) is used, where every data packet is always stored once received by a caching node.

Other studies have been conducted in order analyze the modeling behavior and performance of caching systems where alternative caching strategies to the LCE are considered. By means of the "Che approximation", Laoutaris et al. presented in [37] an analytical model of their caching algorithm called *Leave Copy Down* (LCD). Instead of storing the contents in all intermediate nodes like in LCE, LCD consists on caching an item only in the immediate downstream cache of the hit location. Their model was validated only in the case of two-cache LCD/LRU tandem with a small sized catalog of contents. In [38], the authors extended the work of [34] and proposed a unified framework to analyze the performance of caches (both isolated and interconnected). Their model covers various insertion and eviction policies (including LRU and LCE). They evaluated the accuracy of their proposal through simulation. However, in the case of interconnected caches, the tests were conducted only in the case of a 6-nodes chain with fixed network settings. Extending the work presented in [34], the authors in [39] developed algorithms to approximate the hit probability of the cache replacement policy LRU-K [40] [41] and variants of it. Nevertheless, their proposal is limited only to the case of a single cache.

If we look closely at the state of the art and the aforementioned works, we can see that most of the advanced proposals related to cache modeling are based on the work presented in [34] by Che et al., which is referred to as the "Che approximation". A more in-depth analysis of this work will be presented in the next section.

3.2 Che approximation

In [34], Che et al. proposed a modeling technique to analyze the caching performance by estimating the hit rate of a cache where the LRU replacement policy is used. Their work was investigated further in [35], where a mathematical analysis for the remarkable accuracy of the approximation is given.

Consider a cache with a capacity size of N items operating under the LRU algorithm and let $C = \{c_1, \ldots, c_R\}$ be the set of catalog's contents available for users and that can be saved in the cache. Following the IRM model [35], the probability p_r to request an item c_r from the set of available contents in the network is constant and follows a popularity law (e.g. Zipf law), where the contents are ranked decreasingly according to their popularity from one to R. Let's define τ_r as the generic exponentially distributed inter-request interval for object c_r. In other words, it represents a sequence of inde-

pendent exponential random variables with respective rates (p_r). We define now the following random variables, for $t \geq 0$,

$$\begin{aligned} X_r(t) &= \sum_{i=1, i \neq r}^{R} 1_{\{\tau_i < t\}}, \\ T_C(r) &= inf\{t > 0 : X_r(t) = N\}. \end{aligned} \tag{1.1}$$

$X_r(t)$ and $T_C(r)$ are respectively the number of how much distinct items were requested up to time t (excluding content c_r) and the time at which exactly N different items, other than c_r, have been requested. Therefore, a hit for content c_r will occur in the cache if $X_r(\tau_r) < N$, which is equivalent to $T_C(r) > \tau_r$. If we denote by $h(r)$ the hit probability in the cache of an object c_r, then we have

$$h(r) = P(T_C(r) > \tau_r) = 1 - e^{-p_r T_C(r)}. \tag{1.2}$$

Since at $T_C(r)$, we have exactly N objects in the cache, then

$$N = \sum_{i=1, i \neq r}^{R} 1_{\{\tau_i < T_C(r)\}} = \sum_{i=1, i \neq r}^{R} 1 - e^{-p_r T_C(r)}. \tag{1.3}$$

In [34], the following two approximations were made:

1. $T_C(r)$ is a constant for each item c_r ($r \in \{1, 2, \ldots, R\}$).
2. $T_C(r)$ is a constant with respect to c_r, which will be denoted T_C.

The idea behind these two approximations as explained in [34] is based on the following intuition: as the aggregated arrival rate of all the content requests grows and as long as the cache size is reasonably large, this rate becomes more and more deterministic and reaches a constant. These approximations are made not only for the tractability of the mathematical development, but also for the identification of T_C as an important characteristic time of a given cache. The first approximation implies that the random variable $T_C(r)$ is now a constant that solves

$$N = \sum_{i=1, i \neq r}^{R} (1 - e^{-p_r t}), \tag{1.4}$$

and the hit rates are approximated by

$$h(r) = 1 - e^{-p_r T_C(r)}. \tag{1.5}$$

The second approximation means that $T_C(r) = T_C$ for $r \in \{1, 2, \ldots, R\}$, where T_C is the unique root of the equation

$$N = \sum_{i=1}^{R} (1 - e^{-p_r t}). \tag{1.6}$$

With these approximations techniques, the problem of modeling the hit performance of an LRU cache is now greatly simplified since the interaction between the caching process and a content c_r and all the other operations is mediated by the "characteristic time" T_C.

To sum up and following the Che approximation, the hit rate $h(r)$ of an item c_r ($r \in \{1, 2, \ldots, R\}$) is equal to

$$h(r) = 1 - e^{-p_r T_C}, \tag{1.7}$$

where T_C solves

$$N = \sum_{i=1}^{R} (1 - e^{-p_r t}). \tag{1.8}$$

3.3 Discussion

We presented and discussed in this section most of the important state-of-the-art proposals related to cache modeling. As we have seen, the work of Che et al. [34] is one of the most interesting proposals when dealing with cache modeling and the majority of the works conducted after the publication of the "Che approximation" were built upon on it. Their approximation was first intended to deal with LRU caches operating under the LCE policy and extending it to other caching schemes is not straightforward. We will present later in chapter two our first contribution in this thesis, which consists on a Markov chain-based analytic tool capable of modeling the caching decision process in general in order to be used to model many caching schemes, without being limited to a specific algorithm.

4 Cache allocation in multi-cache networks

4.1 Related work

Many studies have investigated the problem of cache resources allocation and placement in the context of multi-cache networks [42] (e.g., CDN, ICN, etc.).

The work presented in [43] by Krishnan et al. was one of the earliest studies that tackled the problem of the optimal cache resources placement. They examined the cache location problem in the case of transparent in-route caches in the context of web caching. The problem was modeled as a k-median problem, where the objective is to minimize the network traffic flow, and solutions were proposed based on dynamic programming and greedy heuristics. The classic k-median problem consists on finding k centers (i.e. cache resources) such that the clusters they form are the most compact (i.e. lower cost). The authors in [44] and [45] studied the storage capacity allocation in hierarchical content distribution systems through a multi-commodity problem, which generalizes the single commodity k-median problem. They propose a two-step algorithm capable of solving such problem when tree graphs are considered, which has been extended through approximations to cover the case of general graphs. They also provided a greedy algorithm due to the high complexity of the exact solution. The authors in [46] addressed the problem of placing mirrors of Internet content on a restricted set of hosts through modeling it as a slightly different version of the minimum k-center problem, considering the latency as the cost function to be optimized. In the minimum k-center problem, which is an NP-complete one, the objective is to find a placement of a given number of centers (i.e. content servers) such that the maximum distance from a node (i.e. end-user) to the nearest center is minimized. They proposed a greedy algorithm and a heuristic based on nodes degrees to solve the problematic.

Later on and with the proliferation of Content Delivery Networks, many works have focused on replica server (or cache resources) placement solutions in traditional and emerging CDN-based paradigms (cloud-based CDN, NFV-based CDN, etc.). In [47], the authors proposed a solution to the media server placement problem by modeling it using the uncapacitated facility location problem. To ensure the scalability of their algorithm, they have considered the case where all the end-users locations can be potential placement of replica servers. In [48], Rodolakis et al. introduce polynomial and pseudo-polynomial algorithms to solve the replica server placement problem using the splittable soft capacitated facility location model. Their aim is to find the best location of replica servers and the number of servers that should be used and assigned to end-users groups in a way that minimizes the cost and satisfies the Quality of Service (QoS). In [49], Chen et al. studied the problem of building distribution paths and placing Web server replicas in cloud CDNs to minimize the cost incurred on the CDN providers while satisfying QoS requirements for user requests. They provide an ILP formulaion

of the problem that happened to be NP-hard, and propose offline and online heuristics to solve it. In [50], the authors presented a CDNaaS platform (CDN as a Service) that enables the creation of CDN slices across multiple cloud domains and the deployment of virtual resources (including virtual caches) from multiple IaaS providers (Infrastructure as a Service), where different VNFs (Virtual Network Functions) are running. They studied in their work the optimal placement of these VNFs by modeling this problem as two distinct Linear Integer problems, where the aim is to minimize the incurred cost of resource placement and maximize the Quality of Experience (QoE) of the virtual streaming service. By means of the bargaining game theory, they propose a solution that ensures a fair trade-off between the cost and the QoE.

The study in [51] was probably the first attempt to investigate the cache allocation problem in CCN. They used in their study different metrics to measure the centrality of routers like degree, closeness and betweenness in order to decide where the cache should be distributed along the network's nodes. They suggest that deploying more cache resources at the core nodes of the network is better than at the edge. In later works [52] [53], the authors have concluded the opposite, suggesting that placing larger caches at the edge is more effective. Wang et al., in their work [54], have studied the impact of content popularity distribution on caching performance in CCN. They show that placing caches into the network core is better suited for content requests with uniform distribution and that in case of highly skewed popularity demands patterns, pushing cache resources to the edge yields better performance.

4.2 Discussion

Considering a single metric or objective when dealing with the cache allocation problem clearly reduces the complexity of the problem, but has led many studies to find results that appear contradictory. There are many aspects in our proposal in this matter, which will be presented in chapter three, that makes it different from the existing works. We propose a versatile solution that takes into account at the same time more than one performance metric to solve the cache allocation problem and it can be tuned in order to seek some specific results. Moreover, the proposed tool builds a solution by measuring the contribution of all the nodes by the means of an analytic model capable of estimating the network performance, which will allow to take into account the impact of a node's performance on the others. In addition, more than one solution can be

generated for the same use case, which gives more flexibility and enables adapting to additional constraints. We will see later that placing most (or all) of the cache resources at the edge or in the network core cannot be an absolute solution since it will depend on what performance metrics we try to optimize and what outcomes we aim to obtain.

5 Conclusion

We started this chapter by exposing a new paradigm in the computer network area called ICN. We focused after that on one of the ICN solutions, which is the Content-Centric Networking. We have given the necessary theoretical background to help understand the functioning of CCN and specifically its in-network caching feature and how it can improve the network performance. Then, we presented and discussed state-of-the-art proposals that dealt with cache modeling and cache allocation problem and what more could be done when studying these two issues. In the next two chapters, we will present our major contributions related to the different problems that we discussed earlier.

CHAPTER 2

MODELING MULTI-CACHE NETWORKS USING MARKOV CHAINS

Contents

1 Motivations 60

2 System description 61

3 A single LRU cache model under LCE 62

3.1 A comprehensive analysis of LRU caches 62

3.2 Markov chain-based approximation of an LRU cache 65

4 A general Markov chain model of a single LRU cache 68

4.1 A comprehensive analysis of generic LRU caches 68

4.2 A generic model of a single LRU cache 68

4.3 Single cache model for 2Q 72

4.4 Single cache model for LCD 74

5 Multiple caches systems 75

6 Model Evaluation 77

6.1 Tests environment 77

6.2 Model results and analysis 79

6.3 Caching algorithms comparison 85

7 Conclusion 91

Introduction

The previous chapter presented the Information-Centric Networking paradigm with an emphasis on the in-network caching and the different issues related to this feature. In this chapter, we introduce an analytical model based on Markov chains named MACS (Markov chain-based Approximation of Caching Systems) [18]. This model allows us to estimate the cache hit probability under the popular Least Recently Used replacement scheme for a system with multiple caching nodes, where the default caching strategy of CCN, called Leave Copy Everywhere (LCE), is used. Then, an extension of MACS [19] is presented. We proposed a methodology that allows modeling the caching decision process in a more general way, so that it can be used to analyze different caching algorithms other than LCE. We will show how its versatility enables it to mimic efficient caching strategies [20] such as Leave Copy Down (LCD) [37] and Two Queue (2Q) [40]-[41]. We aim with our model to gain more insights on the efficiency of CCN caching strategies, with a focus on the schemes having a good performance/overhead trade-off, by analyzing different network performance metrics like cache hit rate, content provider load and distance reduction ratio. It has to be noted that, even though our model MACS was first proposed in the context of CCN networks, it can also be applied to multi-cache networks in general (e.g., CDN, cloud based CDN, NFV based CDN, etc.).

1 Motivations

Due to the Internet usage evolution over these last years, the current IP-based architecture becomes heavier and less efficient for providing some Internet services. Beyond Content Delivery Networks, Network Operators (NOs) are developing caching capabilities within their own network infrastructure, in order to face the rise in data consumption and to avoid the potential congestion at peering links. Indeed, disseminating caches in the infrastructure not only helps in absorbing the network's congestion, but in addition, brings content closer to users, which allows a reduced latency. These factors explain the enthusiasm of industry and academics around the Content-Centric Networking concept and its in-network caching feature as a mean to improve network's performance and services scalability.

Table 2.1 – Summary of the notations.

Term	Desciption
M	Number of nodes in the network
R	Number of items in the catalog
N	Cache size
S	State space of an LRU cache model
c_r	Content with a popularity/rank r
p_r	Probability to request an item c_r
p'_r	Probability that a node receives an item c_r in a multi-cache network
α	Zipf law distribution parameter
$\beta(r)$	Cache decision probability
$\gamma'_s(r)$	Probability of a content c_r staying at the same position s of the cache
$\pi(r)$	Steady state distribution of the Markov chain
$P_{hit}(r)$	Hit probability of content c_r
$P_{miss}(r)$	Miss probability of content c_r
$MS(r,u)$	Outgoing miss stream ratio from a node u for a content c_r
$\eta_r(v)$	Incoming miss stream ratio at a node v for a content c_r
$Dist(i)$	Distance from where the clients requests were generated to the node v_i

2 System description

Let's recall that in CCN, the content's name is the only identifier of data. To request and retrieve data, two types of packets are commonly used [12]: Interest Packet and Data Packet. Clients can ask for specific data objects by sending Interest packets, which are forwarded towards the data sources using the Forwarding Information Base. A record of the forwarded Interests is kept in the Pending Interest Table in order to keep track of the Interests waiting for a data packet. When a node receives multiple requests for the same content, the Interest packets will be aggregated in one entry in the PIT and only the first one is routed. Once the requested content is found, it is automatically routed back to the clients on the reverse path. All the nodes along this path can store a copy of data to answer future demands.

Let $G = (V, E)$ be the graph representing a general network of caches, where $V = \{v_1, \ldots, v_M\}$ depicts the nodes of the network and $E \subset V \times V$ is the set of links connecting the nodes. Each node in the network is equipped with a caching module used to store contents locally. Let $C = \{c_1, \ldots, c_R\}$ be the set of the catalog's contents available for the users. We assume that all the accessible contents in the system have an identical size and are divided into small packets or chunks, which are in turn of the

same size. The cache capacity is then expressed in terms of the number of contents or chunks that can be stored. All the available contents are stored permanently at one or more servers attached to some nodes within the network. In the rest of the paper and for the sake of readability, we will use the term node/cache interchangeably as well as the terms rank/popularity and content/item/object.

Clients, which are attached to the network nodes, send requests into the network looking for contents. The pattern of these requests is characterized by the Independent Reference Model (IRM) [31]. Suiting the IRM model, users generate an independent and identically distributed sequence of requests from the catalog C of R objects. Specifically, the probability p_r to request an item c_r from the set of available contents in the network is constant and follows a popularity law, where the contents are ranked decreasingly according to their popularity from 1 to R. Since about 80% of Internet traffic is generated by video related applications [2], we address in our work videos services, where the contents feature a skewed popularity distribution. As already argued in many previous studies [55], the latter fits the Zipf law: the probability to request the content of rank r is: $p_r = r^{-\alpha} / \sum_{i=1}^{R} i^{-\alpha}$, where α, the skew of the distribution, depends on the type of the accessible objects [35]. For our approach, we only need to assume that $0 < p_r < 1$, with $R \geq 2$.

We use here the LRU algorithm to manage the node's content store. Other memory management algorithms have been studied in the case of cache modeling like First In First Out (FIFO) and Random Replacement (RR). However, LRU is known to perform much better than FIFO [56] and the expected long-run performances of both RR and FIFO replacement policies were shown to be the same [57]. The Least Frequently Used (LFU) algorithm, which requires a more complex replacement process, is also interesting. However, it presents several shortcomings related to the insertion of new popular contents or the purge of older ones [58]. Moreover, the complexity of the LFU functioning makes it harder to be modeled and analyzed.

3 A single LRU cache model under LCE

3.1 A comprehensive analysis of LRU caches

Let's consider a single node in a Content-Centric Network operating under the LRU replacement policy and having clients attached to it. Whenever a user requests a con-

tent of rank r in the catalog, it will generate either a cache miss if the content is not present in the cache or a hit otherwise. In the latter case, the object will be sent back to the user. In the case of a cache miss, the client's interest is forwarded to the next nodes in the direction of the nearest content server storing a permanent copy (i.e. origin server). Once the object located, it is sent on the reverse path and depending on the caching strategy used in the network, the content will be cached or not at each node that passes by it. In CCN, a node uses a caching scheme in order to decide whether an incoming item should be saved or not. In the case where the default caching strategy LCE is used [37], the contents will always be cached at each node that pass by it.

Consider a cache sized to contain $N \leq R$ items (the usual case is, of course, $N \ll R$). The position occupied by a content in a cache goes from 1 to N, where block 1 represents the top up position and N is the bottom down one (last block before the eviction from the cache). Whenever a local cache hit or a caching decision occurs for a content c_r, it will then be placed at the top position in the cache. Consequently, for any given content $c_{r'}$ occupying position i in the cache, with $r' \neq r$, three actions are possible when a request for c_r is received:

— $c'_{r'}$ will be moved down if the requested content c_r is either not present in the cache or if it occupies block j with $j > i$;
— $c'_{r'}$ will remain at the same position if c_r occupies a block j with $j < i$;
— $c'_{r'}$ will be evicted from the cache if it occupies the N^{th} and last position.

When analyzing the performance of a caching system, we usually focus on hits or misses, which means that we consider the system only when requests for content arrive. This means that in case of a miss, we don't look at the time needed to get a copy of that content in the cache; we only consider which contents are in which positions at arrival times.

Let us call a *configuration* of the cache any vector $\vec{x} = (x_1, \ldots, x_N)$ where x_i is the content present at block or position i. We only consider the case where all positions are occupied because the cases where the cache is partially filled concern only the initial transient phase. Let us denote by S the set of all possible configurations; we have $|S| = R!/(R-N)!$. For realistic values of N and R, $|S|$ is huge. If we consider the evolution of the cache's configuration "just after" the arrival of a request, we define a discrete time homogeneous Markov chain $X = (X_n)$, where X_n is the configuration just after the arrival of the n^{th} request at the node. Chain X is clearly irreducible: from any configuration $(x_1, \ldots, x_N)$, we can move to any other configuration $(y_1, \ldots, y_N)$ after the

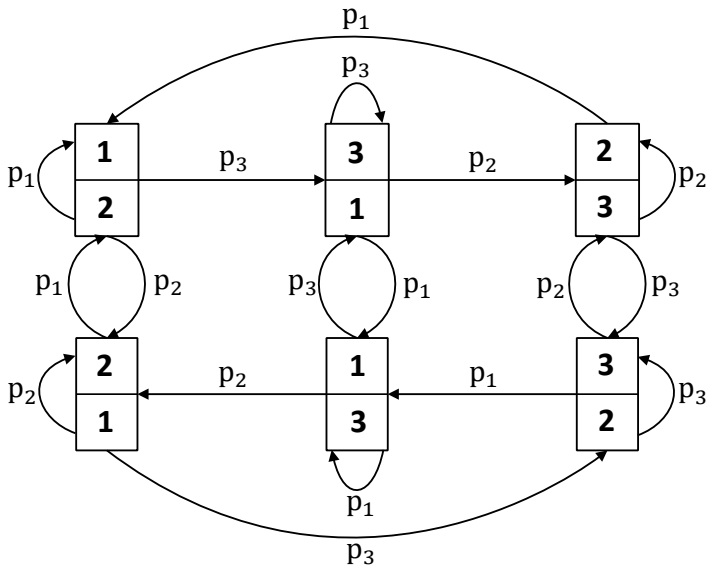


Figure 2.1 – A Markov chain model of an LRU cache where the LCE strategy is adopted with $R = 3$ and $N = 2$.

arrival of a request for content y_N, then for content y_{N-1}, etc. (this event has probability $p_{y_1} \cdots p_{y_N} > 0$). Of course, we consider the case here of always deciding to store the arriving content. It is also aperiodic because if $X_n = (x_1, \ldots, x_N)$, then, with probability p_{x_1} next state is still $(x_1, \ldots, x_N)$. So, X is ergodic.

Knowing the steady-state distribution of X, $\left(\pi_{\vec{x}}\right)_{\vec{x} \in S}$, allows (in theory) to evaluate important performance metrics of a caching system such as the hit probability per content. Assuming the system is in equilibrium, we have that for any $c_r \in C$,

$$\Pr(\text{hit for a request of } c_r) = \sum_{\vec{x} \text{ s.t. } \exists i \in \{1,\ldots,N\} | x_i = c_r} \pi_{\vec{x}}. \tag{2.1}$$

For illustration purposes, let us consider the case of $R = 3$ and $N = 2$, when the content is always cached when received (see Figure 2.1). The state space is

$$S = \{(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)\}.$$

The stationary distribution $\pi_{\vec{x}}$ satisfies the equations

$$\begin{cases} \pi_{(i,j)} = p_i(\pi_{(i,j)} + \pi_{(j,i)} + \pi_{(j,k)}), \quad k \notin \{i,j\}, \\ \sum\limits_{(i,j) \in S} \pi_{(i,j)} = 1. \end{cases} \tag{2.2}$$

Solving the equations (2.2), we have that for a generic configuration $(i, j) \in S$,

$$\pi_{(i,j)} = p_i p_j (1 - p_i)^{-1}.$$

If we want to compute for example the hit probability of content c_1, we obtain

$$\Pr(\text{hit for a request of } c_1) = \pi_{(1,2)} + \pi_{(1,3)} + \pi_{(2,1)} + \pi_{(3,1)} = \frac{p_1(1 - p_2 p_3)}{(1 - p_2)(1 - p_3)}.$$

We can see through this example how the steady-state distribution of the Markov chain representing the dynamics of an LRU cache can be used to evaluate important performance metrics of the caching system. By investigating further the exact Markov chain-based model of on LRU cache, we have found that in the general case (i.e. $2 \leq N < R$),

$$\pi_{\vec{x}} = \prod_{i=1}^{N} \Big(p_i \, (1 - \sum_{j=1}^{i-1} p_j)^{-1} \Big), \quad \vec{x} = (x_1, \ldots, x_N). \tag{2.3}$$

To compute the cache hit probability of any content c_r, we use the formula (2.1), which as we can see is very costly, as it requires a summation over the permutations of all the subsets of size N containing the content index r of the set $\{1, \ldots, R\}$.

As we can see, the problem with the exact analysis of an LRU cache's hit rate is the huge size of the state space S and the complexity of the model. In the next section, we describe an efficient way of obtaining an approximation for such a metric (i.e. cache hit or miss) that is shown to be very accurate by comparing its output to the results of simulations.

3.2 Markov chain-based approximation of an LRU cache

To reduce the complexity of modeling an LRU cache using Markov chains, the process is modeled by considering only one content at a time. Let us consider a Markov chain $X(r) = \big(X_h(r)\big)_{h \geq 0}$ with $N + 1$ states, as depicted in Figure 2.2. This chain will represent the evolution in time of the position occupied by content c_r in the cache, where state $N + 1$ means that content c_r is absent, state 1 means that the object is at the top of the cache and state N where it is at the bottom. Using this approximation, the size of the state space will then be $S = (N + 1) \times R$. The probability of a content staying at the same position upon the reception of an item is represented by $\gamma_s(r)$.

Assume we know $\gamma_i(r)$, $i = 1, 2, \ldots, N, N + 1$, satisfying $0 < \gamma_i(r) < 1$. This means

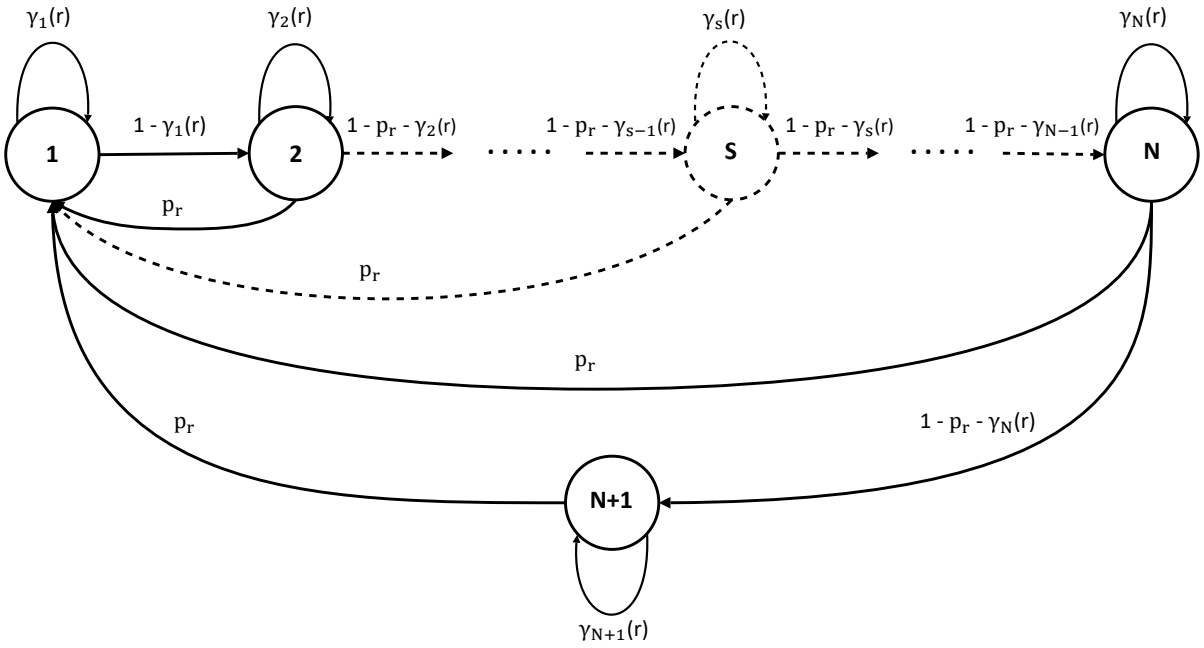


Figure 2.2 – Markov chain model for a content c_r in an LRU cache.

that $X(r)$ is irreducible and aperiodic. Let us denote by $\pi(r) = (\pi_1(r), \pi_2(r), \ldots, \pi_{N+1}(r))$ its equilibrium distribution. Assume now that $\pi(r)$ is exactly the marginal distribution corresponding to content c_r and that the chains $X(1), \ldots, X(R)$ are independent of each other. The probability that a content of rank r remains in the state s of the chain, with $(s \in \{1, 2, \ldots, N+1\})$, upon the arrival of a request is equal to

$$\begin{cases} \gamma_1(r) = p_r, \\ \gamma_s(r) = \sum_{i=1, i \neq r}^{R} p_i (\sum_{j=1}^{s-1} \pi_j(i)), \qquad 2 \leq s \leq N, \\ \gamma_{N+1}(r) = 1 - p_r. \end{cases} \tag{2.4}$$

Let's denote by $T_{i,j}$ the transition probability from state i to state j. Then, $T_{i,j}$ is equal to

$$\begin{cases} T_{i,i} = \gamma_i(r), \qquad 1 \leq i \leq N+1, \\ T_{i,1} = p_r, \qquad 2 \leq i \leq N+1, \\ T_{1,2} = 1 - \gamma_1(r), \\ T_{i,i+1} = 1 - p_r - \gamma_i(r), \qquad 2 \leq i \leq N, \\ T_{i,j} = 0, \qquad j \neq \{1, i, i+1\}. \end{cases} \tag{2.5}$$

The distribution $\pi(r)$ satisfies the Chapman-Kolmogorov equations:

$$\begin{cases} \pi_i(r) = \sum_{j=1}^{N+1} \pi_j(r) T_{j,i}, \qquad 1 \leq i \leq N+1, \\ \sum_{i=1}^{N+1} \pi_i(r) = 1. \end{cases} \tag{2.6}$$

If we develop the system of equations defined in (2.5) and (2.6), we obtain:

$$\begin{cases} \pi_1(r) = \gamma_1(r)\pi_1(r) + p_r(\pi_2(r) + \ldots + \pi_{N+1}(r)), \\ \pi_2(r) = \gamma_2(r)\pi_2(r) + (1 - \gamma_1(r))\pi_1(r), \\ \pi_3(r) = \gamma_3(r)\pi_3(r) + (1 - p_r - \gamma_2(r))\pi_2(r), \\ \ldots\ldots = \ldots\ldots \\ \ldots\ldots = \ldots\ldots \\ \pi_N(r) = \gamma_N(r)\pi_N(r) + (1 - p_r - \gamma_{N-1}(r))\pi_{N-1}(r), \\ \pi_{N+1} = \gamma_{N+1}(r)\pi_{N+1} + (1 - p_r - \gamma_N(r))\pi_N(r), \\ \pi_1(r) + \pi_2(r) + \ldots + \pi_{N+1}(r) = 1. \end{cases} \tag{2.7}$$

The equations derived in (2.7) can be easily solved using (2.4) to finally get:

$$\begin{cases} \pi_1(r) = p_r, \\ \pi_2(r) = \dfrac{p_r(1-p_r)}{1-\gamma_2(r)}, \\ \pi_i(r) = \dfrac{p_r(1-p_r)\prod_{j=2}^{i-1}(1-p_r-\gamma_j(r))}{\prod_{j=2}^{i}(1-\gamma_j(r))}, \qquad 3 \leq i \leq N+1, \\ \pi_1(r) + \pi_2(r) + \ldots + \pi_{N+1}(r) = 1. \end{cases} \tag{2.8}$$

The closed-form expression $\pi_{N+1}(r)$ represents the cache miss probability for a content c_r in a single cache. Its computational complexity is $O(N)$ where N represents the cache capacity (we suppose that the complexity of calculating each $\pi_i(r)$ is a constant). To get the cache hit rate of a content c_r, we simply do

$$P_{hit}(r) = \sum_{i=1}^{N} \pi_i(r) = 1 - \pi_{N+1}(r).$$

4 A general Markov chain model of a single LRU cache

4.1 A comprehensive analysis of generic LRU caches

Let's recall that in CCN, a node uses a caching scheme in order to decide whether an incoming item should be saved or not. In the general case, this decision can be seen as the probability that we denote $\beta(r)$, with which a received content c_r will be stored in a cache. The value of $\beta(r)$ will then depend on the caching strategy adopted by the node. When using the LCE policy, for example, all arriving objects are cached that is, $\beta(r) = 1$ for any content c_r. We will see later in more details the values taken by $\beta(r)$ when a specific caching strategy is used.

As we did in the previous section, let's consider a cache sized to contain $N \leq R$ items. Now, in the general case and whenever a local cache hit or a caching decision occurs for a content c_r, it will then be placed at the top position in the cache with probability $\beta(r)$. Consequently, for any given content $c_{r'}$ occupying position i in the cache, with $r' \neq r$, three actions are possible when a request for c_r is received:

— $c_{r'}$ will be moved down by one position if the requested content c_r is either outside the cache and it has been decided to cache it (with probability $\beta(r)$) or it occupies the block j with $j > i$;

— $c_{r'}$ will remain at the same position if c_r occupies the block j, with $j < i$, or if it is outside the cache and it has been decided to not cache it (with probability $1 - \beta(r)$);

— $c_{r'}$ will be evicted from the cache if it occupies the N^{th} position (last block) and it has been decided to cache c_r (with probability $\beta(r)$), which is outside the cache.

4.2 A generic model of a single LRU cache

Like the case of an LRU cache under the LCE scheme (i.e. $\beta(r) = 1$), we modeled this process using Markov chain $X(r) = \left(X_h(r)\right)_{h \geq 0}$ (Figure 2.3). The major differences are the transition from state $N+1$ to 1, which now includes the caching probability $\beta(r)$, and the probability of a content staying at the same state s (that we denote by $\gamma'_s(r)$). Assume we know $\gamma'_i(r)$, $i = 1, 2, \ldots, N, N+1$, satisfying $0 < \gamma'_i(r) < 1$. This means that $X(r)$ is irreducible and aperiodic. Let us denote by $\pi(r) = (\pi_1(r), \pi_2(r), \ldots, \pi_{N+1}(r))$ its equilibrium distribution. Assume now that $\pi(r)$ is exactly the marginal distribution corresponding to content c_r and that the chains $X(1), \ldots, X(R)$ are independent of each other. The probability that a content of rank r remains in state s of the chain upon

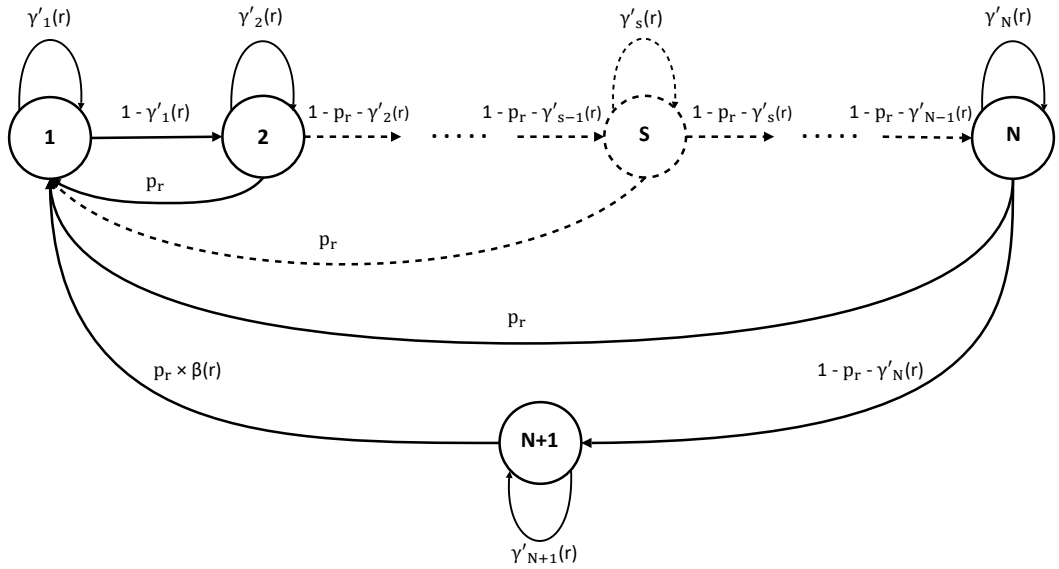


Figure 2.3 – General Markov chain model for a content c_r in an LRU cache.

the arrival of a request, with $s \in \{1, 2, \ldots, N+1\}$, is equal to

$$\begin{cases} \gamma'_s(r) = \gamma_s(r) + \sum_{i=1, i\neq r}^{R} (1-\beta(i)) p_i P_{miss}(i), 1 \leq s \leq N, \\ \gamma'_{N+1}(r) = 1 - p_r \beta(r). \end{cases} \tag{2.9}$$

The value of $P_{\text{miss}}(i)$ used in (2.9) represents the cache miss probability of content c_i (its calculation will be detailed later) and the value of $\gamma_s(r)$ is a special case of $\gamma'_s(r)$ where $\beta(i) = 1$. The first term of the expression of $\gamma'_s(r)$ (i.e. $\gamma_s(r)$) represents the case where a content c_r remains at the same position i when the received item occupies a position j, with $j < i$. The second term of $\gamma'_s(r)$ (i.e. $\sum_{i=1, i\neq r}^{R} (1-\beta(i)) p_i P_{miss}(i)$) corresponds to the event where the received content is not in the cache and it has been decided to discard it. Again, observe that (2.9) is an approximation, but as we will see, its form makes that it leads to a very accurate approximation algorithm. Let's denote by $T_{i,j}$ the transition probability from state i to j in our model. Then, $T_{i,j}$ is defined as follows:

$$\begin{cases} T_{i,i} = \gamma'_i(r), & 1 \leq i \leq N+1, \\ T_{i,1} = p_r, & 2 \leq i \leq N, \\ T_{N+1,1} = p_r \times \beta(r), & \\ T_{1,2} = 1 - \gamma'_1(r), & \\ T_{i,i+1} = 1 - p_r - \gamma'_i(r), & 2 \leq i \leq N, \\ T_{i,j} = 0, j \notin \{1, i, i+1\}. & \end{cases} \tag{2.10}$$

The distribution $\pi(r)$ satisfies the Chapman-Kolmogorov equations:

$$\begin{cases} \pi_i(r) = \sum_{j=1}^{N+1} \pi_j(r) T_{j,i}, & 1 \leq i \leq N+1, \\ \sum_{i=1}^{N+1} \pi_i(r) = 1. \end{cases} \tag{2.11}$$

If we develop the system of equations defined in (2.10) and (2.11), we obtain

$$\begin{cases} \pi_1(r) = \gamma'_1(r)\pi_1(r) + p_r \sum_{i=2}^{N} \pi_i(r) + p_r \beta(r) \pi_{N+1}(r), \\ \pi_2(r) = \gamma'_2(r)\pi_2(r) + (1 - \gamma'_1(r))\pi_1(r), \\ \pi_3(r) = \gamma'_3(r)\pi_3(r) + (1 - p_r - \gamma'_2(r))\pi_2(r), \\ \cdots\cdots = \cdots\cdots \\ \pi_N(r) = \gamma'_N(r)\pi_N(r) + (1 - p_r - \gamma'_{N-1}(r))\pi_{N-1}(r), \\ \pi_{N+1}(r) = \gamma'_{N+1}(r)\pi_{N+1}(r) + (1 - p_r - \gamma'_N(r))\pi_N(r), \\ \pi_1(r) + \pi_2(r) + \ldots + \pi_{N+1}(r) = 1. \end{cases} \tag{2.12}$$

By computing $\pi_{N+1}(r)$, which represents the cache miss probability, we can obtain the cache hit rate $1 - \pi_{N+1}(r)$. We can see from (2.12) that each $\pi_i(r)$ depends on $\pi_{i-1}(r)$ ($2 \leq i \leq N+1$), and once we get $\pi_1(r)$, we can calculate $\pi_{N+1}(r)$ and deduce the cache hit. When $\beta(r) = 1$, we have $\pi_1(r) = p_r$. The equations then derived in (2.12) can be easily solved using (2.4) to finally get

$$\begin{cases} \pi_1(r) = p_r, \\ \pi_2(r) = \dfrac{p_r(1-p_r)}{1-\gamma_2(r)}, \\ \pi_i(r) = \dfrac{p_r(1-p_r)\prod_{j=2}^{i-1}(1-p_r-\gamma_j(r))}{\prod_{j=2}^{i}(1-\gamma_j(r))}, & 3 \leq i \leq N+1, \\ \pi_1(r) + \pi_2(r) + \ldots + \pi_{N+1}(r) = 1. \end{cases} \tag{2.13}$$

In the general case ($0 \leq \beta(r) \leq 1$), $\pi_1(r)$ cannot be computed directly, and it depends on all the other values of $\pi_i(r)$ (for $2 \leq i \leq N+1$). One way to resolve this problem is to modify the expression of $\pi_1(r)$ in order to reduce its dependency on the other variables. To do so, we first add and subtract in the initial expression of $\pi_1(r)$ in

(2.12) the values $p_r\pi_1(r)$ and $p_r\pi_{N+1}(r)$. Then, using the expression of $\gamma_1'(r)$ and since $\sum_{i=1}^{N+1}\pi_i(r)=1$ and $\gamma_1(r)=p_r$, we get a new expression of $\pi_1(r)$ as follows:

$$\begin{aligned}\pi_1(r) &= \gamma_1'(r)\pi_1(r) + p_r\sum_{i=2}^{N}\pi_i(r) + p_r\beta(r)\pi_{N+1}(r),\\ &= \gamma_1'(r)\pi_1(r) + p_r\sum_{i=2}^{N}\pi_i(r) + p_r\beta(r)\pi_{N+1}(r) + p_r(\pi_{N+1}(r) - \pi_{N+1}(r) + \pi_1(r) - \pi_1(r)),\\ &= p_r(\sum_{i=1}^{N+1}\pi_i(r)) + (\gamma_1'(r) - \gamma_1(r))\pi_1(r) + p_r\beta(r)\pi_{N+1}(r) - p_r\pi_{N+1}(r),\\ &= p_r + (\gamma_1'(r) - \gamma_1(r))\pi_1(r) + p_r(\beta(r)-1)\pi_{N+1}(r).\end{aligned} \tag{2.14}$$

The expression of $\pi_1(r)$ can be rewritten as

$$\pi_1(r) = \frac{p_r(1 + (\beta(r)-1)\pi_{N+1}(r))}{1 - (\gamma_1'(r) - \gamma_1(r))}. \tag{2.15}$$

Now, $\pi_1(r)$ depends only on $\pi_{N+1}(r)$. The idea then is to use a fixed-point iteration in order to generate successive approximations of the solution, which consists on finding $\pi_{N+1}(r)$, by starting from an initial guess. To do so, we start by considering an approximate value of $\pi_{N+1}(r)$ (that we denote by $\pi'_{N+1}(r)$) by using the one obtained when $\beta(r)=1$, to compute $\pi_1(r)$. Once we have $\pi_1(r)$, we can calculate $\pi_i(r)$ ($2 \leq i \leq N+1$) to finally obtain a new value of $\pi_{N+1}(r)$.

$$\begin{cases} \pi_1(r) = \dfrac{p_r(1 + (\beta(r)-1)\pi'_{N+1}(r))}{1 - (\gamma_1'(r) - \gamma_1(r))},\\ \pi_2(r) = \gamma_2'(r)\pi_2(r) + (1 - \gamma_1'(r))\pi_1(r),\\ \pi_i(r) = \dfrac{(1 - p_r - \gamma'_{i-1}(r))\pi_{i-1}(r)}{1 - \gamma_i'(r)}, 3 \leq i \leq N+1,\\ \sum_{i=1}^{N+1}\pi_i(r) = 1. \end{cases} \tag{2.16}$$

The value of $\pi'_{N+1}(r)$ is also used to calculate $\gamma_s'(r)$ ($s \in 1,\ldots,N$) by replacing $P_{miss}(r)$ with $\pi'_{N+1}(r)$. These steps are repeated until we converge to the final solution by replacing in each step $\pi'_{N+1}(r)$ by the new computed value $\pi_{N+1}(r)$ (see Algorithm 1). As for the computational complexity of our model, let's recall first that the state space contains $N+1$ elements. The complexity of computing the cache hit of one content c_r is then $O(N)$ (we suppose here that the complexity of calculating each $\pi_i(r)$, $1 \leq i \leq N+1$,

Algorithm 1 Cache hit rate of a content with popularity r and caching decision $\beta(r)$

function GET_CACHE_HIT($r, \beta(r)$)
  $\pi'_{N+1}(r) \leftarrow$ *Get_cache_miss*$(r, 1)$ //Starting point of the iterative procedure ($\beta(r) = 1$)
  $\pi_{N+1}(r) \leftarrow$ *Get_cache_miss*$(r, \beta(r))$
  ϵ : Arbitrarily small positive quantity
  while $|\pi_{N+1}(r) - \pi'_{N+1}(r)| \geq \epsilon$ **do**
    $\pi'_{N+1}(r) \leftarrow \pi_{N+1}(r)$
    $\pi_{N+1}(r) \leftarrow$ *Get_cache_miss*$(r, \beta(r))$
  end while
  return $1 - \pi_{N+1}(r)$
end function

is a constant). To compute the total cache hit, we have to apply MACS to each content of the catalog. Since we have R available items in the catalog, then the complexity of computing the average hit of a single cache is $O(NR)$. Now, in case where we have a multi-cache system containing M nodes and if we want to compute the average hit ratio of all the caches, we obtain a complexity of $O(NRM)$.

4.3 Single cache model for 2Q

In [40], the authors proposed LRU-K, a page replacement algorithm for database disk buffering. The proposal is an enhancement of the classical LRU replacement policy in the sense that a history of requests is maintained for the elements of the cache. Indeed, LRU-K keeps track of the timing of the K last occurrences, which allows having a better idea of the contents' popularity. Thus, the element whose K^{th} most recent access was furthest in the past will be evicted when the cache is full. When K is equal to one (LRU-1), the approach is equivalent to the classical LRU. Note that, most of the LRU-K method gain is achieved when $K = 2$ (LRU-2) [40]. However, LRU-2 suffers from a high complexity, as each element access requires $\log(N)$ operations to manipulate a priority queue (N being the cache size).

Johnson et al. proposed the Two Queue scheme (2Q) [41], which is similar and performs as well as LRU-2 algorithm, but having a constant time overhead. Instead of cleaning cold elements from the main buffer like LRU-2, 2Q admits only the hot ones to the cache. When a request is received by a cache using 2Q, the requested object's hash is first placed in a virtual cache (called $A1$), which is managed as a FIFO. If an item is requested during its $A1$ residency, it is then promoted to the main cache

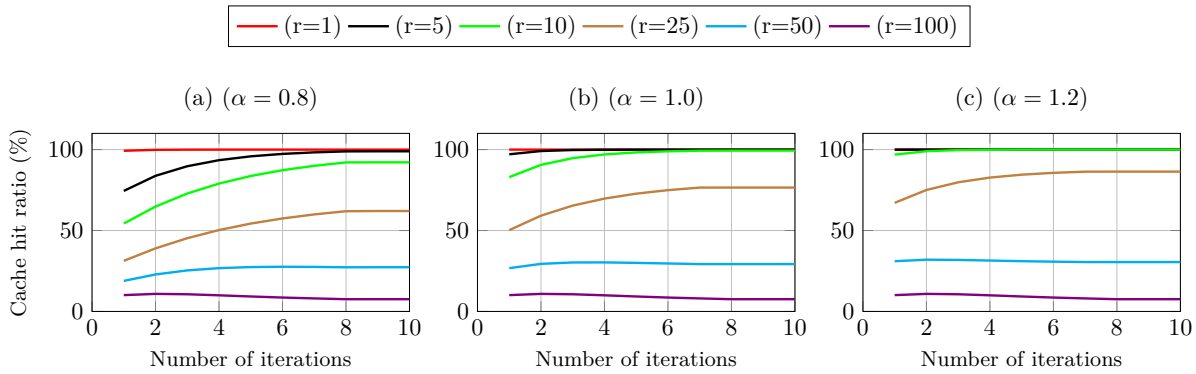


Figure 2.4 – Cache hit ratio of contents with various popularities vs iterations number of the 2Q fixed-point solution under a single node (catalog = 20000, cache size = 200).

(called Am). The authors, then, proposed another version of 2Q in which the $A1$ queue is partitioned into $A1in$ and $A1out$. The $A1in$ queue along with Am form the physical cache and $A1out$ is a virtual cache, which will contain only items hashes. The most recent first accesses will be stored in $A1in$, which will be managed as a FIFO queue. When objects are evicted from $A1in$, they will be remembered in $A1out$. Upon arrival of a request and if it is present in the $A1out$ queue, then it is cached in Am. The item to be discarded in 2Q is chosen either from $A1in$ or Am. However, the sizes of the different queues ($A1in$, $A1out$ and Am) are sensitive to the requests patterns and should be tuned carefully.

In our work, we considered a caching scheme similar to the first version of 2Q where the virtual buffer and the main cache are both managed using LRU. If we apply the model MACS presented previously to the 2Q scheme, $\beta(r)$ will then be equal to the hit probability in the virtual cache of the received content c_r. Since the virtual cache is managed as a classic LRU (i.e. each received item is always cached), $\beta(r)$ of a node v is obtained using equations (2.13):

$$\beta(r, v) = 1 - \pi_{N+1}(r, \mathit{VC}(v)). \tag{2.17}$$

The value of $\pi_{N+1}(r, \mathit{VC}(v))$ represents the miss probability of content c_r in the virtual cache of node v (i.e. $\mathit{VC}(v)$). We can see from Figure 2.4 that the solutions of our algorithm converge quickly in the different tested configurations. The convergence speed depends on the content popularity and the network settings, but in general, it takes about 8-10 iterations for all the contents to converge.

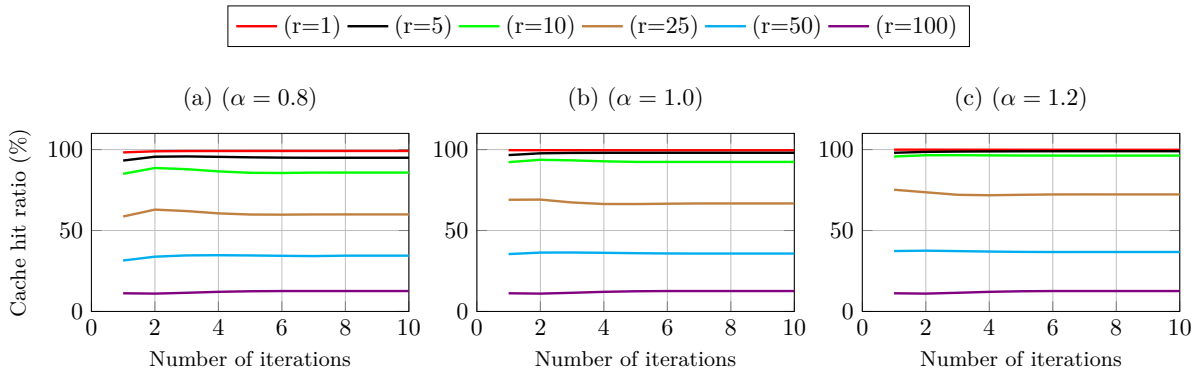


Figure 2.5 – Cache hit ratio of contents with various popularities vs iterations number of the LCD fixed-point solution under a single node (catalog = 20000, cache size = 200).

4.4 Single cache model for LCD

The authors in [37] proposed a caching scheme called Leave Copy down. Under the LCD scheme, a new copy of the requested object is cached only on the node that resides immediately below the location of the hit on the path to the requesting client. Compared to LCE, LCD moves the requested contents progressively from the origin server towards the clients, with each request advancing a new copy of the document one hop closer to the client. LCD aims to reduce the redundancy of the same items at multiple nodes by caching an object at one node at a time and to avoid the amplification of replacement errors. The conducted experiments in [37] had shown the efficiency and good performance of LCD under different configurations and in addition, it is an easy scheme to implement that does not need additional overhead.

If we model the LCD caching strategy using MACS, the value of $\beta(r)$ in this case will be the hit probability of content c_r in the next-hop cache. Using equations (2.16), the $\beta(r)$ of a node v is equal to:

$$\beta(r, v) = 1 - \pi_{N+1}(r, \mathit{NH}(v)). \tag{2.18}$$

The value of $\pi_{N+1}(r, \mathit{NH}(v))$ depicts the miss probability of c_r in the next-hop or parent cache of v (i.e. $\mathit{NH}(v)$). We can see from Figure 2.5 that the solutions of our algorithm converge quickly in the different tested configurations (it takes about 8-10 iterations for all the contents to converge).

5 Multiple caches systems

Following common practice [31]-[34], we assume in this work that after a cache miss and when a content is decided to be cached by a node, it will be downloaded instantaneously. Let's consider a system of multiple CCN nodes where the contents are forwarded according to the Shortest Path Routing (SPR) algorithm [59]. With SPR, when a client's interest cannot be satisfied by a node, it is forwarded along the shortest path to the closest permanent copy of the requested content. In this case, each node has to take into account, in addition to the local requests, the interests that come from other nodes due to a cache miss (we denote this stream of interests by "miss stream" or *MS*). The outgoing miss stream rate from a node u of a content c_r is equal to

$$\mathit{MS}_r(u) = \mathit{req}(r,u) \times \pi_{N+1}(r,u), \tag{2.19}$$

where $\mathit{req}(r,u)$ is the total proportion of requests for c_r received by u and $\pi_{N+1}(r,u)$ is the miss probability of content c_r at u. In CCNs, the interests for the same object received by a node will be aggregated and only the first one is sent to the next nodes. This feature should be considered when computing the total miss stream received by a node having more than one child node. The incoming miss stream ratio for an object with a rank r at a node v, that we denote by $\eta_r(v)$, is equal to

$$\eta_r(v) = \sum_{u:\mathit{NH}(u)=v} \left(\mathit{MS}_r(u) \prod_{w \neq u:\mathit{NH}(w)=v} (1 - \mathit{MS}_r(w)) \right). \tag{2.20}$$

The set $\{u : \mathit{NH}(u) = v\}$ represents the nodes having v as the next hop in the shortest path toward the source. The value $1 - \mathit{MS}_r(w)$ represents the case where an interest sent from the node w is discarded because it was already received by another node. If we consider a multi-cache network where the requests aggregation feature is not present, $\eta_r(v)$ in this case will be simply equal to

$$\eta_r(v) = \sum_{u:\mathit{NH}(u)=v} \mathit{MS}(r,u). \tag{2.21}$$

Now, when dealing with an interconnected network of caching nodes, the probability that a node v will receive a request for c_r will no longer be p_r, but another value that we denote as p'_r, which will take into account in addition to the local requests, the interests

due to a cache miss from previous nodes. For each node v, this value is equal to

$$p'_r = \frac{p_r + \eta_r(v)}{\sum_{k=1}^{R}(p_k + \eta_k(v))} = \frac{p_r + \eta_r(v)}{1 + \sum_{k=1}^{R}\eta_k(v)}. \tag{2.22}$$

In other words, p'_r represents here the proportion of requests received for c_r coming either from clients directly attached to the node (i.e. p_r) or from previous nodes (i.e. $\eta_r(v)$) over the total requests received by the node v for all the items. In the case where a CCN node does not have local requests, p'_r will be then equal to

$$p'_r = \frac{\eta_r(v)}{\sum_{k=1}^{R}\eta_k(v)}. \tag{2.23}$$

Consider again the Markov chain of a generic single LRU cache that we presented previously (see Figure 2.3). For every node v, we can compute the stationary state probabilities as we did in the case of a single node by replacing p_r with p'_r:

$$\begin{cases} \pi_1(r) = \dfrac{p'_r(1 + (\beta(r) - 1)\pi'_{N+1}(r))}{1 - (\gamma'_1(r) - \gamma_1(r))}, \\ \pi_2(r) = \gamma'_2(r)\pi_2(r) + (1 - \gamma'_1(r))\pi_1(r), \\ \pi_i(r) = \dfrac{(1 - p'_r - \gamma'_{i-1}(r))\pi_{i-1}(r)}{1 - \gamma'_i(r)}, 3 \leq i \leq N+1, \\ \sum_{i=1}^{N+1}\pi_i(r) = 1. \end{cases} \tag{2.24}$$

As we mentioned in the previous section, the cache hit probability of a content with popularity r is equal to $1 - \pi_{N+1}(r)$. To compute the cache hit performance of a multi-cache system operating under LCE or 2Q, we start by treating the leaf nodes of the network since in our model each node needs to know all the incoming stream of requests, including those received due to a cache miss from a previous node. Starting from the leaves where there is no miss stream requests, we go through the core nodes of the network until arriving at the source (or root) node where the permanent copies of the catalog's objects are attached.

When a multi-cache system is operating under the LCD strategy, we cannot produce the steady-state hit rates in a bottom up way, like we did with the previous schemes. This is due to the bidirectional dependency a cache's state has with its upstream or downstream node (and vice versa). When modeling LCD and to compute the cache

hit, each node in the network needs to have the incoming stream of requests received from previous nodes due to cache misses. At the same time, it is necessary for each node to know the cache hit rate of the upstream node in order to decide whether the object should be cached or not. To resolve these dependencies, we use a fixed-point iteration method. As a start, we compute the different incoming streams of all nodes assuming that the network operates under the LCE strategy. Since the root node represents the origin server containing all the available contents in the network, its cache hit probability is then equal to one for all the items. Therefore, we can in a top down manner compute the hit ratio of the different caches starting from the root node and going down to the rest of the network using the request streams obtained with LCE. Then, we can repeat these steps using, each time, the new obtained values of equilibrium hit probabilities to recalculate the request streams and then, deduce the different hit ratios until their convergence. The operations needed to compute the hit rates of the different nodes forming a network when the LCD scheme is used are summarized in the iterative procedure below:

1. Calculate the incoming stream of requests of all the nodes starting from the leaves of the network assuming that the network operates under the LCE strategy (i.e., $\beta(r) = 1, r \in \{1, \ldots, R\}$).
2. Compute the hit ratio of each node starting from the network's root where permanent copies of the available contents are attached using the LCD scheme.
3. Recalculate the incoming stream of requests of the network nodes using the new obtained cache hit rates.
4. Update the network's hit rate using the last obtained values of the incoming stream of requests received by each node.
5. Repeat step three and four until convergence of the cache hit values.

6 Model Evaluation

6.1 Tests environment

In order to evaluate the accuracy of our proposals, we compared the analytical models presented, in the previous section, with the results of simulations under *ccnSim* [60], which is a discrete-event and a chunk-level simulator for CCN networks. The accuracy

of MACS, compared to the simulation results, can be affected by many parameters. Our focus on the conducted experiments was on the following key settings: cache size and Zipf law's skew distribution value. As for the network topology, a complete binary tree of 31 nodes was chosen to validate our model where the distance and the latency between each two adjacent nodes are the same [1].

We measured during the tests three metrics: cache hit rate, content provider load and distance reduction ratio. The cache hit rate metric represents the ratio of requests that were served by the caches over the total number of requests sent in the network. The content providers load is defined as the proportion of requests not served by the intermediate caches of the network and thus, retrieved from the main source of contents. The distance reduction ratio represents the average gain obtained in terms of distance that an interest travels before finding a copy of the requested object. The content provider load and the distance reduction ratio (denoted respectively as *CPL* and *DRR*) are both expressed as a percentage and are computed as follows:

$$\textit{CPL} = \prod_{i=1}^{s} P_{miss}(i) \times 100,$$

$$\textit{DRR} = \frac{Dist(s) - \sum_{i=1}^{s} \left(\left(\prod_{j=1}^{i-1} P_{miss}(j) \right) \times P_{hit}(i) \times Dist(i) \right)}{Dist(s)} \times 100.$$

In the definition of CPL and DRR and for sake of clarity, we supposed that the network is formed by a line of s nodes numbered from 1 to s where the clients are attached to node 1 and the content repository is located just after node s. It should be noted that of course, we can compute these metrics in any type of network topology. The values $P_{miss}(i)$ and $P_{hit}(i)$ used here represent respectively the cache hit and cache miss of a network's node v_i. The expression $Dist(i)$ is the distance from where the clients requests were generated to the node v_i (we could also consider the delay between nodes instead of distance and thus calculate the average network delay.). The index s represents the nearest node v_s to the clients to which a content provider is attached (i.e. where a permanent copy of the accessible contents is available).

In the simulation settings, we considered a catalog of contents containing $20,000$ 1-chunk sized objects whose popularity distribution follows the Zipf's law. Permanent copies of the available contents are hosted on one repository attached to the root node of the network. We set a uniform cache store capacity on the CCN nodes, which was

1. Realistic network topologies can be also used as we did in [18].

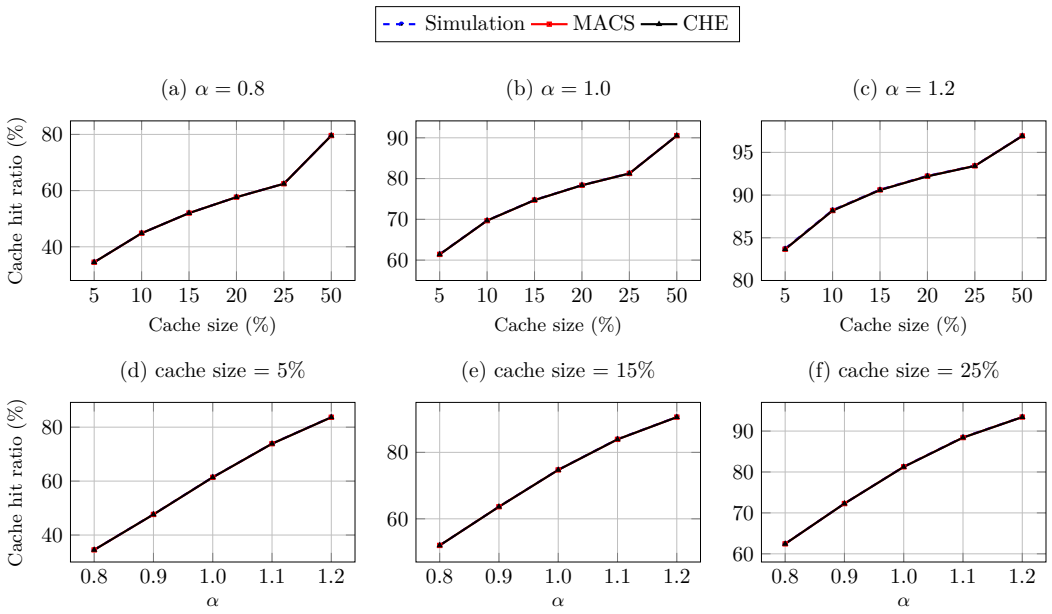


Figure 2.6 – Comparison of MACS and Che approximation in the case of a single cache.

defined as a proportion of the catalog size. Different simulations were conducted with a cache store size varying from 0.1% to 1.0% of the catalog capacity. The clients, attached to the network's leaves, generate requests according to a Poisson process with a rate per client corresponding to one request per second (each client representing an aggregate of users). We tested also different values of the Zipf law's skew parameter α going from 0.8 to 1.2. As shown in many studies [55], these two values correspond to the Zipf popularity exponent in the case of User Generated Content (UGC) and Video on Demand (VoD), respectively. The Least Recently Used (LRU) scheme is used as a cache replacement policy and the virtual buffer capacity used in the 2Q algorithm is equal to the main cache size, to see the model accuracy with different values of the virtual cache size. Next, we will expose and compare the cache hit results obtained with our analytical model and with the ccnSim simulation tool. The simulation results of ccnSim, shown in the graphs, depict the mean values taken over 30 runs, where 10^6 requests are sent in the network after it reaches its stability (i.e., all the caches become full). The error bars represent 99% confidence intervals.

6.2 Model results and analysis

Before we proceed further and go through the different results obtained with our model in the case of multi-cache networks, we evaluate in Figure 2.6 the accuracy of

Table 2.2 – Hit ratio accuracy comparison (in %) between different models of a 2Q single cache under the IRM model with a Zipf's distribution parameter $\alpha = 0.8$. Simulations are based on 10 runs of $10^3 \times R$ requests with a warm-up period of 33 %.

Cache/Catalog	MACS (e1/e2)	Eq.(13) of [39] (e1/e2)	Eq.(10) of [38] (e1/e2)
100/1000	47.533 (1.312/0.161)	47.641 (1.543/0.064)	47.808 (1.899/0.415)
100/10000	27.215 (0.665/0.391)	27.352 (1.172/0.109)	27.404 (1.364/0.302)
1000/10000	52.720 (1.497/0.248)	52.596 (1.288/0.013)	52.746 (1.577/0.300)

MACS and CHE in terms of hit ratio of a single cache by comparing their outcomes to those obtained with the simulation tool ccnSim where the LCE scheme is used. The different plots exposed in Figure 2.6 represent the hit ratio as a function of the cache size (Figures 2.6(a)-2.6(b)-2.6(c)) and Zipf's parameter α (Figures 2.6(d)-2.6(e)-2.6(f)). We can see from the graphs how MACS has the same precision as the Che approximation and that both models are very accurate. As mentioned in Chapter 1, many models from the state-of-the-art proposed 2Q models based on the approximation of Che [38]-[39]. A quick comparison of these proposals with MACS is shown in Table 2.2, where e1 and e2 represent respectively the percent errors in terms of hit ratio relatively to ccnSim and the simulator used in [39] and the virtual cache has the same size as the cache itself. The results in Table 2.2 show a good accuracy of the different tested models without a clear difference between them. Compared to MACS, the Che approximation has a lower complexity, but as we discussed earlier in the previous sections and as we will see in the next results, our proposal is a more general model that can be used not only with a specific caching scheme, but covers various caching strategies.

Figures 2.7-2.8-2.9 compare LCE, 2Q and LCD models (i.e. MACS), respectively, with the simulations conducted under different scenarios within ccnSim, in terms of average cache hit ratio as a function of content popularity. For the sake of clarity, we considered in the graphs only the objects whose popularity goes from 1 to 500. We can see from the charts that our analytical model give in average an accurate hit rate for the whole range of item population with different network settings. When the cache size is set to a low value (0.1% of the catalog), the model performs better even with distinct values of α and for different types of content popularity (see Figures 2.7(a)-2.7(b)-2.7(c)-2.8(a)-2.8(b)-2.8(c)-2.9(a)-2.9(b)-2.9(c)). A slight accuracy reduction in the cache hit ratio per content is observed when the cache capacity is set to 1% of the catalog size

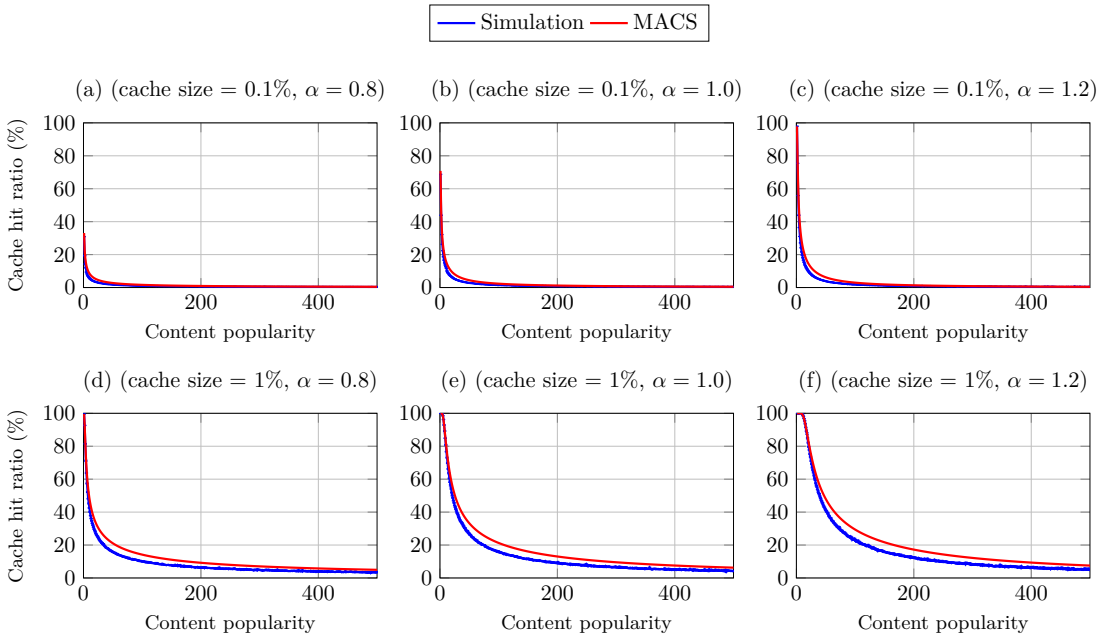


Figure 2.7 – Average hit rate vs content popularity with different parameters and using the LCE scheme.

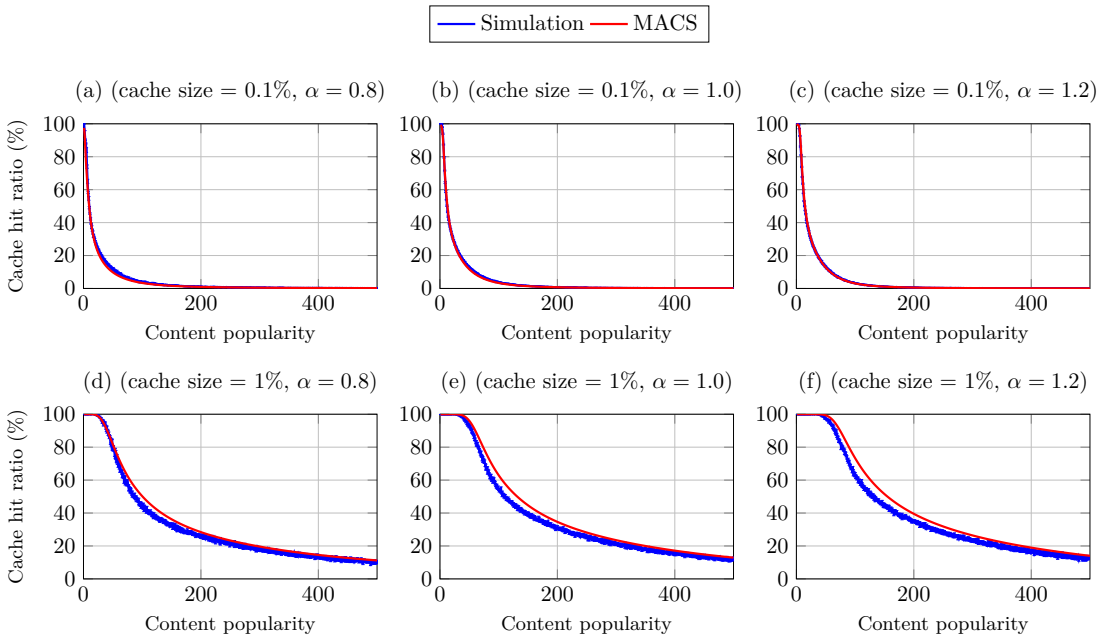


Figure 2.8 – Average hit rate vs content popularity with different parameters and using the 2Q scheme.

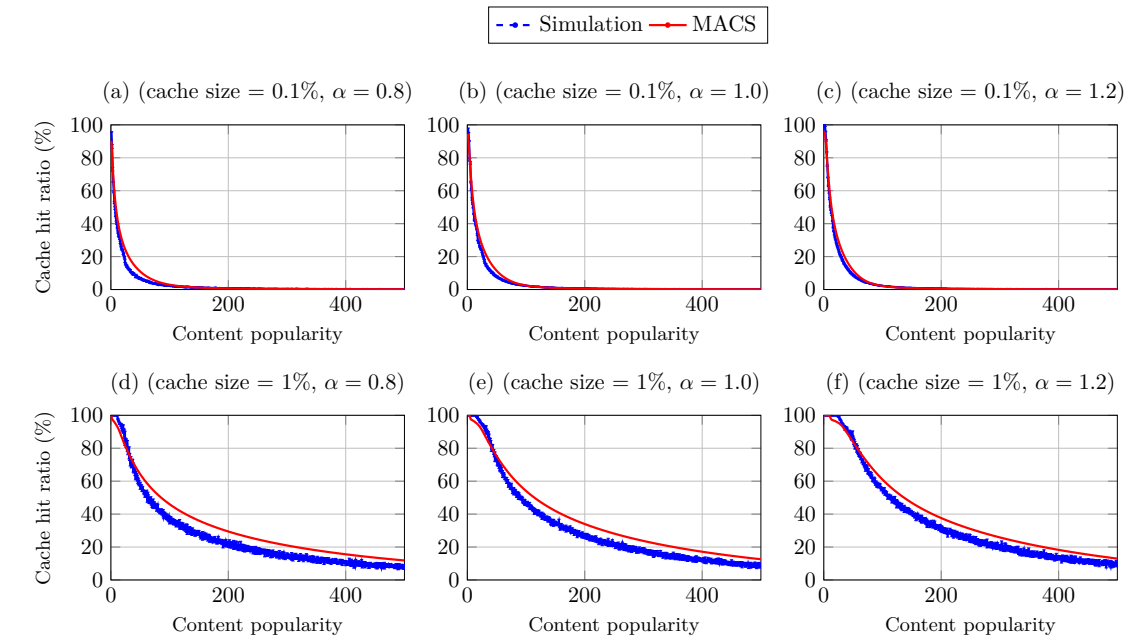


Figure 2.9 – Average hit rate vs content popularity with different parameters and using the LCD scheme.

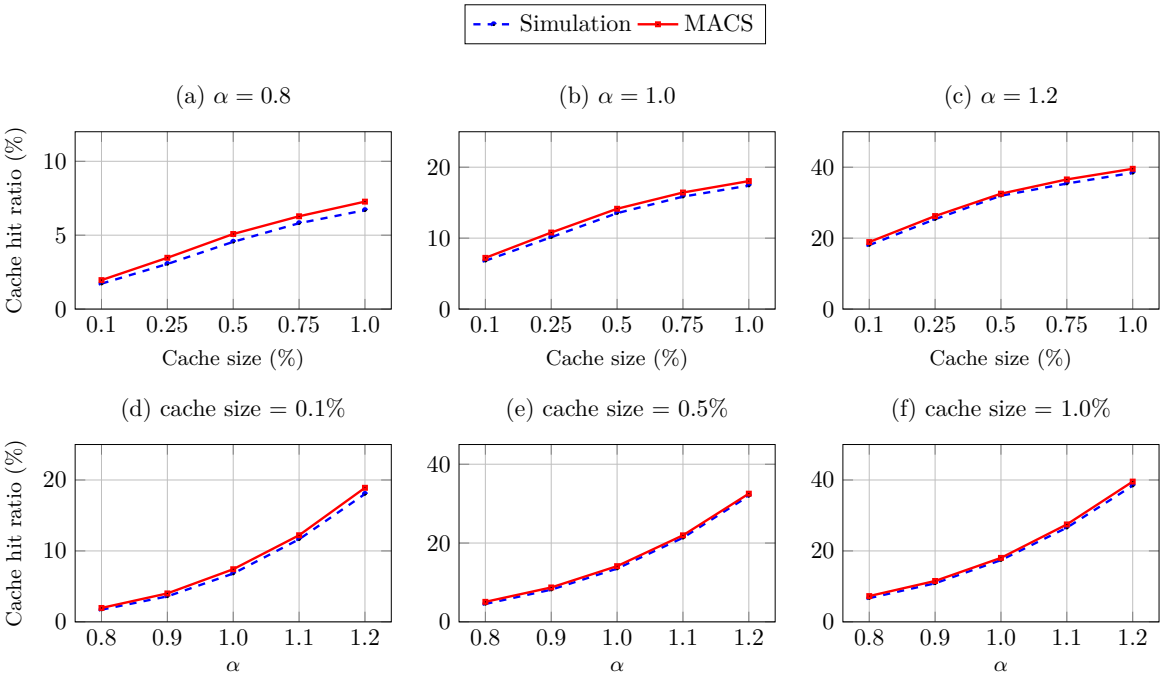


Figure 2.10 – Total hit rate of the network vs different configurations using the LCE scheme.

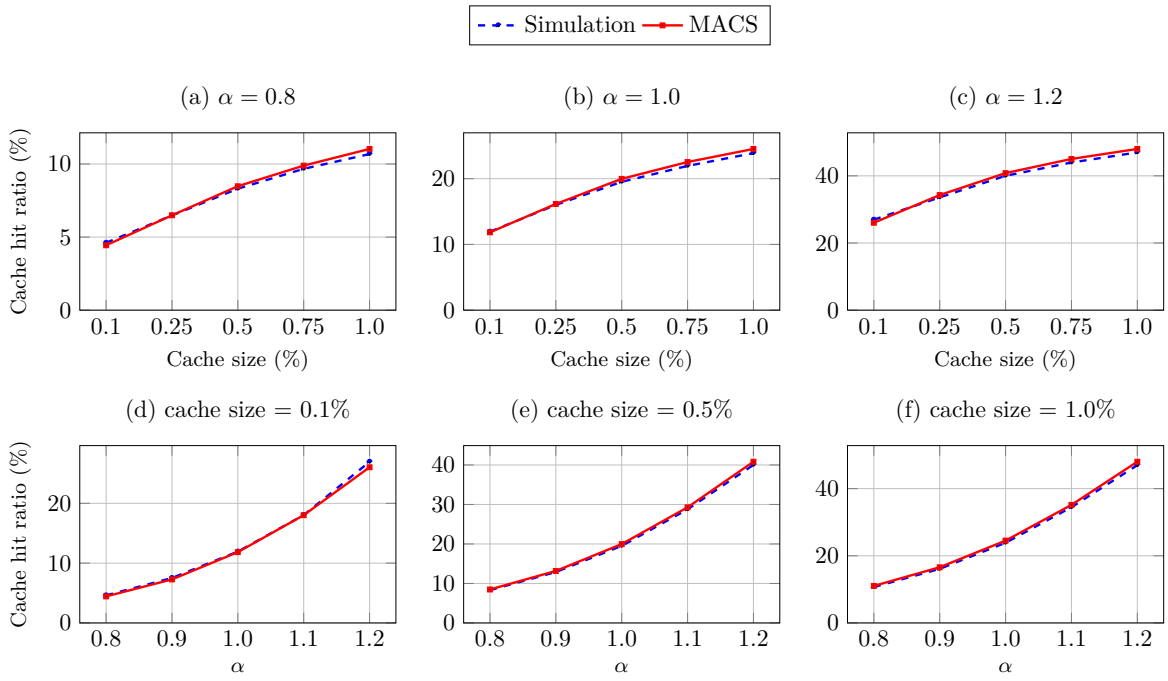


Figure 2.11 – Total hit rate of the network vs different configurations using the 2Q scheme.

(see Figures 2.7(d)-2.7(e)-2.7(f)-2.8(d)-2.8(e)-2.8(f)-2.9(d)-2.9(e)-2.9(f)). An increase of the cache size means more states to be considered in MACS for each content, which makes it harder to estimate the hit ratio of each item in the whole network.

In Figures 2.10-2.11-2.12, we plot the network's total hit rate as a function of the cache size and the Zipf's law skew parameter α of LCE, 2Q and LCD models along with ccnSim. Our aim here is to try a large range of values in the network configurations and see the model's performance in terms of accuracy. The results from the charts show a good accuracy in estimating the overall cache hit performance of the network with various settings. Compared to 2Q and LCE, the error rate in LCD model is a little higher. Let's recall that the complexity of modeling LCD with MACS is higher, as it requires the addition of a fixed-point iteration method to compute the cache hit performance of a multi-cache system, as explained previously, which may increase the error rate.

Figures 2.13-2.14-2.15 display the cache hit accuracy at different levels of the network topology using LCE, 2Q and LCD. Level one represents the leaf caches and level five being the root node. The models accuracy is slightly reduced at higher levels within the network, especially with the LCD model. The main cause of this inaccuracy is related to the estimation of the request streams due to cache miss. Indeed, at the leaf

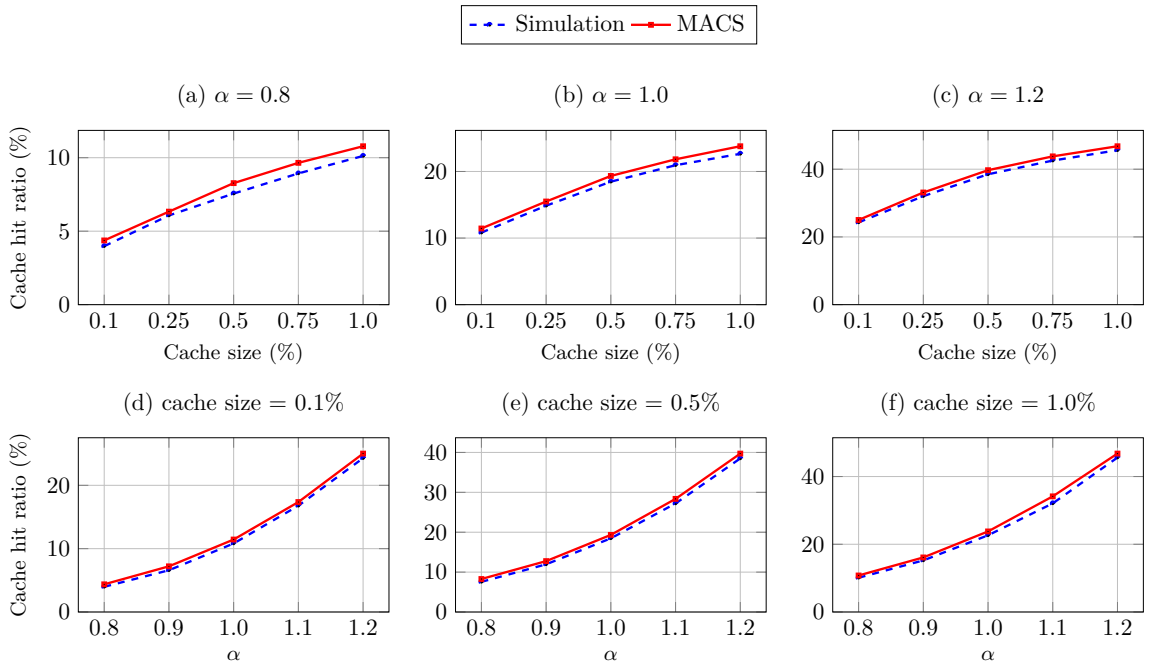


Figure 2.12 – Total hit rate of the network vs different configurations using the LCD scheme.

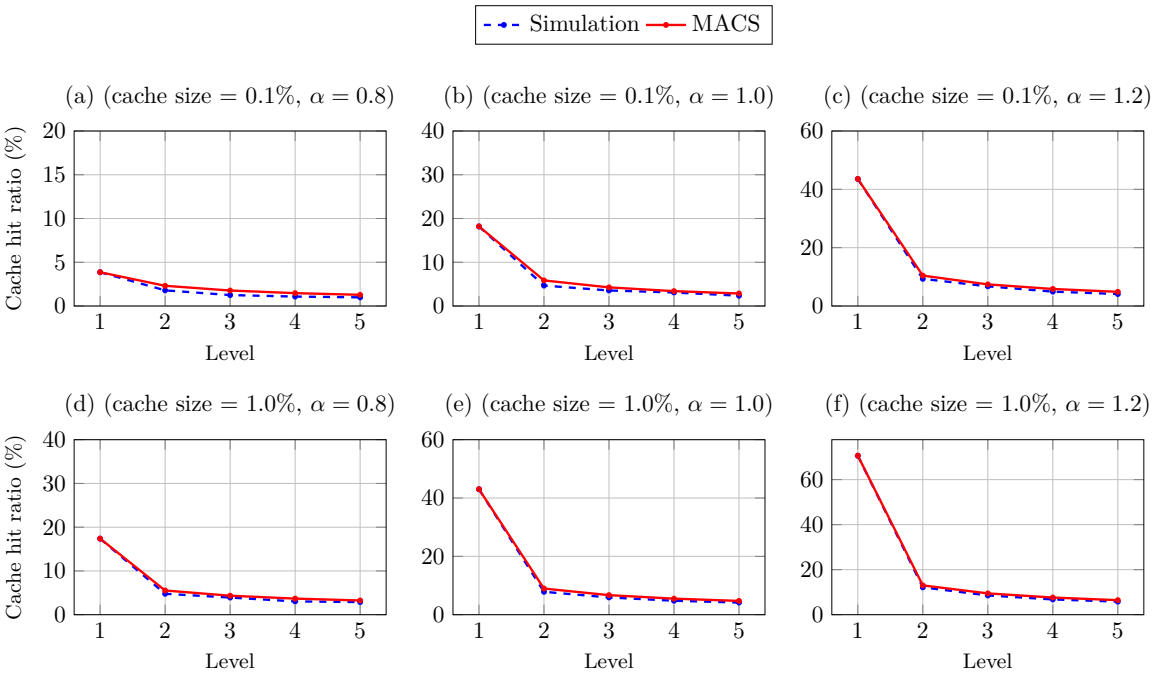


Figure 2.13 – Cache hit ratio at different layers of the network using the LCE scheme.

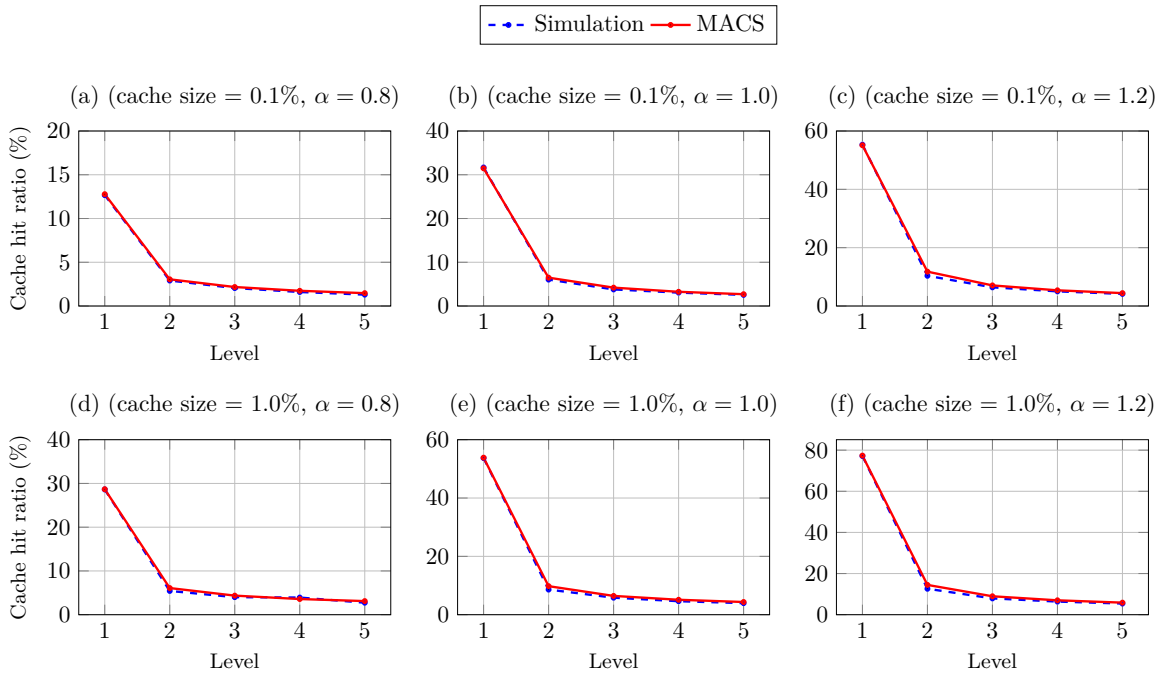


Figure 2.14 – Cache hit ratio at different layers of the network using the 2Q scheme.

nodes, the requests received contain only those generated by the clients, which can be easily estimated. However, the incoming miss streams of the nodes located at the core of the network are more difficult to estimate since they are computed using the cache hit rate of previous nodes.

6.3 Caching algorithms comparison

We compare, in the following, the performance of the different caching strategies that we have modeled and analyzed previously: LCE, LCD, 2Q and Opt (only analytical results are shown). Opt represents the maximum cache hit ratio that we can achieve when the objective is to reduce the distance travelled to retrieve contents. More specifically, if we consider a network where the caches have the same size N (in terms of number of items that can be stored) and the contents popularities are known in advance, then Opt consists on caching in the nodes located at 1-hop from the clients the most N popular contents of the catalog $(c_1, \ldots, c_N)$. Then, the second N most popular items $(c_{N+1}, \ldots, c_{2N})$ will be cached at the 2-hop nodes, etc. The results of Opt shown in the graphs were obtained using MACS by fixing to one the value of $\beta(r)$ for the contents that should be cached at each node and $\beta(r)$ is set to zero for the other items.

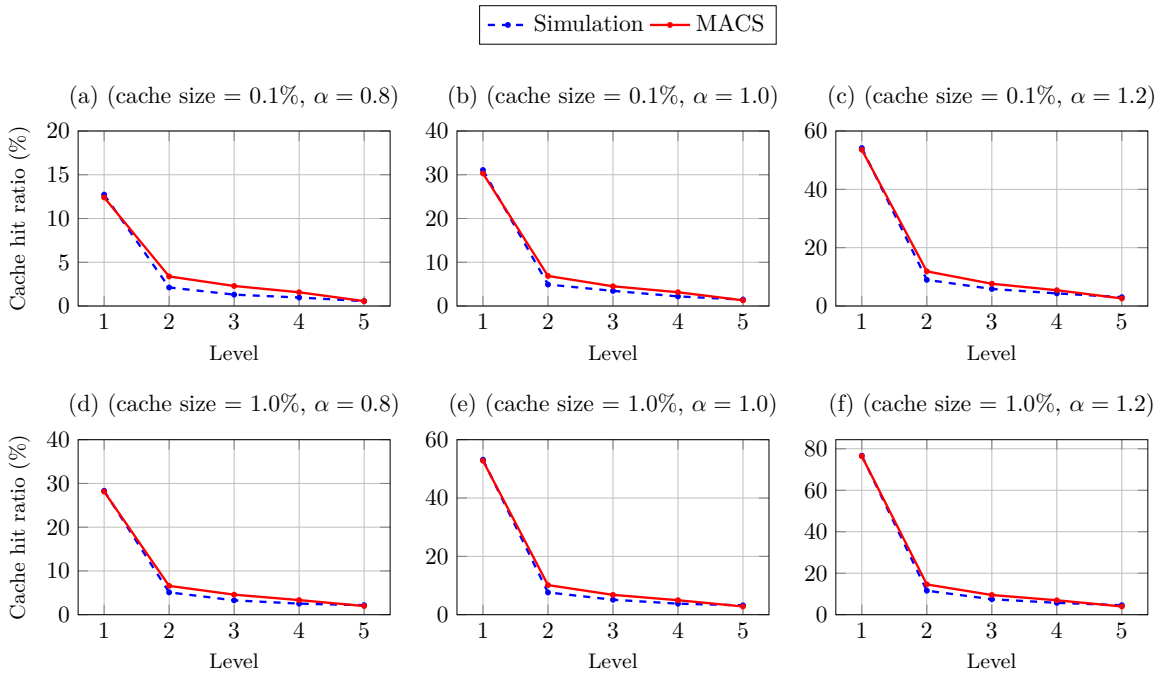


Figure 2.15 – Cache hit ratio at different layers of the network using the LCD scheme.

The cache hit metric is an indicator of caching efficiency in multi-cache networks and generally, as the cache hit of each node increases, the network performance becomes better. Particularly, in traditional network topologies, we have many access caches and fewer core caches (a tree topology is a good example of this type of network). In this type of topology, access caches are linked to clients and often have similar content in the same region and the total cache hit is calculated as the average hit of the different caches in the network. As there is more caches at the access layer level, improving the overall cache hit would first require improving the hit of the access caches. This approach seems to be appropriate since it consists on bringing content closer to users and thus improving their quality of experience. This also has the advantage of reducing traffic in the operator's network. This strategy represents the main objective behind LCD and 2Q. For this reason, and when we made the comparison in terms of cache hit ratio, the target of Opt was to maximize the hop saving in order to cope with the aim of the caching schemes used in the state-of-the-art.

In Figure 2.16, we display the total hit rate of the network as a function of the α value (the Zipf law skew parameter) and the cache size using different caching schemes. The virtual cache size in 2Q was set to 20, which represents approximately the best tuning

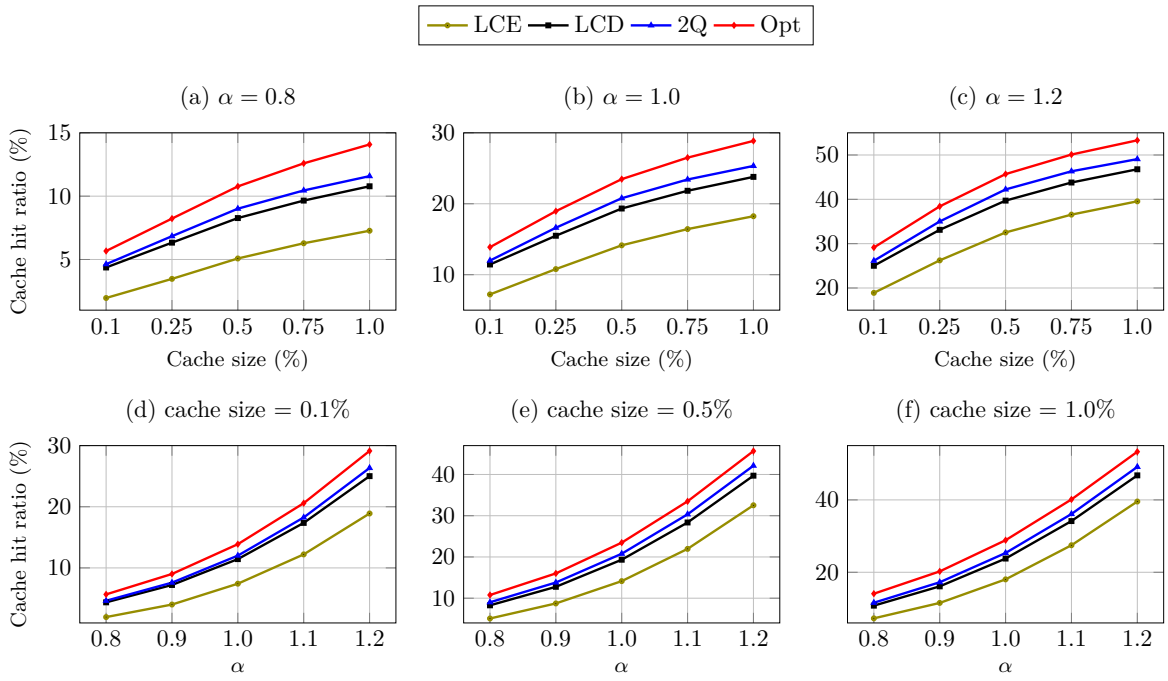


Figure 2.16 – Total hit rate of the network with different parameters and various caching strategies.

in the different use cases that were evaluated. We can see from the charts that LCD outperforms LCE, as it reduces the content redundancy in the network nodes and thus, more distinct objects are available in the caches. Each time a content is requested in LCD, it gets one-hop closer to the leaf nodes, which is an efficient way to detect and keep the popular items as closest as possible to the clients and to let the unpopular ones on higher levels of the network. Despite the efficiency of LCD, the results show the superiority of 2Q compared to the other caching schemes. Thanks to an effective filtering effect by means of a virtual buffer, 2Q admits more popular contents into the cache than the other techniques.

As the cache size increases (Figures 2.16(a)-2.16(b)-2.16(c)), the performance difference between the evaluated caching schemes is reduced and gets closer to the optimal values. Increasing the storage capacity will diminish the impact of caching unpopular items, and the cache will become less affected by the adopted caching scheme. However, a limited cache size will increase the probability of discarding valuable contents to the benefit of the unpopular ones. In this case, the efficiency of the decision caching strategy in accepting only the most popular items is crucial. For example, when α is set to 1.0 and the cache size is equal to 0.1% of the catalog, LCE achieves 53% of

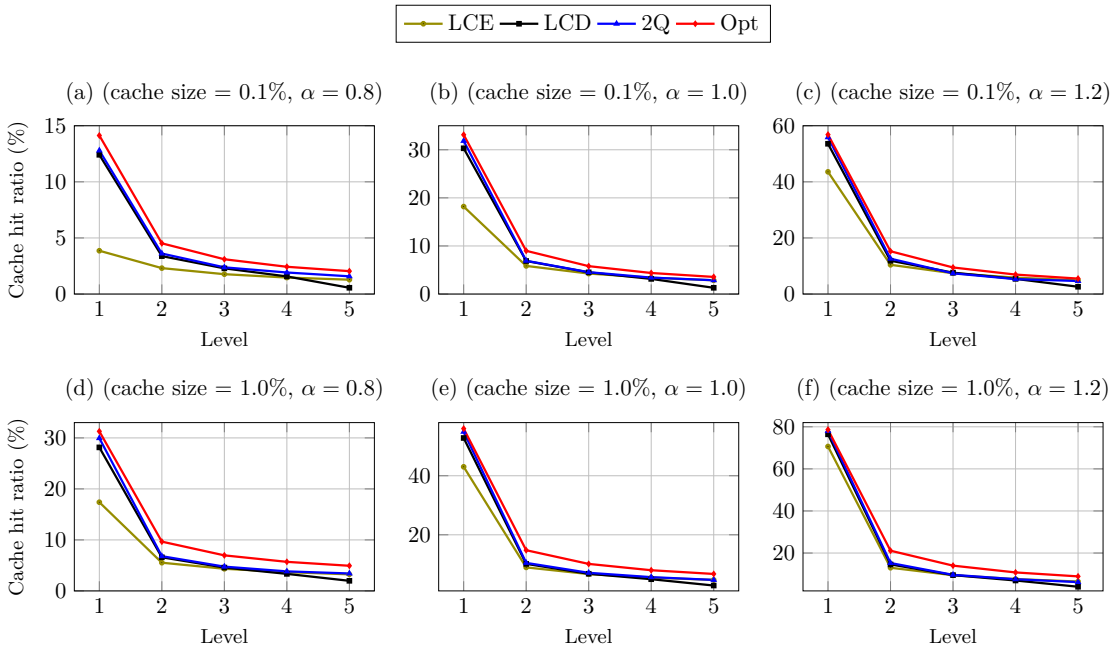


Figure 2.17 – Total hit rate at different layers of the network with different parameters and various caching strategies.

the optimal performance. However, if we set the cache size value to 1.0% of the catalog and we keep the same value of α, LCE performance reaches 63% of the optimal one.

The results also show a clear impact of the contents popularity distribution on the caching efficiency (Figures 2.16(d)-2.16(e)-2.16(f)). A high value of α (above one) means that fewer contents will receive most of the requests and, thus, decreasing the probability of caching many distinct unpopular items independently from the caching strategy. However, when α is set to a low value (below one), the contents popularity becomes more uniform, which makes it harder for the caching scheme to anticipate the most demanded items. For example, when the cache size is set 0.5% of the catalog and α is equal to 0.8, LCE achieves 47% of the optimal performance. In case where the α is set to 1.2 and without changing the cache size value, LCE performance reaches $\%71$ of the optimal one.

Figure 2.17 displays the cache hit ratio at different layers of the networks with different settings and using different caching schemes. We can see from the different graphs that the cache hit performance is much higher at the first level than the other ones, independently from the used caching scheme. The nodes at level one will be the first to receive the clients' requests, which increases their probability to serve the most

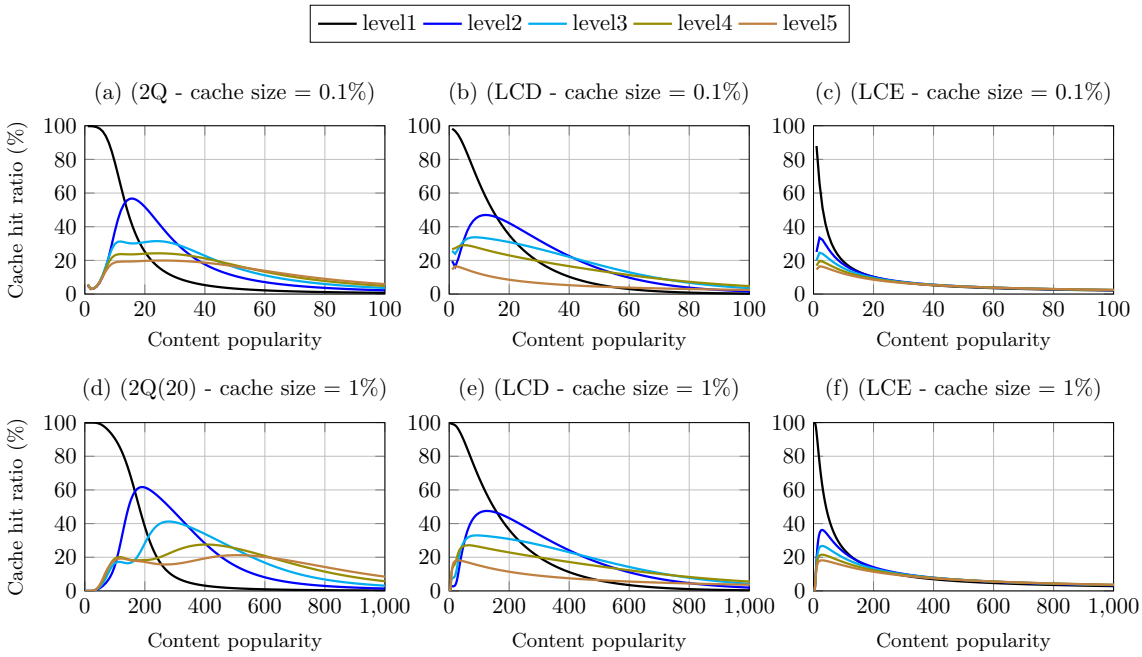


Figure 2.18 – Cache hit ratio vs content popularity at different layers of the network with different parameters and various caching strategies.

popular contents demanded by the users. This will result in nodes at higher levels to cache less popular contents and thus, decreasing their performance. Compared to the total cache hit ratio of the network, it is clear that the first level nodes achieve most of it. The storage capacities located at higher levels of the network cannot be used efficiently and can be considered as wasted resources. As we have highlighted previously during the analysis of the results depicted in Figure 2.16, the cache size and the Zipf law parameter α values have a significant impact on the caching efficiency, as it can be seen from the performance of the nodes at the first level of the network (Figure 2.17). As the cache size and α increase, caching the most popular objects and keeping them in the cache becomes easier, which will result in low performance difference between the considered caching schemes.

Figure 2.18 gives an idea about the contents getting the most hits at each level of the network, for different parameters and caching schemes. For the sake of clarity, the objects having a very low cache hit rate are not shown on the graphs. These results clearly confirm the superiority of LCD and especially, 2Q over the other ones. Indeed, the graphs show clearly that 2Q succeeds in keeping at the nodes of level one more popular items than the other techniques. In LCD or LCE, the top most requested objects

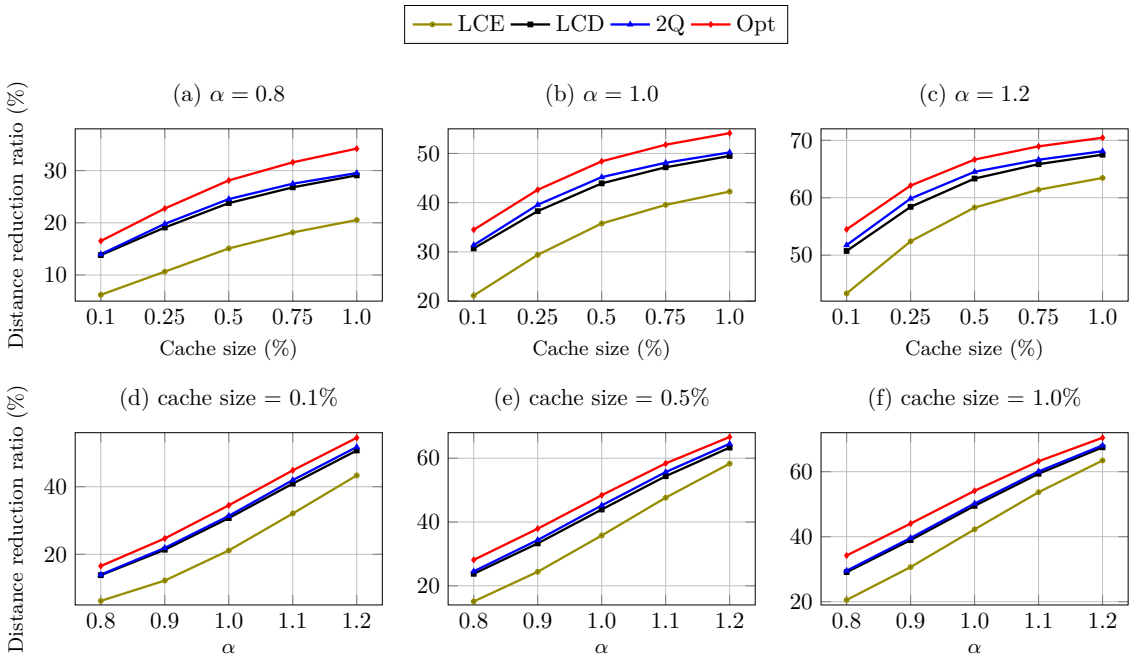


Figure 2.19 – Average distance reduction ratio of the network with different parameters and various caching strategies.

are not served exclusively by the first level nodes, which will increase the average distance to get contents. We can also see from the graphs how the popularity of cached contents and their hit ratio decrease as we go up on higher levels of the network. This can be explained by the fact that caching contents that already exists on previous nodes makes them useless, which increases the replacement errors of items in the cache and, thus, decreases the nodes performance.

In Figures 2.19-2.20, we display the distance reduction ratio and the content provider's load using different caching schemes. The results of Opt presented in Figure 2.19 are obtained by setting as its objective the reduction of distance to retrieve contents, as we explained it previously. In Figure 2.20, Opt was configured to minimize the usage of the content provider, which is achieved by caching the most popular items as near as possible to it. In other words, Opt, in this case, consists on caching in the nodes located at 1-hop from the content provider the most N popular contents of the catalog $(c_1, \ldots, c_N)$. Then, the second N most popular items $(c_{N+1}, \ldots, c_{2N})$ will be cached at the 2-hop nodes, etc. The graphs in Figures 2.19-2.20 confirm what we have obtained in the previous results. The performance of 2Q and LCD are very close, but 2Q remains always slightly better than LCD. Besides, the results indicate

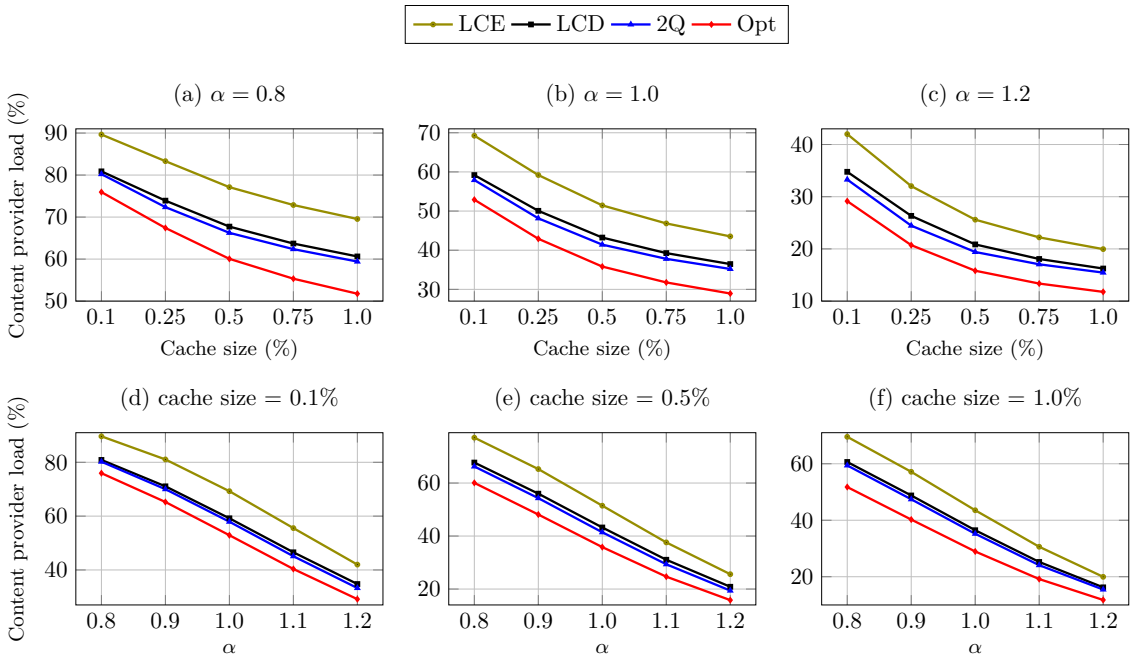


Figure 2.20 – Content provider load of the network with different parameters and various caching strategies.

that increasing the cache size improves, of course, the network performance, but not in a uniform manner. In fact, doubling the cache size does not necessarily double the caching scheme efficiency. Now, if we compare the results of LCD and especially, 2Q to Opt (see Figures 2.16-2.17-2.19-2.20), we can see that there is no much room left for improvement. In addition, a caching scheme that can reach or get closer to the theoretical optimal outcomes, in terms of network performance (i.e., cache hit ratio, content provider load, etc.), will necessarily increase the network operations overhead. The efficiency of the 2Q algorithm and its performance/overhead trade-off makes it a good candidate to be used as a caching strategy in CCN networks.

7 Conclusion

The Content-Centric Networking (CCN) paradigm is one of the most promising architectures for the future Internet. The in-network caching feature provided by CCN has a direct impact on the system performance, and it is important to analyze and evaluate the caching behavior in order to gain insights for optimized CCN caching schemes. We

presented in this chapter MACS, a Markov chain-based Approximation of Caching Systems to estimate the cache hit probability under the popular LRU replacement policy. The model can be applied to different caching systems, not only to CCN, and can be used to compute different performance metrics in addition to the cache hit, such as the content provider load and the distance reduction ratio. The versatility of MACS enables us to model and analyze different caching schemes like LCE, LCD and 2Q. The conducted experiments clearly show the accuracy of our model in estimating the cache hit rate of a multi-cache system and also indicate the efficiency of 2Q in terms of network performance, which makes it a good candidate to be used as a caching strategy.

In the next chapter, we will investigate the impact of cache resources placement in CCN and in multi-cache systems in general. By means of our analytic tool MACS, a new approach is proposed to tackle this problem, and we will see how it can improve significantly the network performance.

CHAPTER 3

EFFICIENTLY ALLOCATING DISTRIBUTED CACHING RESOURCES IN FUTURE NETWORKS

Contents

1 Motivations 94

2 Multi-objective cache placement strategy 95

2.1 System assumptions 95

2.2 Problem formulation 96

2.3 Solving cache allocation problem using GRASP 98

2.3.1 Mono-objective GRASP 98

2.3.2 Multi-objective GRASP 101

3 Performance evaluation 104

3.1 Model configuration 104

3.2 Model results and analysis 105

4 Conclusion 111

Introduction

After presenting our model MACS in the previous chapter, we investigate in this chapter the optimal placement of caching resources. When studying multi-cache networks, the cache allocation problem is one of the most important issues to address, especially due to the very expensive cost of deploying distributed storage capacities along the network. More specifically, it consists on finding how a limited storage budget should be distributed across the network's nodes in order to ensure an effective use of caches. The proposed approach in this chapter [21], which is based on joint cache management between ISPs and CPs, solves the trade-off between minimizing the hit rate in the origin server (with the risk of degrading the quality of experience) and minimizing the distance between clients and the requested contents (with the risk of ineffective caching for the same amount of cache). We model the cache allocation problem as a multi-objective integer nonlinear program (the nonlinearity is confined to the objective functions). Since it is an NP-hard problem [61], we propose the use of the Greedy Randomized Adaptive Search Procedure (GRASP) [62], which has shown its effectiveness in solving combinatorial optimization problems. Moreover, unlike other metaheuristics like evolutionary algorithms, GRASP requires fewer parameters (only one parameter in its basic version).

1 Motivations

During the last decade, ISPs and CPs infrastructures underwent a major metamorphosis driven by new networking paradigms, namely: Software Defined Networks (SDN) [63] and Network Function Virtualization (NFV) [64]. The upcoming advent of 5G will certainly represent the most important achievement of this evolution [65]. In this context, static (planning) or dynamic (on-demand) network resources placement, especially caching, remains an open issue. Indeed, co-locating caching resources at the core of the network optimizes caches, but not the network. Distributing caches optimizes network resources, but reduces the efficiency of caches, due to the existing redundancy at the edge. The optimal placement of caching resources is one of the most important issues to address in multi-cache networks in general, especially due to the very expensive cost of deploying distributed storage capacities along the network.

In previous works that addressed the cache allocation problem, only one perfor-

mance gain metric is generally considered (e.g. access latency, cache hit, etc.) to find a storage distribution solution or to compare between different possibilities. In this work, we use jointly the following two performance metrics to evaluate the cache gain: content provider load and average distance ratio. The first metric represents the amount of contents that were served by the origin server over all the requests sent in the network. The second one depicts the average distance travelled to get contents in the network over the obtained distance without caching. We chose these two metrics for their importance in representing the cost and gain obtained from the use of the in-network caching. A high content provider load means most of the requested contents are not served by the intermediate caches, and thus retrieved from their original location. Accessing the main source of contents is very expensive for network operators, and this is why it is important for them to keep the content provider load as low as possible. On the other hand, a low average distance to get contents means a better Quality of Experience for users (QoE). Hence, a good cache allocation strategy should find the best trade-off between these two metrics.

2 Multi-objective cache placement strategy

2.1 System assumptions

Let $G = (V, E)$ be the graph representing a general network of caches, where $V = \{v_1, ..., v_M\}$ depicts the nodes of the network and $E \subset V \times V$ is the set of links connecting the nodes. Each node in the network is equipped with a caching module used to store contents locally. Let $C = \{c_1, ..., c_R\}$ be the set of the catalog's contents available for the users. We assume that all the accessible contents in the system have an identical size and are divided into small packets or chunks, which are in turn of the same size. The cache capacity is then expressed in terms of the number of contents or chunks that can be stored. All the available contents are stored permanently at one or more servers attached to some nodes within the network. In the rest of the chapter and for the sake of readability, we will use the term node/cache interchangeably as well as the terms rank/popularity and content/item/object.

Clients, which are attached to the network nodes, send requests into the network looking for contents. The pattern of these requests is characterized by the Independent Reference Model (IRM) [31]. Suiting the IRM model, users generate an independent and identically distributed sequence of requests from the catalog C of R objects. As it

was described in the previous chapter, the probability p_r to request an item c_r from the set of available contents in the network is constant and follows a popularity law (i.e. Zipf law), where the contents are ranked decreasingly according to their popularity from 1 to R. In the present work, the LRU algorithm is used to manage the node's content store and two caching schemes will be considered: LCE and 2Q.

2.2 Problem formulation

A cache allocation solution (taken for example by an NFV orchestrator), can be defined by a vector $X = (x_1, \ldots, x_M)$, where x_i represents the amount of storage capacity placed in node v_i. To compute the content provider load or the average distance ratio of a configuration of caches X, we use our model MACS (Markov chain-based Approximation of Caching Systems), which we presented in the previous chapter. MACS is an analytic tool that allows us to estimate the cache hit ratio of an interconnection of caches, which can be used to compute other performance metrics like the content provider load and the average distance ratio. By using MACS and for each cache allocation configuration X, we can evaluate the performance of the whole system in its steady-state and not just during a transient phase.

Since in this work the caching capacity is expressed in terms of the number of contents that can be stored, then we have $x_i \in \mathbb{N}$. As we measure the caching gain through evaluating the content provider load and the average distance ratio in the network, our objective is to find a cache distribution solution such that the evaluation metrics that we have chosen are optimized (i.e. minimized). The cache placement is then formulated as a multi-objective optimization problem as follows:

$$\begin{aligned} \underset{X=(x_1,\ldots,x_M)}{\text{minimize}} \quad & f_1(X), f_2(X) \\ \text{subject to} \quad & \sum_{i=1}^{M} x_i \leq T_c, i = 1, \ldots, M, \\ & x_i \in \mathbb{N}, i = 1, \ldots, M. \end{aligned} \tag{3.1}$$

The value of T_c represents the total cache resources to be distributed in the network. The expressions $f_1(X)$ and $f_2(X)$ are both expressed as a percentage and represent, respectively, the content provider load and the average distance ratio of a cache placement configuration X using MACS. The primary function of MACS is to calculate the cache hit ratio of the network's nodes. Then, we can compute $f_1(X)$ and $f_2(X)$ as

follows:

$$
\begin{aligned}
f_1(X) &= \Big(\prod_{i=1}^{s} P_{\textit{miss}}(i) \Big) \times 100, \\
f_2(X) &= \frac{\sum_{i=1}^{s} \Big(\Big(\prod_{j=1}^{i-1} P_{\textit{miss}}(j) \Big) \times P_{\textit{hit}}(i) \times \textit{Distance}(i) \Big)}{\textit{Distance}(s)} \times 100.
\end{aligned}
\tag{3.2}
$$

The values $P_{\textit{miss}}(i)$ and $P_{\textit{hit}}(i)$ represent, respectively, the cache hit and cache miss of a network's node v_i. The expression $\textit{Distance}(i)$ is the distance from where the clients requests were generated to the node v_i. The index s represents the nearest node v_s to the clients to which a content provider is attached (i.e. where a permanent copy of the contents is available). In the definition of $f_1(X)$ and $f_2(X)$ and for sake of clarity, we supposed that the network is formed by a line of s nodes numbered from 1 to s where the clients are attached to node v_1 and the content repository is located just after node v_s. It has to be noted that of course, we can compute these metrics in any type of network topology and that we could consider the delay between nodes instead of distance and thus calculate the average network delay.

Since here we are dealing with a multi-objective optimization problem, in which we want to minimize $f_1(X)$ and $f_2(X)$, the solutions will be a set of *efficient* points usually called the Pareto frontier. Pareto efficiency or optimality implies that a solution to a multi-objective problem is such that no single objective can be improved without deteriorating another one. In our case, a solution X^* is said to be efficient if there is no other solution X such that $f_1(X) < f_1(X^*)$ and $f_2(X) \leq f_2(X^*)$ at the same time, or $f_2(X) < f_2(X^*)$ and $f_1(X) \leq f_1(X^*)$. Given that integer nonlinear programming is an NP-hard problem, solving the cache allocation problem as we modeled below will come at a very high computational cost. More specifically and due to the nonlinearity of the objective functions, we need to perform an exhaustive search method in order to enumerate all possible candidates that respect the problem constraints and find the set of optimal cache distributions. If we look closely to our formulation of the problem, the task of enumerating all possible candidates comes down to computing the *weak composition* of an integer n into k parts, i.e., writing n as the sum of a sequence of non-negative integers. A weak composition $C_{n,k}$ [66] has a cardinality of

$$
|C_{n,k}| = \binom{n+k-1}{k-1} = \frac{(n+k-1)!}{n!(k-1)!}. \tag{3.3}
$$

Algorithm 2 Application of GRASP to the cache allocation problem

```
1: function GRASP(Max_Iterations, λ)
2:     Solution ← Greedy_Randomized_Construction(λ);
3:     Solution ← Local_Search(Solution);
4:     Best_Solution ← Solution;
5:     for k = 1, ..., Max_Iterations − 1 do
6:         Solution ← Greedy_Randomized_Construction(λ);
7:         Solution ← Local_Search(Solution);
8:         if f(Solution) < f(Best_Solution) then
9:             Best_Solution ← Solution;
10:        end if
11:    end for
12:    return Best_Solution;
13: end function
```

In our case, $n = T_c$ and $k = M$, where the set of M non-negative integers has a sum equal to T_c and represents the cache resources allocated to each node of the network. It is clear that $|C_{n,k}|$ is huge for high values of n and k. Therefore, we propose the use of the meta-heuristic GRASP to solve the cache placement problem (see the model (3.1)).

2.3 Solving cache allocation problem using GRASP

2.3.1 Mono-objective GRASP

The Greedy Randomized Adaptive Search Procedure or GRASP [62] is an iterative process, where each iteration consists basically of two steps: construction and local search. The construction phase seeks to build a feasible solution using a greedy randomized approach, whose neighborhood will be investigated during the local search in order to find a local optimum. The pseudo-code of Algorithm 2 depicts the main blocks of a mono-objective GRASP procedure, where *Max_Iterations* is the number of iterations that are performed (we will see later the role of the parameter λ and the case of multi-objective GRASP). The best overall solution is, then, kept as the final result. The construction phase operations are shown in Algorithm 3. Let's recall first that the solution S to be built during the construction phase is defined by a vector $X = (x_1, \ldots, x_M)$, where x_i represents the amount of storage capacity placed in node v_i. Initially, no cache resources is allocated to the nodes, so at the beginning,

Algorithm 3 Construction phase

```
1: function Greedy_Randomized_Construction(λ)
2:     for i = 1, ..., M do
3:         x_i ← 0; // Cache allocated for each node v_i
4:     end for
5:     S ← (x_1, ..., x_M); // Current solution
6:     CS ← ∅; // Initial candidate set CS
7:     while T_c ≠ 0 do // Construction of the solution S
8:         T_c ← T_c − P_c
9:         for i = 1, ..., M do // Create candidate set CS
10:            x_i ← x_i + P_c;
11:            X ← (x_1, ..., x_M);
12:            CS ← CS ∪ {X};
13:            x_i ← x_i − P_c;
14:        end for
15:        Evaluate the incremental costs f(X) ∀X ∈ CS;
16:        f^min ← min{f(X) | X ∈ CS};
17:        f^max ← max{f(X) | X ∈ CS};
18:        RCL ← {X ∈ CS|f(X) ≤ f^min + λ(f^max − f^min)};
19:        Select an element X* from the RCL at random;
20:        S ← {X* = (x*_1, ..., x*_M)}; // S gets the partial solution X*
21:        for i = 1, ..., M do
22:            x_i ← x*_i;
23:        end for
24:    end while
25:    return S;
26: end function
```

$S = (0, \ldots, 0)$. The set CS will contain the candidate elements, which will be used for the solution S. In our case and at each step, CS will contain a set of cache placement configurations X_i where a partial resource that we denote P_c, is taken from the total available cache T_c and allocated to one of the network's nodes. If we have, for example, a network with three nodes, $T_c = 100$ and $P_c = 10$, the initial candidate set will then be: $CS = \{(10, 0, 0), (0, 10, 0), (0, 0, 10)\}$.

Each candidate is then evaluated with a greedy function in order to build a restricted candidate list *RCL*, which will contain some of the candidate set who have the best evaluation values (e.g., $RCL = \{(10, 0, 0), (0, 10, 0)\}$). The limitation criteria of the list cardinality can be either based on the number of elements or based on their quality, as we did in Algorithm 3 (line 18). The elements added to the *RCL* list will then be those hav-

Algorithm 4 Local search phase

```
1: function Local_Search(S)
2:     do
3:         (x_1, ..., x_M) ← S;
4:         for i = 1, ..., M do
5:             x_i ← x_i − P_c;
6:             for j = 1, ..., i − 1 do // Cache resources transfer between nodes to improve the current solution
7:                 x_j ← x_j + P_c;
8:                 if f(x_1, ..., x_M) < f(S) then
9:                     S ← (x_1, ..., x_M)
10:                end if
11:                x_j ← x_j − P_c;
12:            end for
13:            for j = i+1, ..., M do // Cache resources transfer between nodes to improve the current solution
14:                x_j ← x_j + P_c;
15:                if f(x_1, ..., x_M) < f(S) then
16:                    S ← (x_1, ..., x_M)
17:                end if
18:                x_j ← x_j − P_c;
19:            end for
20:        end for
21:    while Solution S is not locally optimal
22:    return S;
23: end function
```

ing an evaluation value inferior to the threshold (i.e., $f(X) \in [f^{\min}, f^{\min}+\lambda(f^{\max}-f^{\min})]$). The value of λ will control the insertion condition of candidate elements to *RCL* ($\lambda \in [0,1]$). The case $\lambda = 1$ is equivalent to a random construction, while $\lambda = 0$ corresponds to a pure greedy algorithm. Once *RCL* is built, one element is randomly selected and added to the solution S being built. The candidate list CS and the evaluation function $f(CS)$ are then updated and the construction is repeated (line 7 to 24) until the total use of the cache budget T_c. If we consider for example that the second element from *RCL* has been chosen, the current partial solution will then be $S = (0, 10, 0)$ and the new candidate list will contain: $CS = \{(10, 10, 0), (0, 20, 0), (0, 10, 10)\}$.

Once the cache budget T_c is distributed, the local search will seek to improve the generated solution (e.g., $S = (20, 40, 40)$) by evaluating its neighborhood (Algorithm 4). The efficiency of a local search method depends on many aspects, such as the neighborhood structure of the considered solution, the neighbors search technique and the

starting solution itself. Two methods can be used for the neighborhood search: the *best-improving* strategy or *the first-improving* one. The best-improving strategy consists on investigating all the neighbors and the current solution will then be replaced by the best neighbor found. In the case when a first-improving method is used, the current solution will be replaced by the first neighbor whose evaluation value is better. In our case, we used for the neighborhood search a best-improving strategy as follows: starting from the solution generated by the greedy randomized construction, we transfer an amount of cache P_c from one node to another and explore all the possible cases looking for a cache configuration whose evaluation function value is better than the current one (e.g., *Neighbor*$(S) = \{(10, 50, 40), (30, 30, 40),$ etc.$\}$). We repeat these steps until the current solution can no longer be improved, which then will be returned as the output of the local search procedure of GRASP.

As for the complexity of GRASP algorithms (i.e. construction and local search) applied to our formulation of the cache allocation problem, we have a complexity of $O(T_c\ M)$ for the construction phase and $O(M^2)$ for the local search phase.

2.3.2 Multi-objective GRASP

In single-objective GRASP, only one greedy function is used to evaluate the candidate solutions during the construction and local search phases. In multi-objective GRASP [67], we have in the general case k functions ($f_1(X), \ldots, f_k(X)$) to evaluate a candidate element X and the objective functions can be used mainly in two ways:

— Pure construction/local search: this method consists on using one single objective during each phase of the solution generation. The selection of which evaluation function to be used in each iteration can be done using the following approaches:

- ► Pure-random approach: using this method means that the entire construction or local search is guided by only one single evaluation function that we select randomly from the set of the objective functions ($f_1(X), \ldots, f_k(X)$).
- ► Pure-ordered approach: when this method is adopted, we select the greedy function in an ordered way and one at a time and use it in all the steps of the construction phase or the local search. In other words, the candidate elements are evaluated with $f_1(X)$ during the first iteration. Then, we use $f_2(X)$ in the second iteration, and so on until we reach the $k+1$-th iteration, in which we go back again to $f_1(X)$.

— Combined construction/local search: this method consists on using more than one greedy function in each iteration by means of one of the following approaches:
  - ▶ Sequential combined approach: by using this process, each step of the construction or local search is guided by a different evaluation function. The choice of which objective to choose between each construction step can be done either randomly or in an ordered way.
  - ▶ Weighted combined approach: with this method, a weighted combination of the objective functions is used in each step of the construction or local search [1]. We can either change the weights between the steps of each phase or keep them the same during the whole process. It has to be noted that the weights can help us scale evaluations functions having significantly varied magnitudes into similar and comparable ones. In addition, some greedy functions in multi-objective optimization can be minimized while others maximized. In this case, the weights can take positive and negative values in order to take into account this fact.

In our case, we have two evaluation functions to be used in the multi-objective GRASP: $f_1(X)$ and $f_2(X)$ and there are many methods that can be used to compute the outcome of each configuration in the construction and local search phases. For example, one can use $f_1(X)$ in the construction phase and $f_2(X)$ during the local search and vice versa, or choose randomly between the objective functions during each GRASP iteration. We can also consider a weighted combined method ($f(X) = w_1 f_1(X) + w_2 f_2(X)$). The choice of which method to be used to evaluate a candidate solution (or partial solution) can depend on some desired results that should be achieved or some constraints that should be respected.

In our work, we considered two methods to generate the solutions based on the defined objective functions $f_1(X)$ and $f_2(X)$. In the first method, which matches a pure-ordered approach, we use in each iteration one objective function ($f_1(X)$ or $f_2(X)$) during the construction phase. The other one is then used during the local search. Then, we alternate between the selected functions for each phase in the next iteration. For example, if we chose in the first iteration respectively $f_1(X)$ and $f_2(X)$ for the construction and local search phases then, in the second iteration, we will use $f_1(X)$

1. Weighted combined construction: $f(X) = \sum_{i=1}^{k} w_i f_i(X)$, where w_i is the weight of the evaluation function $f_i(X)$.

in the local search and $f_2(X)$ in the construction phase, etc. This approach will allow us to produce diversified solutions of good quality. In the second method, we used a weighted combination of the evaluation functions as follows:

$$f(X) = \Big(0.5 \times \frac{|f_1(X) - 100|}{|\mathit{Opt}(f_1(X)) - 100|} + 0.5 \times \frac{|f_2(X) - 100|}{|\mathit{Opt}(f_2(X)) - 100|}\Big) \times 100. \tag{3.4}$$

The expressions *Opt*$(f_1(X))$ and *Opt*$(f_2(X))$ represent respectively the theoretical optimal values of the metrics $f_1(X)$ and $f_2(X)$ for a fixed cache resources budget T_c. More specifically, *Opt*$(f_1(X))$ is obtained by allocating all the available cache resources at the root node and then, computing the content provider load. As for *Opt*$(f_2(X))$, it is obtained by computing the average distance ratio metric where all the total cache resources are placed at one-hop from the clients. Since here we want to minimize $f_1(X)$ and $f_2(X)$ (expressed as percentages), then the range of values taken by these functions is: $[Opt(f_i(X)), 100]$ ($i = 1, 2$). The value 100 represents the worst case, i.e., where no cache resources were allocated and thus, all the contents are retrieved from the main source and the average distance to get objects is not reduced. So, what we did is to compute how much improvement can be achieved in each of these performance metrics relatively to the optimal values that can be reached. For example, and for a certain configuration $X = (x_1, \ldots, x_M)$, if we have $f_1(X) = 40\%$ and $Opt(f_1(X)) = 20\%$, then $(|40 - 100|/|20 - 100|) \times 100 = 75\%$. This value means that we have achieved 75% of the optimal outcome of the content provider load.

The evaluation function $f(X)$ can be seen as a ratio of efficiency and it can be used in a case scenario where the aim is to improve as much as possible the network's overall performance. It is also a mean to scale the evaluation functions $f_1(X)$ and $f_2(X)$ into similar and comparable magnitudes. In addition, the weighting coefficients (fixed to 0.5 in our case) can be tuned in order to give more importance to one metric over the other. In the case where the weighted combined approach defined by $f(X)$ is adopted, the cache placement is then formulated as the following optimization problem:

$$\begin{aligned} \underset{X=(x_1,\ldots,x_M)}{\text{maximize}} \quad & f(X) \\ \text{subject to} \quad & \sum_{i=1}^{M} x_i \leq T_c, i = 1, \ldots, M, \\ & x_i \in \mathbb{N}, i = 1, \ldots, M. \end{aligned} \tag{3.5}$$

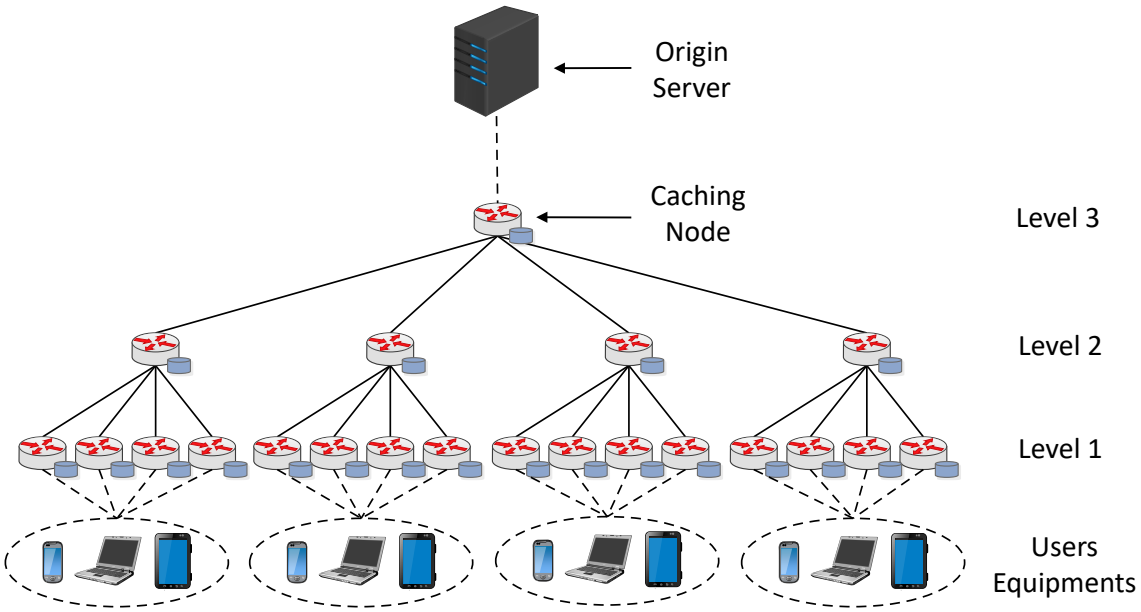


Figure 3.1 – An example of a network architecture for content delivery with cache-enabled nodes.

3 Performance evaluation

In this section, we present the evaluation results of the cache allocation problem using the methods presented previously. We will expose at first different results comparing the outcomes of the metaheuristic to the optimal solutions (i.e. obtained using an exhaustive search approach). Then, we will display some of the results obtained using our application of GRASP to the problem that we have formulated, to have an idea about the solutions that can be proposed by our approaches.

3.1 Model configuration

The different tests were conducted on a typical three-level network that contains 21 nodes forming a perfect 4-ary tree topology where the distance and the latency between each two adjacent nodes are the same (see Figure 3.1). We considered in our experiments a catalog of contents containing $20,000$ 1-chunk sized objects whose popularity distribution follows the Zipf's law. Permanent copies of the available contents are hosted on one repository attached to the root node of the network, and the users are attached to the network's leaves (i.e. level 1 of the network). We tested different

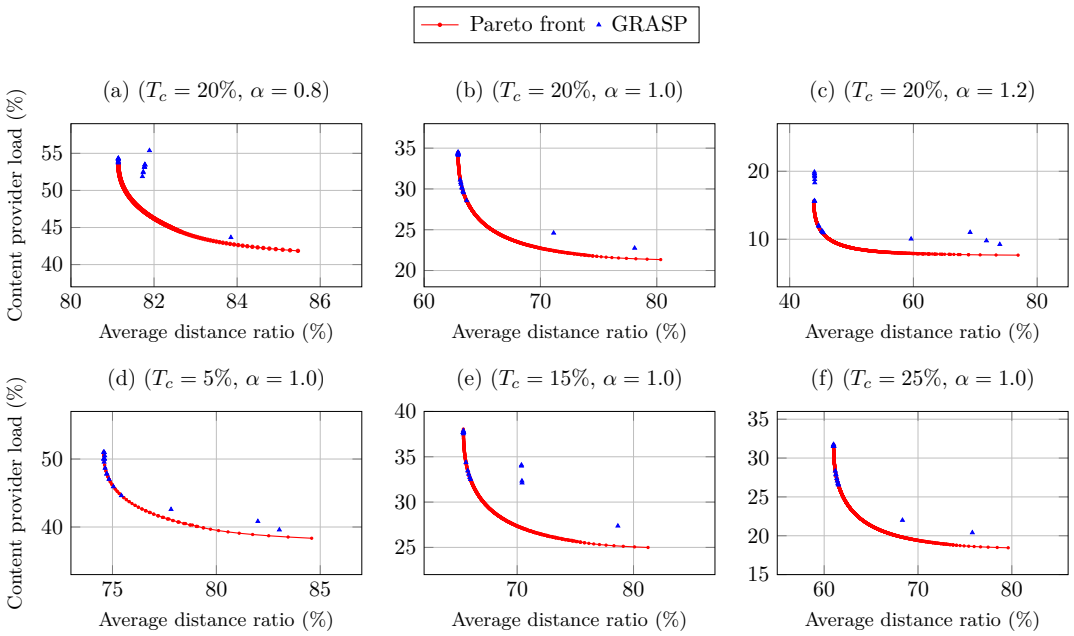


Figure 3.2 – Comparison between the Pareto front and GRASP solutions under the LCE scheme.

values of the Zipf law's skew parameter α going from 0.8 to 1.2. The parameter λ that controls the amounts of greediness and randomness in GRASP was set to 0.5 and the total cache T_c is expressed as a ratio of the catalog size. Two caching strategies were used during the experiments: LCE and 2Q. The 2Q policy is of course more efficient than LCE as it was shown in the previous chapter but the aim here to see the behaviour of our model where different caching strategies are used. As we mentioned previously, the network performance is evaluated using the content provider load and average distance ratio metrics. For sake of simplicity, the cache resources are allocated in a way that the nodes located on the same level of the network have the same cache size. Since there are 16 nodes on the first level of the network, P_c should be a multiple of 16 and in our case, it was set to 16. Thus, a cache resource placement can be described as $X = (x_1, x_2, x_3)$, where x_i represents the total cache allocated at level i of the network.

3.2 Model results and analysis

In Figures 3.2-3.3, we compare the network's performance metrics of the different cache distribution solutions generated by GRASP (20 iterations) with the Pareto

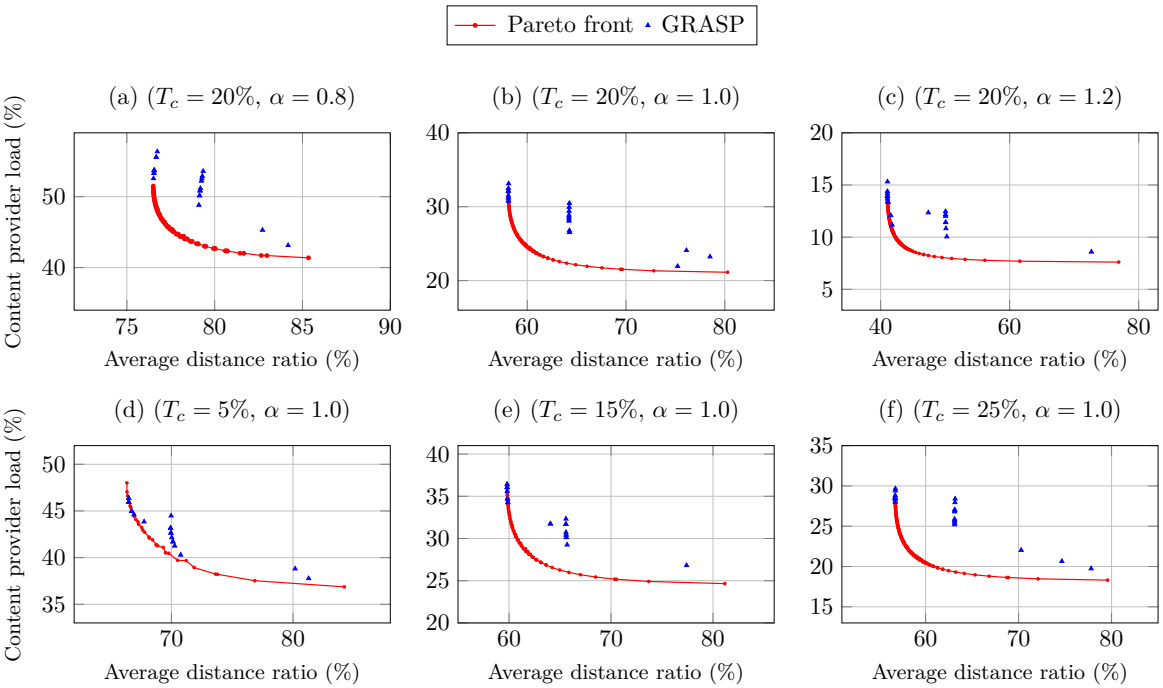


Figure 3.3 – Comparison between the Pareto front and GRASP solutions under the 2Q scheme.

front using the separate evaluation functions approach (content provider load and average distance ratio metrics). We can see from the plots how the solutions produced by GRASP, independently from the used caching scheme, are very close to the set of dominant points in the different tested scenarios, which reflects the good quality of the metaheuristic outcomes.

In Figure 3.4-3.5, we compare the solutions obtained using the weighted combined approach with the optimal values (only the best solution obtained with GRASP is shown). The results in these graphs also show the efficiency of the metaheuristic in giving solutions close to the optimal ones. When the Zipf's parameter α is set to 0.8, we observe that the solutions generated by GRASP are not as close to the optimal values as the results obtained in the other cases. Having a low value of α ($\alpha < 1$) means a lower difference in popularity between the different contents, which makes it more difficult to construct cache distribution solutions using GRASP that approach the optimal performance.

In Tables 3.1-3.2, we display the cache allocation solutions generated by GRASP (only the results of four iterations of GRASP are shown) using separate evaluation

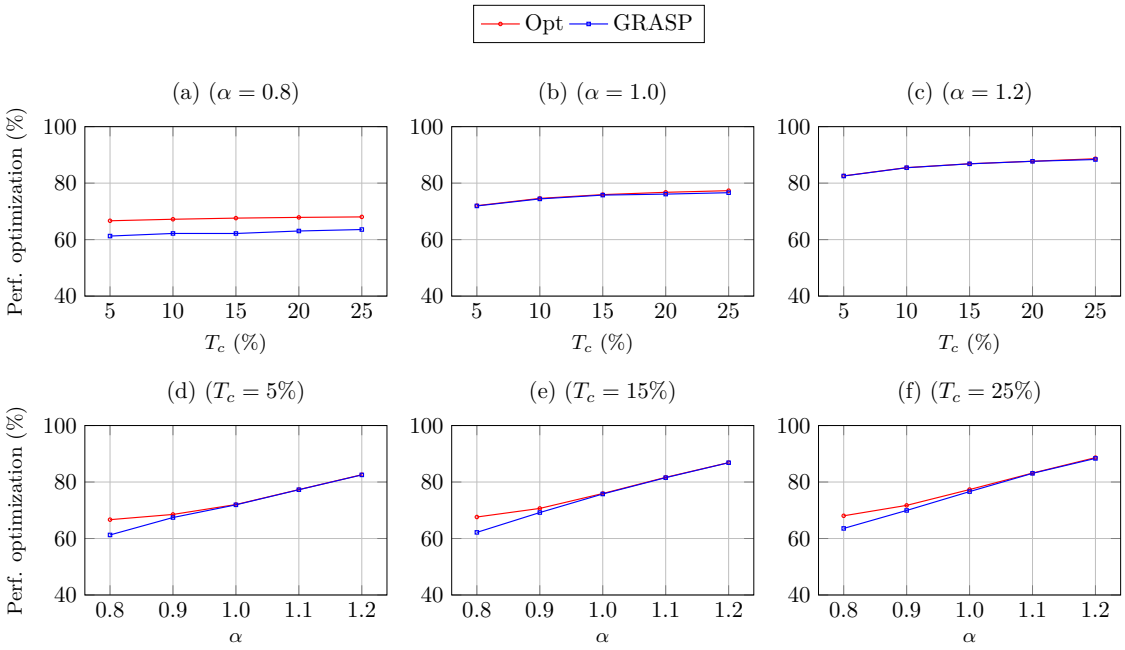


Figure 3.4 – Performance optimization vs total cache size using GRASP compared to the optimal values under the LCE scheme.

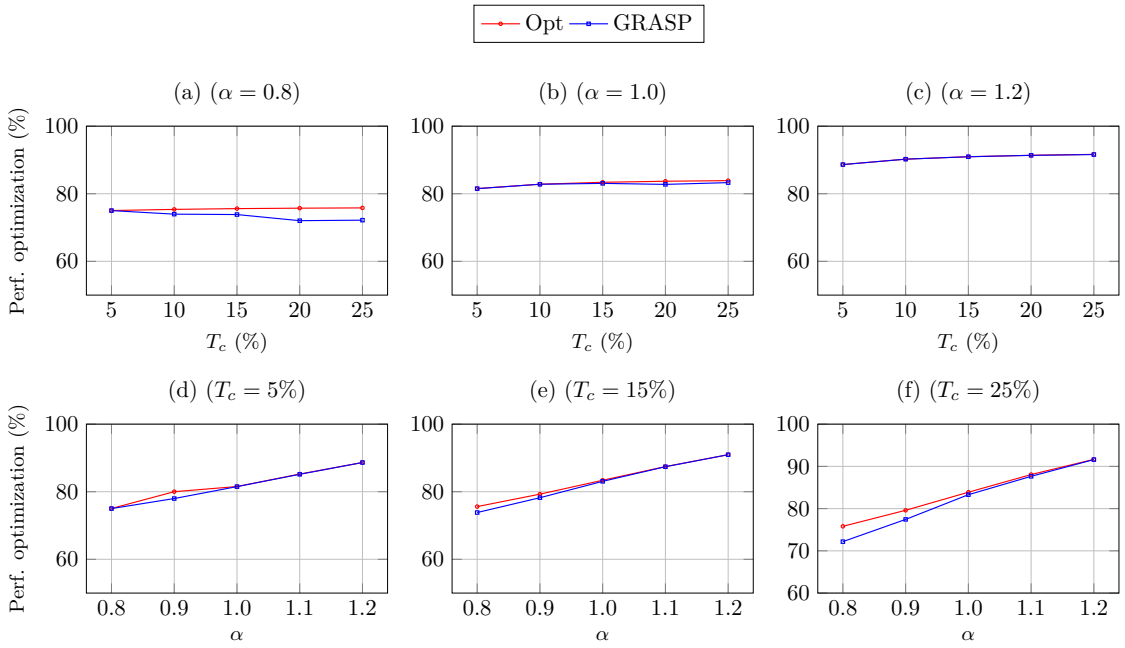


Figure 3.5 – Performance optimization vs total cache size using GRASP compared to the optimal values under the 2Q scheme.

Table 3.1 – Cache allocation solutions using GRASP with two separate evaluation functions when LCE is used.

$T_c = 20\%$				$\alpha = 1.0$			
α	X	$f_1(X)$	$f_2(X)$	T_c	X	$f_1(X)$	$f_2(X)$
0.8	(320,1696,2080)	53.47%	81.78%	5%	(224,208,592)	44.63%	75.42%
	(768,1376,1952)	54.80%	81.15%		(288,208,528)	45.94%	75.04%
	(736,1280,2080)	53.84%	81.14%		(352,224,448)	47.81%	74.72%
	(16,2288,1792)	55.38%	81.89%		(400,240,384)	49.56%	74.59%
1.0	(1536,1168,1392)	34.39%	62.92%	15%	(736,688,1648)	32.50%	65.99%
	(944,880,2272)	28.57%	63.65%		(864,640,1568)	33.09%	65.83%
	(1136,864,2096)	29.54%	63.38%		(992,672,1408)	34.36%	65.59%
	(1280,928,1888)	30.80%	63.14%		(1168,832,1072)	37.54%	65.38%
1.2	(1952,1040,1104)	15.62%	44.00%	25%	(1264,1184,2672)	26.54%	61.52%
	(1760,2176,160)	19.88%	44.00%		(1392,1168,2560)	27.06%	61.39%
	(1456,720,1920)	12.02%	44.67%		(1568,1152,2400)	27.84%	61.25%
	(1088,800,2208)	11.17%	45.24%		(1904,1488,1728)	31.75%	61.00%

functions approach and testing two different caching schemes: LCE (Table 3.1) and 2Q (Table 3.2). Depending on the Zipf law's parameter α, the total cache resources available T_c and the adopted caching strategy, different solutions will be generated by GRASP. The choice of the final solutions between the different proposed ones can be based on some constraints that should be respected. For example, and under the LCE policy, if $T_c = 20\%$ and $\alpha = 1.2$ (Table 3.1) and if the priority is to minimize the content provider load ($f_1(X)$), then the allocation $X = (1088, 800, 2208)$ is the best one. In case when the 2Q scheme is used, if $\alpha = 1.0$ and $T_c = 15\%$ (Table 3.2) and one wants to minimize first the average distance ratio ($f_2(X)$), then the allocation $X = (1408, 768, 896)$ is the one to choose.

In Tables 3.3-3.4, we expose the cache allocation solutions generated by GRASP using a weighted combined evaluation approach. In this case scenario, the objective is to optimize the overall network performance without considering separately the evaluation metrics. The proposed solution will be the cache allocation that gives the best outcome. For example, and under the LCE strategy, if $T_c = 20\%$ and $\alpha = 1.0$ (Table 3.3), the solution $X = (1056, 816, 2224)$ is the best one. In case when the 2Q scheme is used and if we have for example $\alpha = 1.0$ and $T_c = 25\%$ then, the cache distribution $X = (1328, 1056, 2736)$ is the best solution generated by GRASP.

Table 3.2 – Cache allocation solutions using GRASP with two separate evaluation functions when 2Q is used.

$T_c = 20\%$				$\alpha = 1.0$			
α	X	$f_1(X)$	$f_2(X)$	T_c	X	$f_1(X)$	$f_2(X)$
0.8	(64,2240,1792)	52.90%	79.29%	5%	(320,384,320)	44.59%	66.92%
	(1216,832,2048)	52.56%	76.51%		(512,256,256)	46.35%	66.49%
	(1344,896,1856)	53.72%	76.55%		(192,512,320)	43.84%	67.75%
	(1472,1216,1408)	56.35%	76.73%		(64,448,512)	41.70%	70.12%
1.0	(64,2048,1984)	28.60%	64.23%	15%	(64,1280,1728)	30.42%	65.60%
	(64,1984,2048)	30.44%	58.13%		(1152,704,1216)	34.27%	59.88%
	(1600,768,1728)	30.83%	58.11%		(1216,704,1152)	34.70%	59.85%
	(1792,1024,1280)	33.13%	58.12%		(1408,768,896)	36.40%	59.81%
1.2	(448,2432,1216)	12.34%	47.39%	25%	(64,3200,1856)	28.40%	63.17%
	(1728,960,1408)	13.41%	41.78%		(1920,960,2240)	27.99%	56.74%
	(1984,896,1216)	14.13%	41.05%		(1984,1024,2112)	28.53%	56.73%
	(2176,1024,896)	15.31%	41.07%		(2112,1152,1856)	29.66%	56.75%

Table 3.3 – Cache allocation solutions using GRASP with weighted combined evaluation function when LCE is used.

$T_c = 20\%$			$\alpha = 1.0$		
α	X	$f(X)$	T_c	X	$f(X)$
0.8	(592,1248,2256)	62.41%	5%	(176,176,672)	71.92%
	(512,1600,1984)	60.70%		(208,192,624)	71.69%
	(464,1824,1808)	59.55%		(272,176,576)	71.31%
	(432,1904,1760)	59.23%		(352,208,464)	69.91%
1.0	(1056,816,2224)	76.12%	15%	(672,512,1888)	75.76%
	(1152,864,2080)	75.77%		(768,592,1712)	75.40%
	(1232,896,1968)	75.46%		(912,624,1536)	74.85%
	(1328,912,1856)	75.10%		(1040,688,1344)	74.02%
1.2	(1712,176,2208)	87.83%	25%	(1328,1056,2736)	76.61%
	(1824,96,2176)	87.72%		(1408,1088,2624)	76.40%
	(1968,80,2048)	87.70%		(1520,1136,2464)	76.06%
	(2240,16,1840)	87.59%		(1648,1184,2288)	75.62%

Table 3.4 – Cache allocation solutions using GRASP with weighted combined evaluation function when 2Q is used.

$T_c = 20\%$			$\alpha = 1.0$		
α	X	$f(X)$	T_c	X	$f(X)$
0.8	(704,832,2560)	72.03%	5%	(192,128,704)	81.51%
	(768,1088,2240)	70.36%		(192,256,576)	81.44%
	(768,1344,1984)	68.92%		(192,448,384)	80.49%
	(704,1600,1792)	67.79%		(256,448,320)	80.21%
1.0	(1024,576,2496)	82.77%	15%	(640,384,2048)	83.06%
	(1088,896,2112)	81.79%		(704,512,1856)	82.60%
	(1216,1216,1664)	80.35%		(768,640,1664)	82.05%
	(1152,1408,1536)	79.88%		(832,896,1344)	80.89%
1.2	(1152,384,2560)	91.36%	25%	(1152,576,3392)	83.29 %
	(1280,512,2304)	91.21%		(1280,768,3072)	82.72%
	(1408,832,1856)	90.72%		(1344,1088,2688)	81.88%
	(1536,1152,1408)	89.99%		(1472,1344,2304)	80.88%

If we compare between LCE and 2Q in terms of network performance when different cache allocation solutions are used, the 2Q policy generally has better outcomes than LCE. However, it can be possible that LCE has better results in certain cases. For example, if we look at Tables 3.1-3.2 and in the case where $\alpha = 1.0$ and $T_c = 25\%$, the cache allocation $X = (1264, 1184, 2672)$ when LCE is used achieves better performance than the cache allocation $X = (64, 3200, 1856)$ when 2Q is applied. This result reinforces the importance of placing the cache resources in distributed networks and how it can affect drastically the network performance.

We can see from the exposed different results that there is no absolute solution for the cache allocation problem and it is not a question about whether to cache at the edge or in the core. Depending on the network's configuration and the objectives that one wants to achieve, multiple solutions can be adopted. Other metrics can be used as evaluation functions during the building of the solutions instead of those that we used in this work like, for example, the financial cost of cache resources deployment. The aim of our proposal is to give a tool capable of efficiently allocating distributed caching resources and versatile enough to adapt to specific desired network performance and constraints.

4 Conclusion

We addressed in this chapter the problem of how to efficiently allocate cache resources in multi-cache networks. We first model it as a multi-objective optimization problem where our analytic tool MACS is used to evaluate the objective functions. Our formulation of the problem turned out to be an NP-hard problem and finding the optimal solutions in this case is not always practical. Thus, we propose an adaptation of the GRASP metaheuristic to solve the problem using different evaluation methods to generate the solutions. The experimental results have showed how the outcomes of our algorithm are very close to the optimal ones and how the proposed solutions depend on different parameters like the total cache budget. The versatility of our proposal allows it to be applied to any arbitrary network topology, and it can include other cost functions to be used as objective functions.

CONCLUSION

1 Overview

Over the past few years, the Internet usage has switched from a host-centric model to a content-centric approach, especially when dealing with content retrieval and data dissemination. This evolution is mostly driven by the increased popularity of content-oriented services, e.g., Peer-to-Peer file sharing, Video on Demand, video/audio streaming and social networks where users focus more on contents and not on the physical locations from which contents can be retrieved. These new trends in the Internet usage have raised important challenges in the current IP-based infrastructures, especially in terms of content delivery.

To address these challenges, the Content-Centric Networking has been proposed and emerged as one of the most interesting alternatives to the existing network architecture. Instead of the end-to-end principle, the CCN paradigm consists in redesigning the future Internet architecture by focusing on named data and leveraging new features, such as in-network caching, multipath connectivity and multicast data delivery. The content name in CCN is the only identifier of data, and the information exchanges based on establishing communication channels are abandoned.

The ubiquitous in-network caching certainly represents one of the most important features, which impacts directly the content delivery performance. In fact, CCN nodes are equipped with content store modules and have the ability to cache the data that passes by them. Therefore, the end-users' requests (known as interests in CCN), that are routed toward the Content Providers' servers, can be satisfied by the cached data at the intermediate nodes.

In-network caching as provided by CCN has brought a renewed interest in developing efficient tools to study the caching performance of interconnected systems of caches. This will help, indeed, in giving guidance and offering insights about the behaviour of a network of caches. Besides, this can be used to optimize the caching efficiency while increasing the overall system performance, which was the aim of this thesis work where the focus was on cache modeling and cache resources allocation in

CCN and multi-cache networks in general.

In chapter one, we start first by giving an overview of the Information-Centric Networking paradigm to focus after that on Content-Centric Network, which represents a promising ICN solution for the future Internet. Then, we analyze the existing works related to the caching feature of ICNs and of multi-cache systems in general. More specifically, we discuss the issues related to cache modeling and cache resources allocation in multi-cache networks, which represent the core of this thesis work.

The second chapter starts by introducing an analytical model based on Markov chains named MACS (Markov chain-based Approximation of Caching Systems). This model allows us to compute the hit probability of an LRU cache operating under the default caching scheme called LCE where the contents are always saved when received. By proposing a methodology that allows modeling the caching decision process in the general case, we were able to extend MACS so that it can be used to analyze different caching algorithms different from LCE. We aim with our proposal to gain more insights on the efficiency and limits of CCN caching strategies, with a focus on the schemes having a good performance/overhead trade-off through the analysis of different network performance metrics. We have shown how the versatility of our cache modeling tool enables to mimic efficient caching strategies such as LCD and 2Q and it can be easily adapted to represent an interconnection of caches under different schemes. Even though our proposal was first used in the context of CCN networks, it can of course be applied to multi-cache networks in general. In addition to the cache hit, MACS can be used to compute different performance metrics like the content provider load and the distance reduction ratio. The conducted tests have shown the accuracy of our model in estimating the cache hit rate of a multi-cache system under different caching schemes. The obtained results also indicate the efficiency of the 2Q algorithm and its performance/overhead trade-off makes it a good candidate to be used as a caching strategy.

In the third chapter, the optimal placement of caching resources is investigated. The cache allocation issue is one of the most important problems to address when studying multi-cache networks because of the expensive cost of deploying distributed storage resources along the network. In this context, we propose an approach that solves the trade-off between minimizing the hit rate in the origin server and minimizing the distance to retrieve contents. To do so, we modelled the cache allocation problem as a multi-objective integer nonlinear program where our analytic tool MACS is used

to evaluate the objective functions. Our formulation of the problem turned out to be an NP-hard problem. Thus, we proposed an adaptation of the metaheuristic GRASP to solve the problem using different evaluation functions to generate the solutions. The conducted tests have showed the closeness of the metaheuristic's outcomes to the optimal ones. By analyzing the different obtained results, we have shown that there is no absolute solution for the cache allocation problem and in contrast to the conclusions made in previous works, it is not a question about whether to cache at the edge or in the core of the network. The distribution of cache resources will depend on the network's configuration and the objectives that one wants to achieve. The aim of the work presented in the third chapter is to give a tool capable of efficiently allocation distributed caching resources that takes into account more than one performance metric and versatile enough to adapt to specific desired network performance and constraints.

2 Perspectives

The state-of-the-art presented in chapter one shows the extent of an active research domain in the areas of cache modeling and cache optimization in Content-Centric Networks and multi-cache systems in general. With the proliferation of CDN-based paradigms and the upcoming advent of 5G, this domain is rapidly evolving and multiple research opportunities that were not addressed in this thesis and related to our work deserve in our opinion to be looked at.

2.1 Dynamic content popularity

Understanding the evolution of content popularity plays a key role on the cache optimization. It has a direct effect on how caching resources are distributed within networks and on the design of efficient caching strategies and thus, impacting the overall performance of networks. A large amount of existing works in the literature (including our work) that studied caching uses the well-known Independent Reference Model (IRM) along with the Zipf law to shape the content popularity. Under the IRM model, the requests for contents occur in an independent and identically distributed (i.i.d.) sequence. In other words, the probability to request an item c_r from the set of available contents in the network is proportional to a popularity law p_r and does not evolve in time. However, the popularity and the catalog of contents in reality and in many cases can be far

from being stationary, as they are more likely to evolve quickly [68]. Proposing a model to capture the popularity of contents with non-stationary behaviour remains an open issue.

2.2 Caching from an economic point of view

The endless race of content providers towards even more content and even more quality is increasing data consumption as never before. This growth is such that it is undermining the infrastructures of Internet Service Providers, which require to be constantly upgraded in order to relieve the load on the ISPs networks [69] with huge capital investments. This led to the proliferation of Content Delivery Networks, which has become an essential part of the Internet ecosystem and has been a reliable solution for Content Providers to deliver their content. The advent of NFV [64] and ICN-based paradigms, especially caching, as well as the dynamic orchestration of resources bring much more flexibility to cache-based infrastructures and their management. These features will be, therefore, major drivers that will certainly accelerate these developments and make them available to Content Providers [70]. These recent developments represent a great opportunity for ISPs by allowing them to easily develop caching capabilities within their infrastructures, in order to offer services with better Quality of Experience (QoE) for end-users as well as the reduction of traffic at peering levels, transport and core networks. In addition, they will be able to share their infrastructure by offering these caching capabilities to third parties, which makes it possible to multiply sources of revenues. The study of caching along with the cooperation of CDNs and multi-cache networks with ISPs from an economic point of view is becoming a more interesting area to look at due to the enhancement of flexibility in managing cache resources and it is in our opinion a hot topic to investigate.

2.3 Caching in wireless networks

During these last years, there is growing consensus that caching is placed to play a key role in future communication systems and networks, especially with the upcoming advent of 5G [14]. In the different proposals that we have presented in this thesis, the focus was basically on static networks where the system configuration does not evolve much in time, which is not the case in cache-enabled wireless networks. Studying

caching in this type of networks has already started in the literature [71]-[72] and many different research directions have been not sufficiently explored yet. One interesting subject that we did not have the opportunity to investigate in this thesis and that in our opinion needs to be addressed is the analytic modelisation and performance evaluation of cache-enabled wireless networks. In chapter two of this thesis, we proposed a generic model of static multi-cache networks and it is not straightforward to apply it to multi-cache wireless networks. In these latter, features like users mobility and heterogeneity of cache resources (caching at users devices, caching at base stations, etc.) impose additional difficulties and should be taken into account when studying such systems. Designing analytical models of cache-enabled wireless networks remains an open issue.

PUBLICATIONS

— **Hamza Ben Ammar**, Soraya Ait Chellouche and Yassine Hadjadj-Aoul, “CLIC : A Clique-based cooperative caching for Content-Centric Networking”, in IEEE Global Information Infrastructure and Networking Symposium (GIIS), October 2016, pp. 1-6.

— **Hamza Ben Ammar**, Soraya Ait Chellouche and Yassine Hadjadj-Aoul, “A Markov chain-based Approximation of CCN caching Systems”, in IEEE Symposium on Computers and Communications (ISCC), July 2017, pp. 327-332.

— **Hamza Ben Ammar**, Yassine Hadjadj-Aoul, Gerardo Rubino and Soraya Ait Chellouche, “A versatile Markov chain model for the performance analysis of CCN caching systems”, in IEEE Global Communications Conference (Globecom), December 2018, pp. 1-6.

— **Hamza Ben Ammar**, Yassine Hadjadj-Aoul and Soraya Ait Chellouche, “Efficiently allocating distributed caching resources in future smart networks”, in IEEE Consumer Communications and Networking Conference (CCNC), January 2019, pp.1-4.

— **Hamza Ben Ammar**, Yassine Hadjadj-Aoul, Gerardo Rubino and Soraya Ait Chellouche, “On the performance analysis of distributed caching systems using a customizable Markov chain model”, in Elsevier Journal of Network and Computer Applications (JNCA), March 2019, pp. 39-51.

BIBLIOGRAPHY

[1] J. Pan, S. Paul, and R. Jain, « A survey of the research on future internet architectures », *IEEE Communications Magazine*, vol. 49, pp. 26–36, Jul. 2011.

[2] Cisco, *Cisco Visual Networking Index: forecast and methodology, 2016–2021*, White paper, Jun. 2017.

[3] Cisco, *Cisco Visual Networking Index: forecast and trends, 2017–2022*, White paper, Nov. 2018.

[4] A. V. Vasilakos, Z. Li, G. Simon, and W. You, « Information centric network: Research challenges and opportunities », *Elsevier Journal of Network and Computer Applications*, vol. 52, pp. 1–10, Jun. 2015.

[5] T. Hau, D. Burghardt, and W. Brenner, « Multihoming, content delivery networks, and the market for Internet connectivity », *Elsevier Telecommunications Policy*, vol. 35, pp. 532–542, Jul. 2011.

[6] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, « Should Internet Service Providers Fear Peer-assisted Content Distribution? », *in Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC)*, Oct. 2005, pp. 63–76.

[7] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou, « More control over network resources: An ISP caching perspective », *in Proceedings of the 9th International Conference on Network and Service Management*, Oct. 2013, pp. 26–33.

[8] A. J. Smith, « Cache Memories », *ACM Computing Surveys*, vol. 14, pp. 473–530, Sep. 1982.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, « Web caching and Zipf-like distributions: evidence and implications », *in IEEE INFOCOM Proceedings*, Mar. 1999, pp. 126–134.

[10] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, « A survey of information-centric networking », *IEEE Communications Magazine*, vol. 50, pp. 26–36, Jul. 2012.

[11] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. C. Polyzos, « A survey of information-centric networking research », *IEEE Communications Surveys and Tutorials*, vol. 16, pp. 1024–1049, Jul. 2014.

[12] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, « Networking Named Content », *in Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, Dec. 2009, pp. 1–12.

[13] G. Rossini and D. Rossi, « A dive into the caching performance of Content Centric Networking », *in IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*, Sep. 2012, pp. 105–109.

[14] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, « The Role of Caching in Future Communication Systems and Networks », *IEEE Journal on Selected Areas in Communications*, vol. 36, Nov. 2018.

[15] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, « On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration », *IEEE Communications Surveys Tutorials*, vol. 19, pp. 1657–1681, May 2017.

[16] K. S. Reddy, S. Moharir, and N. Karamchandani, « Effects of storage heterogeneity in distributed cache systems », *in 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2018, pp. 1–8.

[17] A. Mahanti, N. Carlsson, A. Mahanti, M. Arlitt, and C. Williamson, « A tale of the tails: Power-laws in internet measurements », *IEEE Network*, vol. 27, pp. 59–64, Jan. 2013.

[18] H. Ben-Ammar, S. Ait-Chellouche, and Y. Hadjadj-Aoul, « A Markov chain-based Approximation of CCN caching Systems », *in IEEE Symposium on Computers and Communications*, Jul. 2017, pp. 327–332.

[19] H. Ben-Ammar, Y. Hadjadj-Aoul, G. Rubino, and S. Ait-Chellouche, « A versatile Markov chain model for the performance analysis of CCN caching systems », *in IEEE Global Communications Conference*, Dec. 2018, pp. 1–6.

[20] H. Ben-Ammar, Y. Hadjadj-Aoul, G. Rubino, and S. Ait-Chellouche, « On the performance analysis of distributed caching systems using a customizable Markov chain model », *Journal of Network and Computer Applications*, vol. 130, pp. 39–51, Mar. 2019.

[21] H. Ben-Ammar, Y. Hadjadj-Aoul, and S. Ait-Chellouche, « Efficiently allocating distributed caching resources in future smart networks », *in IEEE Annual Consumer Communications Networking Conference*, Jan. 2019, pp. 1–4.

[22] A. Zahemszky, B. Gajic, C. Rothenberg, C. Reason, D. Trossen, D. Lagutin, J. Tuononen, and K. Katsaros, « Experimentally-Driven Research in Publish/Subscribe Information-Centric Inter-Networking », *in Testbeds and Research Infrastructures: Development of Networks and Communities*, vol. 46, May 2011, pp. 469–485.

[23] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, « Elsevier Network of Information (NetInf) - An information-centric networking architecture », *Computer Communications*, vol. 36, pp. 721–735, Apr. 2013.

[24] T. Koponen, M. Chawla, B. G. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica, « A Data-oriented (and Beyond) Network Architecture », *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 181–192, Aug. 2007.

[25] M. Amadeo, C. Campolo, A. Molinaro, and G. Ruggeri, « Content-centric wireless networking: A survey », *Elsevier Computer Networks*, vol. 72, pp. 1–13, Oct. 2014.

[26] J. Kurose, « Information-centric networking: The evolution from circuits to packets to content », *Computer Networks*, vol. 66, pp. 112–120, Jun. 2014.

[27] M. Zhang, H. Luo, and H. Zhang, « A Survey of Caching Mechanisms in Information-Centric Networking », *IEEE Communications Surveys Tutorials*, vol. 17, pp. 1473–1499, Aug. 2015.

[28] H. Jin, D. Xu, C. Zhao, and D. Liang, « Information-centric mobile caching network frameworks and caching optimization: a survey », *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, p. 33, Feb. 2017.

[29] W. F. King, « Analysis of Paging Algorithms », *in IFIP Congress*, Aug. 1971, pp. 485–490.

[30] P. Flajolet, D. Gardy, and L. Thimonier, « Birthday paradox, coupon collectors, caching algorithms and self-organizing search », *Discrete Applied Mathematics*, vol. 39, pp. 207–229, Nov. 1992.

[31] A. Dan and D. Towsley, « An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes », *SIGMETRICS Performance Evaluation Review*, vol. 18, pp. 143–152, Apr. 1990.

[32] E. J. Rosensweig, J. Kurose, and D. Towsley, « Approximate Models for General Cache Networks », *in IEEE INFOCOM Proceedings*, Mar. 2010, pp. 1–9.

[33] I. Psaras, R. Clegg, R. Landa, W. Chai, and G. Pavlou, « Modelling and Evaluation of CCN-caching Trees », *in Proceedings of the 10th International IFIP Conference on Networking*, May 2011, pp. 78–91.

[34] H. Che, Y. Tung, and Z. Wang, « Hierarchical Web caching systems: modeling, design and experimental results », *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1305–1314, Sep. 2002.

[35] C. Fricker, P. Robert, and J. Roberts, « A versatile and accurate approximation for LRU cache performance », *in Proceedings of the 24th International Teletraffic Congress*, Sep. 2012, pp. 1–8.

[36] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, « Performance evaluation of hierarchical TTL-based cache networks », *Computer Networks*, vol. 65, pp. 212–231, Jun. 2014.

[37] N. Laoutaris, H. Che, and I. Stavrakakis, « The LCD interconnection of LRU caches and its analysis », *Performance Evaluation*, vol. 63, pp. 609–634, Jul. 2006.

[38] M. Garetto, E. Leonardi, and V. Martina, « A Unified Approach to the Performance Analysis of Caching Systems », *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 1, pp. 1–20, May 2016.

[39] N. Gast and B. V. Houdt, « TTL approximations of the cache replacement algorithms LRU(m) and h-LRU », *Performance Evaluation*, vol. 117, pp. 33–57, Sep. 2017.

[40] E. J. O'Neil, P. E. O'Neil, and G. Weikum, « The LRU-K Page Replacement Algorithm for Database Disk Buffering », *ACM SIGMOD Record*, vol. 22, pp. 297–306, Jun. 1993.

[41] T. Johnson and D. Shasha, « 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm », *in Proceedings of the 20th International Conference on Very Large Data Bases*, Apr. 1994, pp. 439–450.

[42] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib, « A Survey on Replica Server Placement Algorithms for Content Delivery Networks », *IEEE Communications Surveys Tutorials*, vol. 19, pp. 1002–1026, Nov. 2017.

[43] P. Krishnan, D. Raz, and Y. Shavitt, « The Cache Location Problem », *IEEE/ACM Transactions on Networking*, pp. 568–582, Oct. 2000.

[44] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, « On the optimization of storage capacity allocation for content distribution », *Computer Networks*, vol. 47, pp. 409–428, Oct. 2004.

[45] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, « Joint object placement and node dimensioning for Internet content distribution », *Information Processing Letters*, vol. 89, pp. 273–279, Mar. 2004.

[46] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, « Constrained mirror placement on the Internet », *in Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, Apr. 2001, pp. 31–40.

[47] H. Yin, X. Zhang, T. Zhan, Y. Zhang, G. Min, and D. O. Wu, « NetClust: A Framework for Scalable and Pareto-Optimal Media Server Placement », *IEEE Transactions on Multimedia*, vol. 15, pp. 2114–2124, Dec. 2013.

[48] G. Rodolakis, S. Siachalou, and L. Georgiadis, « Replicated Server Placement with QoS Constraints », *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, pp. 1151–1162, Oct. 2006.

[49] F. Chen, K. Guo, J. Lin, and T. L. Porta, « Intra-cloud lightning: Building CDNs in the cloud », *in IEEE INFOCOM*, Mar. 2012, pp. 433–441.

[50] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, « Optimal VNFs Placement in CDN Slicing Over Multi-Cloud Environment », *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 616–627, Mar. 2018.

[51] D. Rossi and G. Rossini, « On sizing CCN content stores by exploiting topological information », *in IEEE INFOCOM Workshops*, Mar. 2012, pp. 280–285.

[52] I. Psaras, W. K. Chai, and G. Pavlou, « In-Network Cache Management and Resource Allocation for Information-Centric Networks », *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 2920–2931, Nov. 2014.

[53] S. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, « Less Pain, Most of the Gain: Incrementally Deployable ICN », *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 147–158, Aug. 2013.

[54] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, « Design and Evaluation of the Optimal Cache Allocation for Content-Centric Networking », *IEEE Transactions on Computers*, vol. 65, pp. 95–107, Jan. 2016.

[55] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, « Impact of traffic mix on caching performance in a content-centric network », *in Proceedings IEEE INFOCOM Workshops*, Mar. 2012, pp. 310–315.

[56] M. Chrobak and J. Noga, « LRU is Better than FIFO », *Algorithmica*, vol. 23, pp. 180–185, Feb. 1999.

[57] E. Gelenbe, « A Unified Approach to the Evaluation of a Class of Replacement Algorithms », *IEEE Transactions on Computers*, vol. C-22, pp. 611–618, Jun. 1973.

[58] S. Podlipnig and L. Böszörmenyi, « A Survey of Web Cache Replacement Strategies », *ACM Computing Surveys*, vol. 35, pp. 374–398, Dec. 2003.

[59] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, *OSPFN: An OSPF based routing protocol for Named Data Networking*, Technical Report NDN-0003, Jul. 2012.

[60] R. Chiocchetti, D. Rossi, and G. Rossini, « ccnSim: An highly scalable CCN simulator », *in IEEE International Conference on Communications*, Jun. 2013, pp. 2309–2314.

[61] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, « Nonlinear integer programming », *in 50 Years of Integer Programming 1958-2008*, Nov. 2010, pp. 561–618.

[62] T. A. Feo and M. Resende, « Greedy Randomized Adaptive Search Procedures », *Journal of global optimization*, vol. 6, pp. 109–133, Mar. 1995.

[63] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, « Toward an SDN-enabled NFV architecture », *IEEE Communications Magazine*, vol. 53, pp. 187–193, Apr. 2015.

[64] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, « Network function virtualization: Challenges and opportunities for innovations », *IEEE Communications Magazine*, vol. 53, pp. 90–97, Feb. 2015.

[65] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, « What Will 5G Be? », *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 1065–1082, Jun. 2014.

[66] D. R. Page, « Generalized Algorithm for Restricted Weak Composition Generation », *Journal of Mathematical Modelling and Algorithms in Operations Research*, vol. 12, pp. 345–372, Dec. 2013.

[67] R. Marti, V. Campos, M. Resende, and A. Duarte, « Multiobjective GRASP with Path Relinking », *European Journal of Operational Research*, vol. 240, pp. 54–71, Jul. 2014.

[68] Y. Zhou, L. Chen, C. Yang, and D. M. Chiu, « Video Popularity Dynamics and Its Implication for Replication. », *IEEE Transactions on Multimedia*, vol. 17, pp. 1273–1285, Aug. 2015.

[69] P. Bertin, T. Mamouni, and S. Gosselin, « Next-Generation PoP with Functional Convergence Redistributions », *in Fiber-Wireless Convergence in Next-Generation Communication Networks: Systems, Architectures, and Management*. Jan. 2017, pp. 319–336.

[70] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, « A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions », *IEEE Communications Surveys Tutorials*, Apr. 2018.

[71] G. Alfano, M. Garetto, and E. Leonardi, « Content-centric wireless networks with limited buffers: When mobility hurts », *IEEE/ACM Transactions on Networking*, vol. 24, pp. 299–311, Oct. 2016.

[72] D. Liu, B. Chen, C. Yang, and A. Molisch, « Caching at the wireless edge: design aspects, challenges, and future directions », *IEEE Communications Magazine*, vol. 54, pp. 22–28, Sep. 2016.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Sur des modèles pour l'évaluation de performance et le placement des ressources de cache dans les réseaux multi-cache.

Résumé : Au cours des dernières années, les fournisseurs de contenu ont connu une forte augmentation des demandes de contenus vidéo et de services riches en média. Compte tenu des limites de la mise à l'échelle du réseau et au-delà des réseaux de diffusion de contenu, les fournisseurs de services Internet développent leurs propres systèmes de mise en cache afin d'améliorer la performance du réseau. Ces facteurs expliquent l'enthousiasme à l'égard du concept de réseau centré sur le contenu et de sa fonction de mise en cache en réseau. La quantification analytique de la performance de la mise en cache n'est toutefois pas suffisamment explorée dans la littérature. De plus, la mise en place d'un système de *caching* efficace au sein d'une infrastructure réseau est très complexe et demeure une problématique ouverte. Pour traiter ces questions, nous présentons d'abord dans cette thèse un modèle générique et précis de cache nommé MACS (Markov chain-based Approximation of Caching Systems) qui peut être adapté très facilement pour représenter différents schémas de mise en cache et qui peut être utilisé pour calculer différentes mesures de performance des réseaux multi-cache. Nous avons ensuite abordé le problème de l'allocation des ressources de cache dans les réseaux avec capacité de *caching*. Moyennant notre outil analytique MACS, nous présentons une approche permettant de résoudre le compromis entre différentes mesures de performance en utilisant l'optimisation multi-objectif et nous proposons une adaptation de la métaheuristique GRASP pour résoudre le problème d'optimisation.

Mots clés : Réseau centré sur le contenu, Mise en cache, Chaînes de Markov, Allocation de caches, Optimization multi-objectif, GRASP.

Title: On models for performance evaluation and cache resources placement in multi-cache networks.

Abstract: In the last few years, Content Providers (CPs) have experienced a high increase in requests for video contents and rich media services. In view of the network scaling limitations and beyond Content Delivery Networks (CDNs), Internet Service Providers (ISPs) are developing their own caching systems in order to improve the network performance. These factors explain the enthusiasm around the Content-Centric Networking (CCN) concept and its in-network caching feature. The analytical quantification of caching performance is, however, not sufficiently explored in the literature. Moreover, setting up an efficient caching system within a network infrastructure is very complex and remains an open problem. To address these issues, we provide first in this thesis a fairly generic and accurate model of caching nodes named MACS (Markov chain-based Approximation of Caching Systems) that can be adapted very easily to represent different caching schemes and which can be used to compute different performance metrics of multi-cache networks. We tackled after that the problem of cache resources allocation in cache-enabled networks. By means of our analytical tool MACS, we present an approach that solves the trade-off between different performance metrics using multi-objective optimization and we propose an adaptation of the metaheuristic GRASP to solve the optimization problem.

Keywords: Content-Centric Networking, Caching, Markov chains, Cache allocation, Multi-objective optimization, GRASP.