Popularity-Based Caching Strategies for Content Centric Networking

César Bernardini

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Stratégies de Cache basées sur la popularité pour Content Centric Networking

THÈSE

présentée et soutenue publiquement le 05/05/15

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

César Bernardini

Composition du jury

Président :	Le président	
Rapporteurs :	Dr. Toufik AHMED	Professeur, Université Bordeaux 1, France
	Dr. Dario ROSSI	Professeur, TELECOM ParisTech, France
Examinateurs :	Dr. Guy LEDUC	Professeur, Université de Liège, Belgique
	Dr. Olivier Perrin	Professeur, Université de Lorraine, France
	Dr. Thomas SILVERSTON	Maître de conférences, Université de Lorraine, France
	Dr. Olivier FESTOR	Professeur, Université de Lorraine, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Acknowledgments

The sky is grey and a cold breeze blows while my coffee cool down slowly. The weather invites to reckon and a thought of an ending story runs over and over into my head. This story characterized by hours and hours of work. Hours of work motivated by the support of great people, who have encouraged me to go further and pursuit my dreams.

In all the years, I have spent an unmeasurable time in the laboratory with several scientific, who try to push the science to their limits. From all of them, I would like to extend my gratitude to my supervisors *Prof. Olivier Festor* and *A. Prof. Thomas Silverston* for the hours spent reviewing the manuscript and several discussions. Both of them dealt with the complicated task of organizing my way of thinking. Next, I would like to thank to my collegues of the Madynes team for the encouragement and help in all the matters of life: *Samuel*, *Olivier*, *Rodney*, *Elian*, *Said*, *Anthéa*, *Gaëtan*, *Juan Pablo*, *Martín*, *Eric*, *Kevin*, *Wazen*, among many invaluable others. A special line for the *Eng. Francois Despaux* for listening, brain-storming and supporting me during this three years.

Outside the laboratory, I have been fortunate for making other incredible friends. I am grateful to my Argentinian friends who went to visit me and all the others who waited there for me and remind me how much I love home. I am thankful to my friends of the *Sport Nautique de Nancy* for teaching me the art of rowing and spending amazing time together. I would like to thank my french roommates and specially thank to *Charlotte* for becoming a sister to me.

Without my family support, I will not be able to stay all these years in Europe. Thus, I would like to thank to my parents, *Susana* and *Pedro*, for all the encouragement and the calls on Saturday and Sunday afternoons. Next, my brothers, *Pablo* and *Javier*, for taking care of the family in my absence and forgive me the absence in extremely important events such as the mariage of *Pablo* and *Laura* and the born of *José Ignacio*.

All these would not be possible without the continuous and unlimited support of *Nadège*. Je te remercie infiniment.

It is with a lot of pleasure that I will remind all these rainy and cold years in Nancy with uncountable meetings and stories.

Gracias.

Contents

Résumé de la thèse **3**

1 Introduction, Contexte et Motivation . . . 3

1.1 Réseaux Orientés au Contenu . . . 4

1.2 Stratégies de Cache . . . 5

2 Contributions . . . 6

2.1 Environment Commun de Simulation . . . 6

2.2 Stratégie basée sur le contenu . . . 9

2.3 Stratégie basée sur l'information sociale . . . 10

3 Organisation de la thèse . . . 12

Chapter 1 General Introduction **13**

1.1 Problem Statement . . . 14

1.2 Manuscript Organization . . . 15

State of the Art **17**

Chapter 2 Information and Content Centric Networks **19**

2.1 Information Centric Networks . . . 20

2.1.1 ICN Features . . . 20

2.2 ICN Architectures . . . 23

2.2.1 Content Centric Networks . . . 24

2.2.2 FP-7 Pursuit . . . 25

2.2.3 SAIL NetInf . . . 27

2.2.4 Other Architectures . . . 28

2.2.5 Comparison of Information Centric Networks architectures . . . 28

2.3 Choosing the ICN architecture: Content Centric Networking . . . 30

2.4 Summary . . . 32

Chapter 3 Caching in Content Centric Networking **35**
3.1 Caching in Content Centric Networking 36
3.1.1 Limitations of Caching in the Current Internet 36
3.1.2 Caching Properties of Content Centric Networks 37
3.1.3 Improving Performance of Content Centric Networking by means of Caching 38
3.1.4 Other Research Challenges on Caching in Content Centric Networking 39
3.2 Replacement Policies or Caching Strategies 42
3.2.1 Replacement Policies 42
3.2.2 Caching Strategies 43
3.3 Caching Strategies 43
3.4 Caching Metrics 48
3.4.1 Caching Performance Metrics 48
3.4.2 Traffic 51
3.5 Summary 53

Caching Strategies for Content Centric Networks **55**

Chapter 4 Comparing CCN Caching Strategies **57**
4.1 Survey of the Simulation Environments for Caching Strategies in CCN 58
4.1.1 Related Work 58
4.1.2 The Need for a Common Evaluation Scenario 60
4.2 Common Evaluation Scenario 61
4.3 Comparison of Caching Strategies 63
4.3.1 Simulation Environment 63
4.3.2 Results 64
4.3.3 Summary 67
4.4 Summary 68

Chapter 5 Popularity-based Caching Strategy for CCN **69**
5.1 Most Popular Caching (MPC) Strategy 70
5.1.1 MPC Example Scenario 70
5.2 Experimental results and analyses 71
5.2.1 Simulation Environment 71
5.2.2 MPC parameters tuning 72
5.2.3 MPC vs. LCE 74
5.3 Discussion 76

5.4 Summary . . . 77

Chapter 6 Socially-Aware Caching Strategy for CCN **79**
6.1 Introduction . . . 79
6.2 Social Network Model . . . 80
6.2.1 Model . . . 81
6.2.2 SONETOR: Social Network Traffic Generator . . . 82
6.3 SACS: Socially-Aware Caching Strategy . . . 83
6.4 Experimental results and analyses . . . 85
6.4.1 Simulation Environment . . . 85
6.4.2 Experiments on the Inet Topology . . . 86
6.4.3 Experiments on the common evaluation scenario . . . 88
6.4.4 Experiments on PlanetLab . . . 89
6.5 Discussion . . . 91
6.6 Summary . . . 92

Chapter 7 General Conclusion **93**
7.1 Contributions summary . . . 94
7.1.1 Fair-Comparison of the State of the Art Caching Strategies . . . 94
7.1.2 Popularity-Based Caching Strategies . . . 94
7.2 Perspectives . . . 95
7.2.1 Comparing against current Internet methods . . . 95
7.2.2 Model of Social Networks . . . 95
7.2.3 Popularity-Based Strategies . . . 96
7.2.4 Keep digging into social network information . . . 96
7.3 List of Publications . . . 97

Appendixs **99**

Appendix A Impact of OSN into Content Popularity Model **99**
A.1 Simulation Parameters . . . 99
A.2 Stand-alone Scenario with OSN Traffic . . . 101
A.3 Mixed Scenario with OSN and Regular Traffic . . . 101
A.4 Conclusion . . . 103

Appendix B Characterization of Publications in Pinterest **105**
B.1 Pinterest Overview . . . 106
B.2 Measurement Methodology . . . 106
B.2.1 Data Collection . . . 106

B.2.2 High Level Characteristics of Pinterest Dataset 107
B.2.3 Who publishes in Pinterest? 107
B.3 Study of Users' Publications 108
B.3.1 Activities: Users' Publications 109
B.3.2 Links to external websites 110
B.3.3 Filesize of Images 111
B.3.4 Popularity of Images 111
B.3.5 Classification of the Images 113
B.3.6 Lexical analysis on Pins' description 113
B.4 Related Work 114
B.5 Conclusion 115

Glossary **117**

Bibliography **119**
1 Bibliography 119
2 Webography 127

List of Figures

1 Les topologies des fournisseurs d'Internet. . . . 6
2 Comparison des Stratégies de Cache . . . 7
3 Évaluation de MPC. . . . 11
4 Évaluation de *SACS* dans le scénario commun d'évaluation. . . . 12

2.1 CCN Architecture, extracted from [20] . . . 25
2.2 The PURSUIT Architecture . . . 26
2.3 SAIL NetInf Architecture . . . 27
2.4 Selection of ICN architectures in the community . . . 32

3.1 Leave Copy Everywhere (LCE) Caching Strategy . . . 43
3.2 Leave Copy Down (LCD) Caching Strategy . . . 44
3.3 MAGIC Caching Strategy . . . 45
3.4 ProbCache Caching Strategy . . . 46
3.5 Cache "Less" for More. . . . 47
3.6 Example of CCN request to illustrate the metrics. . . . 49
3.7 Inter-AS Traffic . . . 52

4.1 Simulation environments in the literature. . . . 61
4.2 ISP-level topologies. . . . 62
4.3 Simulator architecture . . . 63
4.4 Comparison of the Caching Strategies . . . 65

5.1 MPC Workflow example. . . . 71
5.2 MPC parameters tuning: Simulation Time . . . 73
5.3 Tuning of Popularity Threshold (5.3a and 5.3b). . . . 74
5.4 MPC Parameters Tunning . . . 74
5.5 Evaluation of MPC with a countour plot in a small scenario. . . . 75
5.6 Evaluation of MPC with a countour plot in a Youtube-like scenario. . . . 76

6.1 Social Network Model. . . . 81
6.2 Users' interaction model. Users can perform several activities within sessions. . . . 82
6.3 SONETOR Workflow. . . . 83
6.4 Format of Sonetor traces . . . 83
6.5 Example of a social network on top of a CCN architecture. . . . 84
6.6 Evaluation of *SACS* in *Inet* Evaluation Scenario. . . . 87
6.7 Evaluation of *SACS* in Common Evaluation Scenario. . . . 89
6.8 Location of PlanetLab nodes. . . . 90

6.9 Evaluation of SACS in PlanetLab. 90

A.1 Stand-alone Scenario: impact of the OSN on the content popularity model 100
A.2 Mixed Scenario: impact of the OSN on the content popularity model 103

B.1 Probability distribution for pins, followers and followees (CCDF). 108
B.2 Pinterest's Pinners. 108
B.3 Model of users' activities in Pinterest. 110
B.4 Most linked domains on the pins. 111
B.5 Study on Pinterest images: Filesize (B.5a), Popularity (B.5b) and Categories (B.5c). 112
B.6 Lexical analysis of pins description. 114

Résumé de la thèse

1 Introduction, Contexte et Motivation

Aujourd'hui, les américains regardent environ 15 minutes de vidéos en ligne chaque jour. Ceci représente une consommation mensuelle de 16 Giga-octets par utilisateur [8]. Soit 66% du trafic [7]. Smartphones, tablettes, consoles de jeux vidéos et Smart Tv sont de plus en plus utilisés. Au vu de la tendance actuelle, la consommation de vidéos en ligne devrait être doublée en 2017. Les appareils de télévision connectés à l'Internet vont être multipliés par quatre, 100 heures de vidéo vont être ajoutés chaque minute sur Youtube. Considérant le volume de données représentant l'ensemble des films ayant été produits jusqu'à présent. Ce volume va en 2016 traverser l'Internet toutes les 3 minutes [7]. En consequence, les 15 minutes de vidéo regardées quotidiennement en ligne vont se transformer en heures de comsomation[1].

Cela engendrera des problèmes considérables sur l'Internet actuel. Actuellement, il est construit dans un paradigme de communication qui n'est pas conçu pour la transmission des vidéos. Contrairement aux réseaux de diffusion qui envoient un titre à des millions de personnes simultanément, L'Internet actuel retransmet les mêmes videos à chaque demande. Or comme 10% du contenu représente 90% du trafic de l'Internet, la congestion du réseau est inévitable. De nouvelles solutions sont donc nécessaires pour réussir à satisfaire la demande.

Dans ce contexte, des serveurs temporaires de stockage ont été utilisés dans tout l'Internet pour servir le même contenu demandé plusieurs fois. Ces serveurs temporaires de stockage s'appellent systèmes de cache, où simplement, caches. Dans cette tendance, Content Centric Networking (CCN) est apparu comme une nouvelle architecture pour l'Internet de l'avenir. En CCN, les caches seront disponibles dans chaque nœud et le réseau deviendra un réseau des caches.

Les caches n'ont jamais été deployés à une telle échelle. En mettant des caches dans chaque nœud, le problème de caches devient très complexe: ils nécessitent une augmentation de l'espace disponible pour sauvegarder le contenu. Pour gérer les caches, des stratégies de cache sont utilisées. Les stratégies de cache décident quel contenu sera sauvegardé et où. Dans la littérature, plusieurs stratégies de cache ont été proposées comme Leave Copy Everywhere [62], Leave Copy Down [93], MAGIC [70], ProbCache [53] and Cache "Less" For More [54]. Cependant, il n'est pas évident de savoir quelle stratègie de cache devrait être mise en œuvre dans quel scenario. De nouvelles solutions peuvent être proposées pour construire des stratégies de cache qui améliorent la performance du réseau.

Les principales contributions de la thèse sont les suivantes:

- Les stratégies des caches de la littérature ont été résumées. Elles ont été comparées dans un environnement commun de simulation. Leave Copy Everywhere, Leave Copy Down, MAGIC, ProbCache et Cache "Less" For More. Ainsi, le choix de la stratégie de cache adaptée à chaque scenario est possible.

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

- Une stratégie de cache basée sur la popularité de contenu a été proposée: Most Popular Caching Strategy (MPC). MPC cache seulement le contenu populaire. En suprimmant le contenu impopulaire des caches, MPC garde des ressources pour des contenus qui vont être demandés de façon plus probable statistiquement.

- Une stratégie de cache basée sur l'information des réseaux sociaux: Socially-Aware Caching Strategy (SACS). SACS détecte les utilisateurs les plus importants des réseaux en utilisant l'information des réseaux sociaux. SACS cache de façon proactive le contenu produit par les utilisateurs les plus importants du réseau. Ainsi, SACS dépasse les autres stratégies de cache quand les utilisateurs interagissent dans un réseau social.

1.1 Réseaux Orientés au Contenu

Les Réseaux Orientés Contenu ou Content Centric Networking (CCN) représentent une nouvelle architecture pour l'Internet du futur. CCN prétend reconstruire la pile de protocoles de l'Internet, en suprimmant la pile TCP/IP.

L'architecture CCN est principalement basée sur deux primitives: *Interest* et *Data*. Un consommateur demande un contenu en envoyant un message *Interest* dans le réseau. Un nœud recevant cette requête et possédant le contenu peut y répondre avec un message *Data*. Le contenu sera alors transmis au demandeur et chaque nœud du chemin sur le réseau pourra conserver en mémoire cache les données.

Tous les messages sont redirigés par des routeurs de contenu (CR) en utilisant un mécanisme anycast. Un CR maintient trois structures de données: *Forwarding Information Base* (FIB), *Pending Interest Table* (PIT) and *Content Store* (CS). La FIB recense les noms du contenu avec les interfaces de sortie. Elle est utilisée pour transmettre les messages *Interest* vers ses destinations. La PIT recense les interfaces d'entrée par lesquelles les messages d'Interest sont arrivés. Lorsque le message *Data* arrive avec le contenu, la PIT est utilisée pour renvoyer le contenu vers le nœud émetteur. Finalement, le CS sauvegarde le contenu qui est passé par le CR. Le CS est utilisé comme une mémoire cache pour les rêquetes futures.

Quand un message d'*Interest* arrive, le CR extrait le nom du contenu demandé. Le CR vérifie dans son CS si il y a un contenu avec ce nom. Si le contenu est trouvé, un message Data avec le contenu est envoyé vers l'interface d'où le *Interest* est arrivé. Si le contenu n'est pas trouvé, une détermination du préfixe de correspondance le plus long est faite avec le nom dans la table PIT. Cette vérification permet de savoir si il y a déjà une réponse attendue pour une réquête avec le même nom. Si une entrée est trouvée, le message d'*Interest* est supprimé. Si aucune entrée n'est trouvée, le CR vérifie dans sa table FIB par quelle interface transmettre le message d'*Interest*.

L'architecture actuelle d'Internet contient des ressources limitées pour le cache et ces serveurs de cache sont placés à des endroits fixes. En revanche, CCN est un réseau de caches avec des serveurs des caches localisés partout. En plus, CCN est un réseau conscient du contenu qui le traverse (content-aware).

Ces deux caractéristiques impliquent l'apparition de nouvelles propriétés :

- *Transparence du cache.* Avec la transparence des caches, les utilisateurs n'ont pas connaissance que le contenu provient de caches. Ainsi, les utilisateurs reçoivent le contenu de la part du nœud localisé au plus près d'eux, car les ressources sont archivées par chaque nœud.

- *Ubiquité du cache.* Les ressources de cache dans l'actuelle architecture de l'Internet sont placées dans des endroits fixes [14]. La localisation du contenu est déterminée en résolvant

un modèle analytique établi lors de précédentes demandes sur ce même réseau. Dans CCN, les caches sont omniprésents et disponibles partout. Leur localisation n'est plus figée. Le contenu de ces caches doit évoluer lorsque des décisions sont prises de manière dynamique et adaptées à l'environnement.

- *Granularité fine des contenus en cache.* L'unité de paquet dans CCN est le bloc. Un contenu est composé de plusieurs blocs et la même unité est utilisée dans le réseau CCN. Comme tous les paquets sont nommés, toutes les blocs de contenu sont nommées. Le changement d'unité par rapport aux réseaux IP (paquet) fait apparaître le problème suivant. Dans l'Internet il n y a pas de schéma commun de nommage des contenus. Si le même contenu est transmis avec différents protocoles les fonctionalités de cache ne peuvent pas être partagées. Comme il n'y a pas un schéma commun de nommage des contenus, il n'est pas possible de déterminer si deux paquets transmettent le même contenu. CCN est créé avec un schéma commun de nommage des contenus. Le contenu transmis via différents protocoles partage le même nom. Ainsi, différentes logiciels et protocols peuvent utiliser des ressources de cache pour fournir le contenu démandé avec un autre protocole que le protocol d'origine. Cette nouvelle caractéristique permet de reduire le temps pour récuperer un contenu et augmenter la performance des caches.

CCN étant un réseau de cache, l'organisation individuelle des caches devient essentielle. Des stratégies de cache sont necessaires pour organiser et gérer la collection de ressources disponibles dans un réseau CCN. Dans cette thèse, nous elaborons des solutions pour améliorer l'efficacité des caches. Nous soutenons que meilleure efficacité des caches va entraîner un meilleur comportement de l'ensemble du réseau.

1.2 Stratégies de Cache

Les stratégies de cache sont des politiques de gestion. Elles décident quelle information sauvegarder et à quel endroit, satifaisant ainsi les objectives globaux du réseau. Ces décisions incluent un large éventail de possibilités, comme le placement du contenu, la détection de patrons de trafic, ou encore la selection du type d'information.

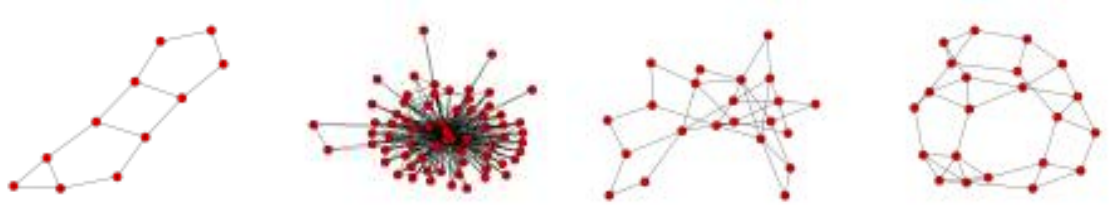
Les stratégies de cache fournissent des mécanismes efficaces pour être en mesure de faire face au trafic du réseau fortement lié à la distribution de contenu à l'échelle globale. Les stratégies de cache utilisent les mémoires de cache pour absorber du traffic en faisant des copies des contenus populaires dans des endroits stratégiques. En reduisant le trafic, on s'attend à une meilleure qualité de service et à pouvoir libérer des ressources du réseau qui pourront être déstinées aux autres activités du réseau.

Les principales stratégies de cache de la littérature sont résumées dans la liste suivante:

- *Leave Copy Everywhere (LCE).* Elle consiste à sauvegarder tous les messages de contenu qui passent dans le réseau. Cette stratégie est la plus utilisée et notamment dans CCN [62].

- *Leave Copy Down (LCD)* [93]. Quand un message d'Interest est trouvé dans un cache, la stratégie de cache copie le contenu dans le cache du voisin vers la réquête originale.

- *Max-Gain In-Network Caching (MAGIC)* [70]. Cette stratégie de cache cherche à maximiser le bénéfice en mettant un contenu dans un cache, et en même temps à minimiser la perte en replaçant un autre contenu. Les maximisation et minimisation sont calculées avec les informations de popularité.



(a) Abilene (b) DTelecom (c) GEANT (d) Tiger

Figure 1: Les topologies des fournisseurs d'Internet.

- *ProbCache* [53]. Probabilistic In-Network CAching (ProbCache) est une stratégie basée sur la probabilité. Elle distribue le contenu dans un chemin de caches. Avec une loi de probabilité, ProbCache sélectionne quel est le meilleur nœud pour sauvegarder un contenu.
- *Cache "Less" For More* [54]. Cache "Less" For More est une stratégie de cache basée sur une métrique de centralité qui utilise le concept de *betweenness centrality* pour améliorer le bénéfice d'introduire un nouvel élément. De plus, elle réduit la redondance par rapport au LCE.

Toutes ces stratégies de cache ont des objectifs différents et elles utilisent différent types d'informations pour réagir. Cache "Less" For More cherche à obtenir des résultats similaires à LCE tandis qu' elle reduit le nombre des opérations de remplacement de contenu dans les caches. MAGIC cherche à maximiser l'efficacité de l'ajout d'un nouveau contenu dans les caches en utilisant la métrique du *Cache Hit*. Par rapport aux types d'informations, diverses alternatives ont été prises en compte dans la littérature. Cache "Less" For More utilise des informations de topologie lorsque MAGIC et ProbCache utilisent des informations de popularité. Cependant parmi toutes ces stratégies, ce n'est pas évident d'identifier laquelle fonctionne le mieux. Nous affirmons que l'on peut construire des mécanismes alternatifs pour des scénarios qui n'ont encore jamais été considérés. Pour réussir cet objectif, nous proposons un framework commun. Dans ce framework, nous comparons toutes les stratégies de cache et décidons quelles sont les meilleures dans chaque situation.

2 Contributions

Dans cette thèse, nous avons elaboré des nouvelles stratégies de cache pour CCN. Plus précisément, nous présentons un framework d'évaluation des stratégies de cache de l'état de l'art et nous dévélopons deux nouvelles stratégies de cache basées sur la popularité. La première contribution traite le framework d'évaluation des stratégies de cache. Dans le framework, les stratégies de cache sont comparées entre elles. La deuxième contribution est une stratégie de cache basée sur la popularité de contenu. Finalement, notre dernière contribution est une stratégie de cache basée sur la popularité des utilisateurs.

2.1 Environment Commun de Simulation

Un scénario commun d'évaluation a été défini. Il utilise les paramètres les plus utilisés dans la littérature.

Par rapport à la topologie, l'architecture de l'Internet du futur est construite sur une architecture CCN. Cette nouvelle architecture de l'Internet sera organisée avec des fournisseurs

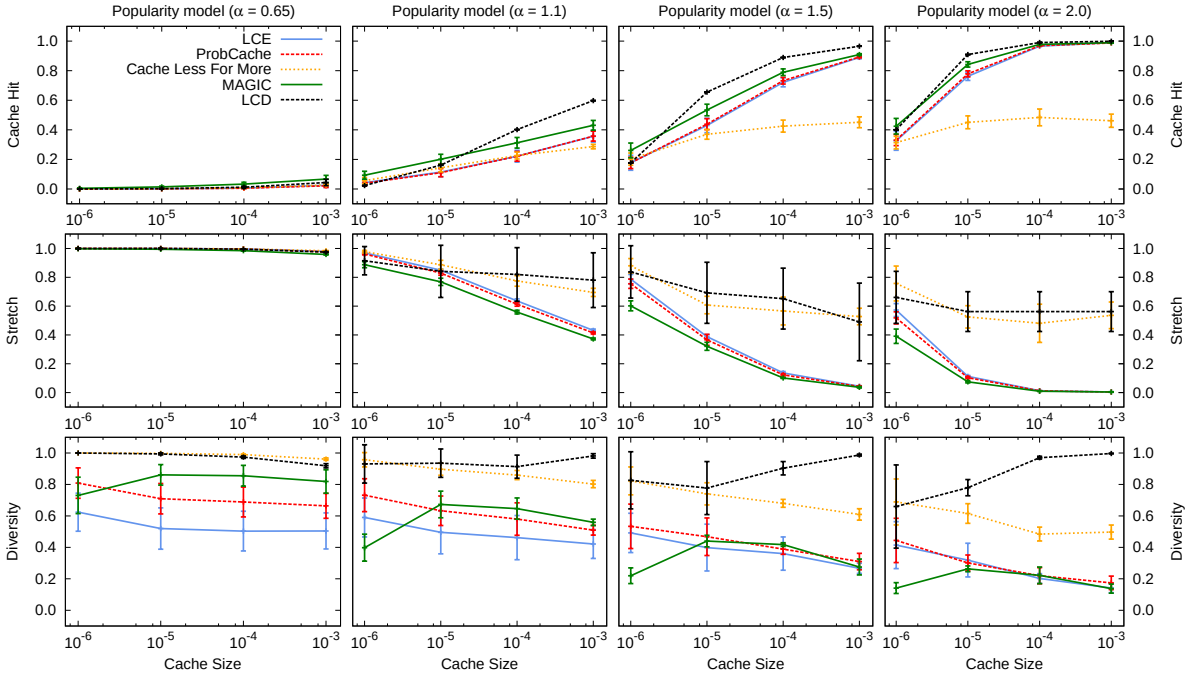


Figure 2: Comparison des Stratégies de Cache

Parameters	
Catalog	10^6
Popularity Model	MZipf($\alpha = \{0.65; 1.1; 1.5; 2.0\}, \beta = 0$)
Topologies	Geant, Dtelecom, Abilene, Tiger
Cache ratio	$\{10^{-6}; 10^{-5}; 10^{-4}; 10^{-3}\}$

Table 1: Scénario commun d'évaluation

Internet et son infrastructure sera probablement proche de l'actuelle. Les topologies de fournisseurs d'Internet sont donc les meilleures alternatives pour évaluer les stratégies. De toutes les topologies disponibles, quatre topologies de fournisseurs d'Internet: Abilene, DTelecom, GEANT et Tiger. Elles sont présentées dans le Figure 1 [62].

Par rapport au modèle de popularité de contenu, des modèles de popularité basés sur la loi de probabilité MZipf ont été utilisés. Le paramètre α de MZipf varie dans la littérature parmi les valeurs 0.65, 1.1, 1.5 et 2.0. La valeur 0.65 se réfère à un scénario avec très peu de contenus populaires et ce scénario ressemble à une loi de probabilité uniforme.. La probabilité de choisir un contenu est similaire à une loi de probabilité uniforme. La valeur 1.1 se réfère à un scénario avec de la popularité normale. Quand nous parlons de popularité normale, nous nous référons à une loi MZipf sur laquelle 90% de requêtes démandent le 60% des contenus. Finalement, deux autres scénarios avec une forte popularité égale à 1.5 et 2.0 sont considérés. Dans la reste de cette thèse, ces scénarios sont référencés comme scénarios de *base, normal et à haute popularité.*

Par rapport à la taille du catalogue de contenu, un catalogue de 10^4 contenus a été le modèle le plus utilisé dans la littérature. Cependant, ce volume est très petit et il représente un catalogue de contenus qui ne ressemble pas à la taille du catalogue de contenus de l'Internet. Un catalogue de 10^6 contenus a donc été considéré. La taille du cache va changer de 1 jusqu'à 1000 contenus; ce qui correspond à une facteur de reduction de 10^{-6} jusqu'à 10^{-3} par rapport au catalogue. Les

tailles de caches sont considérées comme uniformes (tous les nœuds ont la même taille de cache). La politique de remplacement (CRP) utilisée dans la comparaison est Last Recently Used (LRU). Rosenweig et al [57] montrent que les CRPs peuvent être groupées dans des groupes d'équivalence, i.e., différentes CRPs vont exhibir la même performance. Tous les paramètres de notre scénario d'évaluation sont résumés dans la Table 1.

Metriques d'Évaluation

Les stratégies de cache vont être comparées entre elles suivant les métriques suivantes: *Cache Hit*, *Stretch* et *Diversity*.

Le *Cache Hit* est la probabilité qu'un contenu soit présent dans un cache le long du chemin. Pour chaque contenu trouvé dans un cache on considère une opération de *hit* ou sinon, une opération de *miss*. Les stratégies sont désignées pour augmenter le nombre d'opérations de *hit*. Le *Cache Hit* est detaillé dans l'équation 1 : $hits_i$ and $miss_i$ se réfèrent au numéro de contenus démandés par messages d'*Interest* trouvés ou non dans les caches.

$$CacheHit = \frac{\sum_{i=1}^{|N|} hits_i}{(\sum_{i=1}^{|N|} hits_i) + \sum_{i=1}^{|N|} miss_i} \tag{1}$$

Le *Stretch* est aussi une métrique très populaire pour l'évaluation des stratégies de cache [67, 53, 54, 58, 62, 70]. Le Stretch se réfère au pourcentage du chemin traversé pour récupérer un contenu. Le $hops\_walked_i$ décrit le nombre de sauts entre le nœud qui a émis la rêquete i et le nœud qui répond à la réquête avec le contenu de son cache. $total_hops_i$ décrit le nombre de sauts depuis le nœud qui émet la réquête i jusqu'au nœud qui fournit le contenu. Si la valeur tend vers 0, les réquêtes sont satisfaites grâce à du contenu à proximité des utilisateurs et la performance du réseau est meilleure.

$$Stretch = \frac{\sum_{i=1}^{|R|} hops\_walked_i}{\sum_{i=1}^{|R|} total\_hops_i} \tag{2}$$

La *Diversité* représente la proportion de contenus différents stockés dans les caches. La *Diversité* des contenus en cache est présentée dans l'Équation 3.12. Les valeurs de la *Diversité* sont comprises entre $\frac{1}{|N|}$ et 1: Lorsque la valeur tend vers $\frac{1}{|N|}$, cela veut dire que les caches du réseau contiennent plusieurs copies des mêmes contenus; sinon (i.e. la valeur tend vers 1), cela veut dire que les caches du réseau contiennent des contenus différents.

$$Diversit = \frac{len(\bigcup_{n=1}^{|N|} C_n)}{\sum_{n=1}^{|N|} len(C_n)} \tag{3}$$

Cache Evictions compte le nombre de fois que la politique de replacement a été appelée (Équation 3.9). Dans notre thèse, nous utilisons *Ratio of Cached Elements* (Equation 3.10). Elle est définie comme la proportion relative des *Ratio of Cache Evictions* entre deux stratégies de cache.

$$Cache\ Evictions = \sum_{i=1}^{|N|} o_i \tag{4}$$

$$Ratio\ of\ Cached\ Elements = \frac{Cache\ Evictions\ for\ Strategy\ \#1}{Cache\ Evictions\ for\ Strategy\ \#2} \tag{5}$$

Évaluation

La Figure 4 montre la comparaison des stratégies de cache dans l'environment commun de simulation. La Figure est divisé en 3 lignes et 4 colonnes: chaque ligne représente une métrique différente (*Cache Hit*, *Stretch* et *Diversité*); chaque colonne montre une configuration différente de scénario de popularité (parametre α de MZipf). Dans chaque cellule, l'axe des abscisses représente les tailles du cache et les ordonnés, la metrique. Pour chaque valeur, chaque expérience a été répètée trois fois dans des topologies différentes.

Dans la première ligne de la Figure 2, nous comparons la *Cache Hit* des stratégies de cache dans le différents scénarios de popularité (α=0.65, α=1.1, α=1.5 and α=2.0). Une attention particulière doit être portée aux petits caches. Le plus grand est le cache, le plus facile d'avoir des meilleures résultats. Cependant, le but est en ayant le minimum de ressources (taille de cache) à améliorer la performance des caches. Par exemple, lorsque l'on considère un catalogue de 10^{13} contenus, un cache de 10^{-6} contenus se réfère à un cache de dix millions de contenus. Lorsque une taille de cache de 10^{-3} contenus se réfère à dix milliards de contenus.

Par rapport à la *Cache Hit*, MAGIC et LCD montrent les meilleures résultats. Dans les scénarios avec un basse diversité, MAGIC offre des meilleurs résultats avec des petites caches. Lorsque la taille des caches augmente, MAGIC est surpassée par LCD.

LCE et ProbCache montrent presque le même niveau de performances. Cependant, ProbCache est meilleure que LCE parce qu'elle invoque moins d'opérations d'appels à les politiques de remplacement.

La stratégie Cache "Less" For More a montré des résultats différents par rapport aux autres stratégies. Dans les scénarios avec une basse popularité ($\alpha = 0.65$ and $\alpha = 1.1$), notamment pour les petits caches (10^{-6} et 10^{-5}) Cache "Less" For More obtient le même niveau de performance que les autres stratégies. Cependant, avec une haute popularité, ces résultats ne sont pas bons du tout et toutes les autres stratégies de cache montrent des meilleures résultats ($\alpha = 1.5$ et $\alpha = 2.0$).

MAGIC et LCD ont montré les meilleurs résultats de performances de *Cache Hit*. Nos résultats ont montré que LCD devrait être utilisé comme remplaçant de LCE, la stratégie de cache par défaut en CCN. Si on devait choisir entre LCE et ProbCache, ProbCache est une meilleure option dû au nombre moins élevé des opérations de cache faites dans le réseau. Finalement, on considère que Cache "Less" For More est utile dans les scénarios avec popularité basse et normal et avec un cache plus petit que 10^{-5} contenus.

Le scénario commun de simulation a permis de comparer toutes les stratégies de cache de la littérature. Cependant, nous soutenons que des mécanismes alternatifs de cache peuvent être proposés. Dans cette thèse, nous proposons deux stratégies de cache basée sur la popularité des contenus et des utilisateurs du réseau.

2.2 Stratégie basée sur le contenu

MPC est une nouvelle stratégie de cache reposant sur la popularité de contenu et elle est construite pour CCN. MPC cache uniquement le contenu populaire. MPC caches moins de contenu que LCE et aussi MPC améliore les performances des cache lorsque il reduit la consomation de ressources.

Avec MPC, chaque nœud compte localement le nombre de requêtes pour chaque nom de contenu et stocke le couple *(Content Name; Popularity Count)* dans une *Table de Popularité*. Dès qu'un nom de contenu atteint localement un *Seuil de Popularité*, le contenu est marqué comme étant populaire et si le nœud contient le contenu, il suggère à ses voisins de le cacher

grâce à une nouvelle primitive de *Suggestion*. Ces messages de *Suggestion* peuvent être acceptés ou non, suivant les politiques locales de gestion des ressources. La popularité des contenus pouvant diminuer avec le temps après leur *Suggestion*, le *Compteur de popularité* est réinitialisé à une *Valeur d'Initialisation* afin de ne pas inonder les voisins avec le même contenu.

En plus de la mémoire cache pour les nœuds CCN, MPC a besoin d'espace pour stocker la *Table de popularité*. Par exemple, la conservation d'un million d'entrées dans la table représenterait 1Gigaoctet de mémoire en utilisant 1023 octets pour chaque nom de contenu et 1 octet pour le *Compteur de popularité* (nous avont utilisé des longueurs fixes pour simplifier les calculs).

Évaluation

Sur la Figure 3a, le *Cache Hit* de MPC est clairement plus important que celui de LCE et dépasse les 85%. Même quand LCE est à son plus haut niveau comme avec les topologies Level3 ou DTelecom, MPC surpasse toujours CCN.

Le *Ratio of Cached Elements* en cache est présenté sur la Figure 3b. Pour LCE avec sa stratégie de mise en cache systématique, les contenus sont toujours stockés et ce seuil atteint 100%. MPC cache beaucoup moins de contenus que LCE : environ 20% avec les topologies *Tree, Abilene, Geant et Tiger2*. MPC effectue ainsi moins d'opérations mémoire. MPC cache plus de contenus sur les topologies *DTelecom* et *Level3* (80% and 60% respectivement) mais toujours moins que LCE. Pour ces deux topologies, l'augmentation vient du fait qu'elles ont un degré de connectivité des nœuds élevé (10 ou 11 voisins dans ces topologies). Dans de telles topologies, MPC cache d'avantage en dehors des chemins de livraison de données (off-the-path) parce qu'il envoie plus de messages *Suggestion* pour stocker les contenus chez les nœuds voisins.

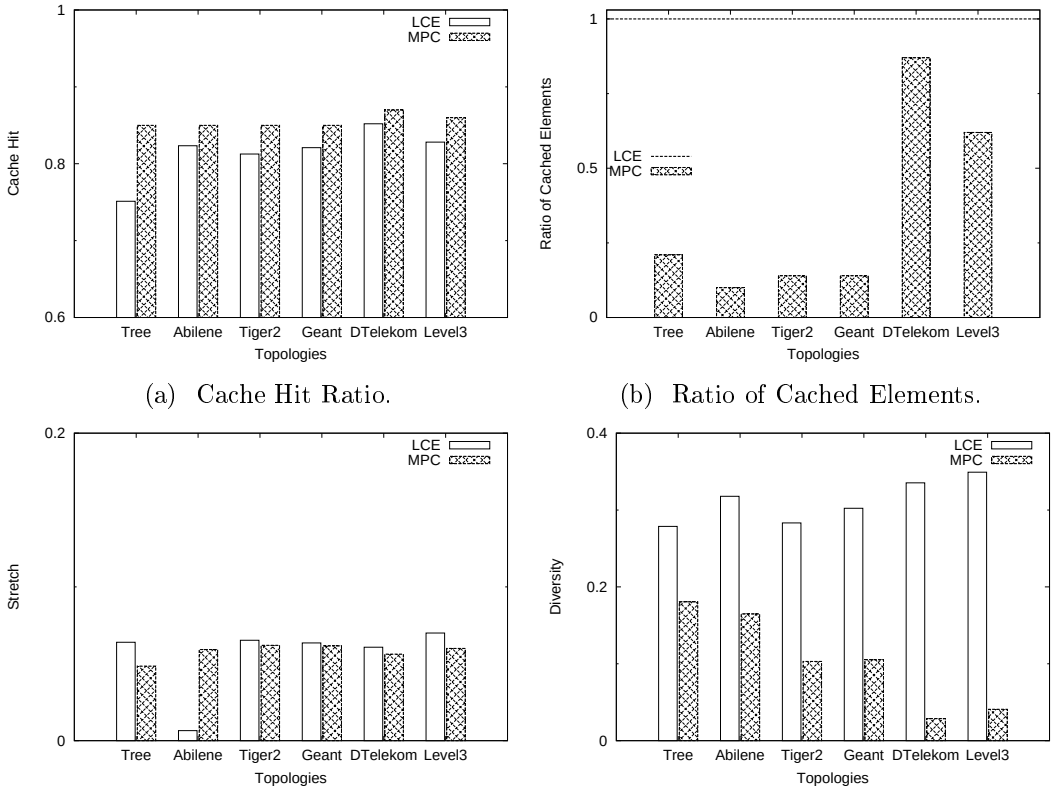
La *Diversité* des contenus en cache est présentée sur la Figure 3d. Pour LCE, la *Diversité* s'étend de 28% jusqu'à 35% pour toutes les topologies; celle de MPC est moins importante de 3% à 18%. Bien sûr, il était attendu que MPC soit moins efficace que CCN concernant la *Diversité*; puisque MPC est conçu pour ne stocker que les contenus populaires, limitant "par nature" la *Diversité* des contenus dans les caches de nœuds. Cependant, ni LCE ni MPC ne permettent d'atteindre un haut niveau de *Diversité*.

Enfin, nos expériences de simulation montrent que la stratégie MPC surpasse la stratégie LCE car elle atteint un taux supérieur de *Cache Hit*. En outre, MPC met en cache moins de contenu. Elle permet donc d'économiser des ressources telles que la mémoire et le nombre d'opérations de cache.

2.3 Stratégie basée sur l'information sociale

SACS (Socially-Aware Caching Strategy) est notre seconde nouvelle stratégie de cache conçue pour CCN et basée sur les informations sociales des utilisateurs. Cette stratégie donne priorité aux contenus qui proviennent d'utilisateurs populaires influents. Elle cache ces contenus de façon pro-active dans le réseau CCN. En effet, les utilisateurs avec un grand nombre de relations sociales (utilisateurs populaires) seront des leaders d'opinion et auront par conséquent plus d'influence que les utilisateurs avec peu de relations sociales. Les contenus produits par des utilisateurs influents seront donc certainement plus consommés que ceux des autres. Par exemple, lorsque un utilisateur populaire de Twitter envoie un message, celui-ci est beaucoup plus retransmis (*reTweet*) qu'un message d'un utilisateur ordinaire ayant peu de contacts. Cette stratégie de cache SACS est, la première pour CCN, à se baser sur les informations sociales des utilisateurs.

Avec SACS, le contenu publié par des utilisateurs influents est répliqué de façon proactive dans les plus courts chemins du réseau social vers leurs voisins, avant même que ceux-ci ne le



(a) Cache Hit Ratio.

(b) Ratio of Cached Elements.

(c) Stretch.

(d) Diversity.

Figure 3: Évaluation de MPC.

demandent. Cela augmente la disponibilité des contenus provenant d'utilisateurs influents et réduit le nombre de messages CCN dans le réseau. Ainsi, SACS privilégie le contenu provenant d'utilisateurs influents.

Les utilisateurs influents d'un réseau social sont déctetés avec l'utilisation des métriques de centralité *Pagerank* [77] ou *Eigenvector* [76]. Ces métriques permettent de calculer un score pour chacun des utilisateurs dans le réseau social en fonction de leur importance. Un utilisateur sera considéré comme important si son score est supérieur à la moyenne des scores du réseau social.

Évaluation

La Figure 4 montre la comparaison de *SACS* et LCE dans l'environment commun de simulation. La Figure 4 est divisée en 3 lignes et 4 colonnes: chaque ligne montre une métrique différente (*Cache Hit*, *Stretch* et *Diversité*); chaque colonne montre une différente configuration d' un scénario de popularité (paramètre α de MZipf). Dans chaque cellule, l'axe des abscisses représente les tailles du cache et l'axe des ordonnées, la métrique.

Dans l'aperçu de la Figure 4, il est notable que *SACS* surpasse toujours LCE au regard des métriques *Cache Hit* et *Stretch*. Elle garde un haut *Cache Hit* lorsque la *Stretch* est réduite. Cela veut dire que les caches d'un réseau CCN vont être capables de répondre à plus de requêtes et donc réduire la distance parcourue par le contenu.

Le point faible de *SACS* est sa faible diversité. Il est rare que la diversité de *SACS* atteigne 10% avec un scénario avec une popularité de α égal à 0.65. Dans tous les scénarios présentés, la

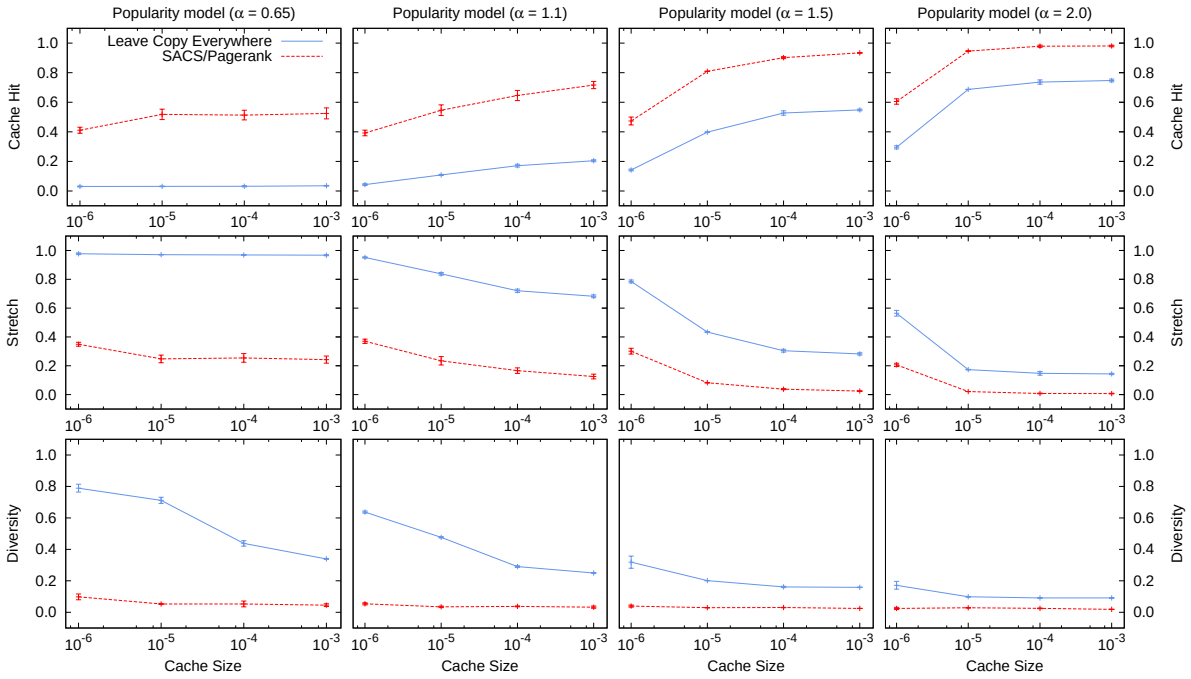


Figure 4: Évaluation de *SACS* dans le scénario commun d'évaluation.

diversité de *SACS* tend vers sa valeur minimale (i.e. $\frac{1}{N}$).

Par rapport aux topologies, la performance de *SACS* ne change pas dans les scénarios avec une popularité différente. Ce phénomène est traduit par un écart type très bas entre les résultats. Lorsque l'écart type est plus petit que 0.03 dans les différentes topologies de fournisseurs d'Internet, *SACS* devrait montrer les mêmes résultats que d'autres topologies.

3 Organisation de la thèse

Le manuscrit est composé de deux parties, suivies par une conclusion et deux annexes.

La première partie détaille les réseaux ICN et la problématique des stratégies de cache. On commençe dans un premier temps par l'état de l'art des architectures ICN. Nous motivons le choix de CCN comme l'architecture la plus appropriée pour étudier le problème des stratégies de cache. Ensuite, nous présentons les stratégies de cache disponibles dans la littérature et les métriques à utiliser dans le reste de la thèse.

La deuxième partie contient les deux contributions principales de la thèse. On commençe par une évaluation des stratégies de cache presentées précédemment.

Dans la suite, nous présentons nos deux stratégies de cache basées sur la popularité: MPC et SACS. Dans la présentation de SACS, nous incluons un outil qui génére du trafic social: SONETOR.

Ensuite, nous résumons les contributions de cette thèse avec nos perspectives de recherche.

Deux annexes complètent la thèse. La première annexe détaille un cas d'utilisation du simulateur SONETOR, il montre les changements causés dans le modèle de popularité par les réseaux sociaux. La deuxième annexe decrit une étude realisée sur les publications du réseau social Pinterest. Ces publications ont été utiles pour alimenter le générateur du trafic social SONETOR.

1

General Introduction

Contents

1.1 Problem Statement 14

1.2 Manuscript Organization 15

Internet was initially built as a military experiment where the target was to share resources. Due to the expensive cost and scarcity of resources, only a small number of users had access to it. Nowadays, the situation has changed: prices of Internet-capable devices have dropped and the number of persons accessing it have exceeded the 2.9 billions [137]. Today, there exist over one device per capita and this number is expected to grow to nearly three devices in 2016 [11]. However the number of users is not the only aspect that has changed in the Internet. Available and transferred content in the Internet has skyrocketed as well. Google has already indexed 1 trillion web pages [42]. Every 5 minutes, 3.5 Petabytes cross the Internet. Smart phones transfer 1.4 Exabytes monthly [11]. These numbers are expected to keep growing. In 2016, networks will deliver 12.5 Petabytes every 5 minutes and Smartphones will transfer 9 Exabytes monthly [11]. VoD traffic will be equivalent to 4 billion DVDs per month [7]. Unfortunatelly, the evolution of the Internet architecture has not followed the growing patterns of Internet users and transferred data.

The original Internet architecture has been altered with incremental changes to provide all the services that the Internet offers today. The first Internet prototype, ARPANET, connected just one precise type of computers, microcomputers. Later on, other types of networks, such as satellite and mobile ratio networks, were interconnected through ARPANET. To handle this interconnection, the TCP/IP stack was designed. In 1981, the Domain Name System (DNS) was created to reduce the complexity to find hosts into this network of networks. Lately, the emergence of World Wide Web (WWW) permitted Internet to reach 600 million users in the late 2000s [13] and this number has kept growing since then. In less than a century, Internet evolved from serving a few selected group of scientists to reach more than 2.9 billion users. It moved from serving non-profit scientific and military activities to be an important part of the world economy. Indeed, Internet accounts for 3.4% of GDP in both G8 and BRICS countries [9]. It adapted from exchange of static files and messages to serve hours of digital video and multi-party video conferences. Nevertheless, the communication paradigm has remained unchanged and the network operation is still led by an end-to-end model.

The current communication paradigm of the Internet presents many limitations. First, the Internet architecture can offer multiple copies of content, but these copies are not linked together. In order to share resources of the network additional overlay mechanisms have been proposed over

the TCP/IP stack. Fast and reliable content transfer requires application-specific mechanisms such as Content Delivery Networks (CDN) or Peer to Peer (P2P) services. Second, security is achieved by third-party applications and services. Trust in retrieved content is not easy to achieve and most of the connections rely on untrustworthy locations. Third, the fact of retrieving content is strongly related to its location. Every Internet packet is addressed following source and target addresses. Every time content is requested, addresses are needed. Our requests are strongly linked to an address, signaling a particular location, when users do not care about location but getting the content as fast as possible.

The direct manner to resolve this problem is to replace *where* with *what* [16]. Host-to-host communication was an abstraction to solve the problems of the 60s. We argue that a paradigm based on named content fits better for today's communication. Based on this premise, Information Centric Networks (ICN) were proposed. ICN architectures are clean-state designs of the Internet, where the content is located at the center of the scene. In contrast to the current Internet where packets are addressed according to source and destination nodes (i.e. IP addresses), ICN architectures address content according to content name at the network layer. Every packet is named and this name serves to route content. The aim of ICN is to develop an architecture for content delivery. Several architectures have been proposed for ICN such as CCN [16], SAIL NetInf [124] and FP-7 PURSUIT [23]. These architectures have been defined giving high priority to satisfy end-users expectations and prioritizing efficient content distribution. Among these new architectures, Content Centric Networking (CCN) has attracted considerable attention from the research community.

1.1 Problem Statement

Today, users largely consume videos over the Internet. In the United States, the national average of daily video consumption in the Internet is about 15 minutes. These 15 minutes amount to 16 Gigabytes per month per user [8]. This accounts for 66% of all internet traffic [7]. Smartphones, tablets, Game consoles and Smart TV are all used to streaming video and by 2017 the number of smart devices number will double. In particular, web-enabled TV has a four-fold increase. Mobile video will increase 14-fold between 2013 and 2018 [7]. 100 hours of videos will be uploaded every minute to YouTube. Youtube, Netflix and many other services offer an unlimited selection of video files. Moreover, filesizes have grown in the last years with the emergence of high definition technologies such as High Definition Television (720p or 1080p) or Ultra High Definition Television (4k). Therefore, the 15 minutes of video daily consumption will become hours of consumption. The 16GB of American video traffic per month will become 600GB [122].

This is a serious problem for the Internet because it is a network built over an outdated communication paradigm. Unlike traditional broadcast networks which send one title to millions across the network at one time, the Internet transmits the same videos many times over. In fact, 10% of content represents 90% of the Internet traffic [122]. Congestion is getting out of control and new mechanisms are needed to fulfill Internet quality requirements.

In this context, temporary storage servers have been used all across the Internet to serve content requested many times. These temporary storage servers are called caching systems or simply caches. More recently, Content Centric Networking (CCN) has appeared as an architecture to replace the Internet architecture. With CCN, caches will be available at every node and the network will become a network of caches.

Caches have never been deployed in such a large scale. Having caches at every node makes the cache management problem very complex: it increases the available space for storing content

but the complexity to manage them increases at the same time. To manage the caches, caching strategies are used. Caching strategies decide what, when and where content is stored. In the literature, many caching strategies have been proposed such as Leave Copy Everywhere [62], Leave Copy Down [93], MAGIC [70], ProbCache [53] and Cache "Less" For More [54]. However, it is unclear which caching strategy fits best for every possible scenario and if better strategies can be defined. This is the subject of this thesis.

The contribution of this thesis are the following:

- After a study of the state of the art, we found out that diverse caching strategies have been proposed. They include solutions such leaving copies at every node or detecting the optimal node to create the copy. They use diverse information such as topological structure or the popularity of the content. However, these caching strategies have never been fairly compared. A first contribution of this thesis we **define a common framework**, where we are going to compare all the state-of-the-art caching strategies. This common framework is built based on the parameters selected from the literature.

- The second contribution consists in the **comparison of the caching strategies**. We analyze them according to three metrics: Cache Hit, Stretch & Diversity. The results show that there does not exist a best caching strategy. Every caching strategy overcomes other ones in at least one particular scenario, which is detailled in the thesis.With the comparison of the caching strategies, we discovered that the concept of popularity has not been fully exploiting.

- We then propose a **caching strategy based on the popularity of content**: Most Popular Caching Strategy (MPC). MPC caches only popular content. By purging unpopular content from caches, MPC saves resources and keeps on the caches elements likely to be requested.

- Finally, we propose a **caching strategy based on information from social networks**: Socially-Aware Caching Strategy (SACS). SACS detects influential users in a network based on social networks information. SACS proactively caches content produced by influential users into the network. Thus, SACS overcomes other caching strategies when users interact in a social network fashion.

1.2 Manuscript Organization

The manuscript is organized in two parts, the conclusions and two annexes.

The first part of the manuscript provides a state of the art in Information Centric Networks (ICN) and Content Centric Networks (CCN). Chapter 2 aims at explaining the principles of both architectures – ICN and CCN, presenting the architectures in-depth together with the different alternatives proposed for ICN and with a particular emphasis on CCN – the architecture we build open in this thesis. In a second step, we dig into the problem of caching in CCN in Chapter 3. We discuss properties of a CCN network and the research challenges that will imply its implementation. In this chapter, we describe the alternative solutions proposed in the literature with illustrative examples. Furthermore, we create a common ground for all the metrics proposed in the literature and we selected the most representatives of them.

The second part of the manuscript presents the contributions of the thesis. The analysis of existing caching strategies led us to find diverse simulation environments. The presence of diverse simulation environments complicates the cross-comparison of the proposed solutions. In

Chapter 4, we set a common evaluation framework. We describe and detail all the proposed simulation environments and we select the best parameters for the common scenario. Once the common scenario is presented, we compare all the caching strategies proposed in Chapter 3. This study shows the need for alternative mechanisms and new caching strategies. In Chapter 5, the Most Popular Caching (MPC) strategy is presented. MPC caches only popular content and purge the unpopular content from the caches. MPC is based on the premise that if we only store content more likely to be consumed, it is unlikely to spend resources on content that will be barely requested. Subsequently, we explore the possibility of adding information of social networks into CCN. We use the information of social networks to build a caching strategy. In Chapter 6, we present the Socially-Aware Caching Strategy (SACS). We conjecture a small number of users counts a huge amount of social relationships, dominates the activity and receives most attention from other users [74]. We argue that they produce content that is more likely to be consumed by others, and in consequence their content must be favored and replicated in priority. SACS prioritize content from Influential users of the social network. This caching strategy was evaluated in several environments ranging from the common evaluation framework to PlanetLab experiments.

The conclusions and a summary of our findings is presented in Chapter 7. We pay special attention to the scope of all our proposals and present open challenges that remain to be addressed in this architecture.

Finally, we include two annexes. The Annex A describes an use-case of the SONETOR (presented in Chapter 6). With this use-case, we describe the impact that the use of social network has in the popularity model selected during evaluation. The AnnexB is related to SONETOR too. It presents a characterization of the publications in Pinterest. Pinterest is a social network website such as Facebook or Weibo that focuses mostly in images. The characterization of the publications was used to create configuration files that will feed the SONETOR simulator. These feeds will be useful to recreate the behavior of Pinterests' users.

State of the Art

2

Information and Content Centric Networks

Contents

2.1 Information Centric Networks 20
2.1.1 ICN Features 20
2.2 ICN Architectures 23
2.2.1 Content Centric Networks 24
2.2.2 FP-7 Pursuit 25
2.2.3 SAIL NetInf 27
2.2.4 Other Architectures 28
2.2.5 Comparison of Information Centric Networks architectures 28
2.3 Choosing the ICN architecture: Content Centric Networking . 30
2.4 Summary 32

The Internet is currently mostly used for accessing content. Indeed in the 2000s, P2P traffic counted for about 80% of the overall Internet traffic. Nowadays, video streaming services such as YouTube represent the most important part of the Internet traffic. It is expected that the sum of all forms of video (TV, VoD and P2P) will be approximately 86% of global consumer traffic by 2016 [7]. However, the Internet was not designed for delivering content at large scale.

The Internet was designed for host-to-host communications. The aim was to share a limited number of expensive resources between a small number of users. These resources were located in precise locations and the exchange of information used messages sent from a source address to a destination address. Today, the scenario has changed. From a small number of users connected to the Internet, we have passed to 2.9 billions connected users. The number of devices has skyrocketed and its prices have drastically decreased. Moreover, the usage of Internet has changed. Nowadays, everything is available on the Internet from politics to tourism passing from diverse subjects such as sport, fashion or science. Users request every kind of content to the Internet and it is up to the Internet to retrieve the content. In order to retrieve it, the communication paradigm requires the requester to contact a server holding the content, in a host-to-host fashion. However, this communication paradigm does not seem to be optimal. Content shall be retrieved from nearest location. Users do not care about location of content. Based on the premise that users care about content and not its location, Information Centric Architectures were proposed. ICN sets the content at the center of the scene. Content becomes

the primary unit of ICN. Every single ICN packet is identified with a name that serves to route and forward content. Hence, new Information-Centric Networking architectures (ICN) such as CCN [16], SAIL NetInf [21, 22] and FP-7 PURSUIT [23] have been defined giving high priority to efficient content distribution at large scale. Among all these new architectures, Content Centric Networking (CCN) has attracted considerable attention from the research community [113].

The contributions of this chapter are the following:

1. We define ICN. We present the paradigm, its main features and its differences against the current Internet. The discussion is centered around four features: named content, caching, mobility and security.

2. We present the main ICN architectures proposed in the literature: CCN [16], SAIL NetInf [21, 22] and FP-7 Pursuit/PSIRP [23]. We analyze their implementations of the ICN paradigm and we select the architecture that fits the best with our problem: CCN.

This chapter is organized as follows. We describe the main expected features for Information Centric Networks in Section 2.1. We then introduce Information Centric Networks architectures in Section 2.2 and we motivate the choice of Content Centric Networking for our studies in Section 2.3. We summarize the chapter in Section 2.4.

2.1 Information Centric Networks

Information Centric Networks (ICN) is a clean-state design for the Internet architecture. The ICN architecture replaces the host-to-host communication paradigm by a network architecture based on named information. Thus, the content becomes the primary unit of this architecture.

ICN architectures support many features. Every ICN message in the network is named. Content becomes independent from location, application, storage and means of transportation. Thus, ICN enables transparent in-network caching and replication of content. Multicast, anycast and broadcast are natively supported. Security features are incorporated into the ICN architecture. All these features are discussed in this section, highlighting their main differences with the current Internet architecture.

2.1.1 ICN Features

Named content

The current Internet names content using Uniform Resource Identifiers (URIs) [27] and it names hosts with IP addresses. A URI is a string of characters used to identify the name of a resource while an IP address is a numerical label assigned to identify each device. The IP protocol defines an IPv4 address as a 32 bit number or an IPv6 address as a 128 bits number. However, the most used type of URI is the Uniform Resource Location (URL). The URL not only identifies an Internet resource but also specifies the means of obtaining the representation. The URLs are largely used as addresses for finding resources.

In the Internet, every URL is bound to an IP address. URL names are used to address web servers in a simple, easy-to-remember and human-readable fashion. This naming scheme is already composed of more than 10^9 names [42]. However, the issue is that names bind necessarily to addresses. Every time we request a piece of content, we request a URL which is translated into an IP address. As a matter of fact, multiple servers storing the same piece of content identify the same resource with distinct URLs. If we have distinct URLs for a common content, we are

not able to link together these pieces of content. We need to remove the mapping between URLs and hosts.

ICN architectures propose to name solely content. Every piece of content available in an ICN network has a name associated. These names must be decoupled from location of content.

The names of an ICN architecture may be classified in two: flat or hierarchical names. We analyze flat names and hierarchical names in terms of four properties: human-readable, location-independent, self-certifying and aggregation-capable. Human-readable property refers to names that can be understood by any human being. Location-independent means that names are not linked to the location where the names were created. The self-certifying property refers to a property where a name is generated from the content itself and can be used to check integrity of content. Aggregation-capable is a property that refers to the possibility of joining multiple names into a larger category.

Flat names are arbitrary bit strings. Flat names are usually calculated by algorithms that maps data to a fixed length string. P2P networks serve of flat names to address content [94]. Regarding properties, flat names are difficult to aggregate and not human-readable. However, flat names are usually self-certifying and location independent. As flat names are usually not-human-readable, an additional service is required to translate from human-friendly names to flat names. For example, eMule uses human-readable keywords which are translated them into flat names.

The hierarchical names are specified in a freeform hierarchy. Any name is composed of a sequence of human-understable strings separated by a character (e.g. a slash – /). For example, `/President/JuanDomingoPeron/` or `/music/Madonna/LaIslaBonita` are valid hierarchical names. Hierarchical names are more usable and remain unchanged even if the certification algorithms evolve [35]. Hierarchical names are human-readable and easily aggregated. Every hierarchical name is not self-certifying. Location independency of names is possible to achieve. However, human-readable names have implicit semantics associated. It means that names may be ambiguous or may contain information which causes the lost of the location-independence property. For example, sound is an homonym in English. It refers to an ocean inlet larger than a bay or to a noise or music. The hierarchical name `/sound/Aventure` may refer to the Sound located in the islands Malvinas Argentinas or some piece of music related to adventures.

To sum up, Internet names its resources using URLs. These URLs bind resources to locations, which means that the Internet naming scheme is location-dependent. However in a content delivery network, the naming scheme should be location-independent. The names should be independent from their location. Thus, the network may serve requests from the nearest location which may be a major improvement for video streaming services and large files download. We argue that the naming scheme used in the current Internet is not appropiate for content delivery. Information Centric Networks propose a naming scheme most addapted for today usage.

Caching

In the current Internet architecture, packets can not be reused. Every time similar content is requested, new distinct packets are generated. Even if a large number of users request a common piece of content, packets will be re-created every time [16]. The caching is performed at the application layer and caching applications are deployed for particular protocols [93]. Caching applications understand particular application's protocols and extract content from the payload to store it in its temporary storage memory. It means that the current Internet architecture can not link together any two similar pieces of content retrieved through different protocols. Furthermore in certain cases, two instances of a same packet generated through the same protocol

can not be linked together. For an optimal content delivery, every piece of content may generate the same packet in order to be re-used multiple times.

ICN includes caching capabilities at every node and it is therefore a network of caches. This network of caches has three basic properties [93]: (1) end-users are being served from intermediary nodes without being aware (cache transparency); (2) caches are placed everywhere in the network (cache ubiquity); (3) ICN handles packets at chunk level, every packet is self-identified by a name (fine-granularity of cached content). These properties open many research challenges which will be described in Chapter 3.

ICN supports the re-use of packets. The same packets are generated every time the same content has to be transmitted. These packets may be temporary stored at intermediary nodes to serve forthcoming requests for content.

Caching is the ICN feature that may have the major impact on future Internet. Based on their structured caches, Content Delivery Networks (CDN) will serve 50% of global traffic by 2018 [7]. If the Internet is replaced by ICN, caches will be located everywhere and it is expected that a larger than 50% traffic will be served from ICN. In addition, CCN performs caching operations at chunk levels [93]. The current Internet names content at the application layer and fragmentation depends on the protocol used. In CCN, small self-identifiable names are used for every distinguishing every packet. There exist a fine-granularity of named content. This feature permits a CCN network to identify the content being sent into the network. Being content aware in the network offers the posibility to take smart decision about content to be cached or not. With the replacement of Internet architecture, caching will have a major impact on network performance.

To sum up, the current Internet architecture is not optimal for content delivery. It is difficult to link together two pieces of similar content and its caching capabilities are limited. ICN architectures simplifies the linking of similar pieces of content. ICN architectures includes caching capabilities at every node. Every ICN node may serve content. We argue that in-network caching capabilities are a necessary feature for implementing a planet-scale delivery network.

Mobility

According to the Cisco Forecast [7], Mobile video traffic accounted for 53 percent of the whole Internet traffic in 2013. Smartphones will reach 66 percent of total mobile data traffic by 2018 and mobile traffic will increase 11 times by 2018. However, The internet design was conceived with a static vision of the users in mind. With mobility, multiple problems emerge and are summarized in [95].

Within the current Internet, many applications and protocols assume that a host is located at one place (uni-homed). Various infrastructure decisions follow this assumption. By definition an HTTP request is received over a single HTTP connection. As such, it is difficult to exploit multiple network interfaces. For example, Internet is not optimized to maintain multiple connections to a service such as Facebook. If we connect to Facebook with our personal computer and personal cell phone, two independent connections are generated to request content. In contrast, the ICN approach is multihomed. ICN does not bind a flow with a specific interface. As ICN packets describe content, multiple devices requesting the same pieces of content are going to request the same packets. Thus, ICN may re-use packets and to optimize the delivery path.

Many mobility mechanisms attempt to maintain the node network address for long-term. Protocols such as BitTorrent are based on the fact that a node's IP address is not going to change and in case it does it must be re-registered on the system. In contrast, ICN decouples names from their location. As ICN names describe items, they are independent from the location

of the node.

Lastly, many Internet protocols are connection oriented. More precisely, TCP is a connection-oriented protocol and most of the Internet traffic is based on it [91]. Every time mobile users change their location, connection must be re-established. As every TCP connection is different and in order to receive large bulks of data, the same connection may be re-established if the connection location is changed. ICN communications are made at network level. ICN nodes request named content and they expect packets in return not necessarily from the same hosts. These packets may be self-verified by users and the re-connection process is not required. ICN nodes may receive 50% of the content from one server and 50% from other server. Then, ICN nodes may verify packets and re-build content without caring about the TCP connection.

To sum up, the current Internet architecture was conceived in an era of static computers. Nowadays, mobile devices are everywhere but the Internet has many problems to deal with them. ICN is an architecture conceived in this context and we argue that it is a better alternative for dealing with mobility problems.

Security

Internet has many security issues. Since its early days, the Internet allows to forward traffic in any direction. However, security was considered a low-priority issue due to the limited and trustful environment of its users. Scientists and military units were the only users of the Internet and they were conscious of their experiments in the network. In addition, the number of sensitive information was minimal and the cost of storing information was extremely high. Nowadays, Internet serves billions of users. Many sensitive data is transfered every second and this sensitive information is stored in many different servers.

Denial of Services attacks and malicious traffic are common and it is complicated to defend against them. Many solutions have been deployed to cope with them such as Firewalls, Honeypots, SPAM filters and Deep Packet Inspection (DPI) algorithms. By decoupling destination from messages and changing the model from sender-driven to receiver-driven, ICN expects to simplify these complicated tasks and be able to prevent attacks by including security features in its design. The ICN paradigm brings encryption support to the network layer. Intermediary nodes are not able to inspect the content. They are aware of the content they pass – by their name – but they can not access the data itself. ICN architectures are reluctant to certain types of Internet Denial of Services attacks and flash crowds access.

To conclude, we argue ICN architectures are better alternatives for the Internet architecture in terms of security features. ICN architectures are designed with the conception of security in constrast to the current Interent architecture.

2.2 ICN Architectures

Many ICN approaches have been proposed so far. They focus on clean-state design of the Internet. These architecture address the problems and limitations of the current Internet. The ICN architectures include *Content Centric Networks* (CCN) [16], *Publish Subscribe Internet Technology* (FP-7 PURSUIT) [23], *Scalable & Adaptive Internet SoLutions* (SAIL) [21, 22] and other architectures that received less attention from the scientific community: Dona [18], CONVERGENCE [116] and MobilityFirst [120].

2.2.1 Content Centric Networks

Content Centric Networking (CCN) is a clean-state ICN architecture for the future Internet. CCN envisions reshaping the Internet protocol stack, by completely removing the TCP/IP stack.

Names in CCN are hierarchical and similar to URIs [27]. Names are made up of components and divided by a separator. Every component can be anything such as a human-readable name, a hash or some arbitrary length string. For example, `/INRIA/webpages/Bernardini/home.html`, `/newspapers/UK/BBC/2014-10-12/10:25/abcd1234` and `/\$@aab12*!/xyz123` are valid CCN names. In CCN a request matches a name if this name is a prefix of a request's name. For example, `/INRIA/webpages/Bernardini/home.html` will match `/INRIA/webpages/Bernardini/home.html/v1` and `/INRIA/webpages/Bernardini/home.html/v2`. The suffix of a CCN name describes the versioning and sequence number.

CCN communication is based on two messages: *Interest* and *Data*. *Interest* messages are generated by end-users and express the intention for a particular content. The response arrives as a *Data* message. All the messages are forwarded hop-by-hop by Content Routers (CR) in an anycast fashion. CR maintains three data structures: *Forwarding Information Base* (FIB), *Pending Interest Table* (PIT) and *Content Store* (CS). The FIB maps CCN names to output interfaces and it is used to forward *Interest* messages to their destination. The PIT keeps track of the incoming interface which *Interest* messages did arrive in order to forward back the *Data* answer to its original requester. Finally, the CS stores information (i.e. *Data* messages) that has passed through the CR. The CS serves as local cache for serving future incoming requests.

When an *Interest* arrives, the CR extracts the *Interest* name and looks up into its CS for content stored that matches the name. If content is found, it is automatically sent back through the interface over which the Interest arrived and the *Interest* is discarded. Otherwise a longest prefix match is performed into CR's PIT table to decide whether the CR is waiting for a request of this content. If an entry is found, the *Interest* is discarded. If no entry is found, the CR's FIB table is checked to decide the interface to redirect the *Interest*.

In Figure 2.1, we show a communication example of CCN. The example illustrate the exchange of messages between *Subscriber* and *Publisher 1* to retrieve content `/aueb.gr/ai/new.htm`. Table descriptions are represented with different colors after the traversal of *Interest* and *Data* messages, with titles in blue and red respectively. The *Subscriber* sends an *Interest* message to *Content Router (CR) A*. *CR A* checks its PIT table and detects that there are not pending requests for content `/aueb.gr/ai/new.htm` in this node. Then, it selects the best path to find information according to its FIB table doing a lookup operation. It forwards the *Interest* message through node C because the longest prefix matching indicates that `/aueb.gr/cs` is a prefix of `/aueb.gr/`. Immediately, *CR A* adds an entry for content `/aueb.gr/ai/new.htm` in its PIT table.In *CR C*, there are not entries in the PIT table. The FIB table details that prefixes of `/aueb.gr/` should be forwarded through *Publisher 1*. In consequence, *CR C* inserts a new entry on its PIT table for the content and the *Interest* arrives to Publisher 1. The *Interest* message has arrived to *Publisher 1* where the requested content is located. *Publisher 1* creates the *Data* message in response to the *Interest*. The *Data* message follows the information left in every node in its PIT table, which explains the path towards *Subscriber*. Finally, the content arrives to the *Subscriber* and all the caches in between may store the content in there *Content Store* tables (i.e. Cache) according to the caching strategy chosen. We detail this process in the Chapter 3 and, in this example, we assume that a copy is left in every node that content has passed through.

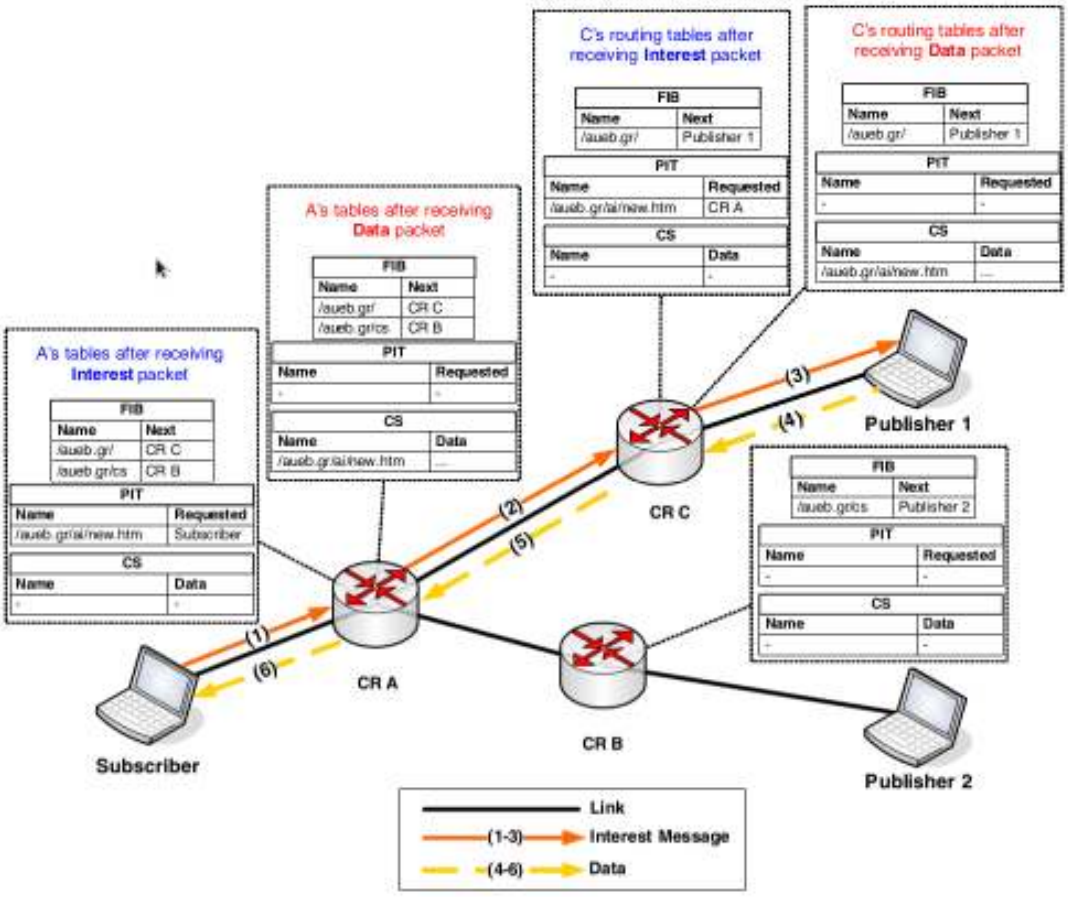


Figure 2.1: CCN Architecture, extracted from [20]

2.2.2 FP-7 Pursuit

The Publish Subscribe Internet Technology (PURSUIT) project proposes an architecture that completely replaces the IP layer with a publish-subscribe layer. It is funded by the EU Framework 7 Program and it is the continuation of the *Publish Subscribe Internet Routing Paradigm* (PSIRP) [23].

The architecture is composed of three elements: *rendez-vous*, *topology manager* and *forwarding*. The *rendez-vous* links subscriptions to publications. The *topology manager* creates and maintains routes between publishers and subscribers while *forwarding* is in charge of delivering data from one location to another.

The information is identified by two pairs of IDs: *scope ID* and *rendez-vous ID*. The *scope ID* is the category of the information such as music genre, video streaming content or communities of users. The *rendez-vous ID* is the flat identification for the object itself. A piece of information may be published several times with different scopes. For example, a picture of the Eiffel Tower may be published in a scope of tourism and in the personal scope of the author.

The publisher announces new information to its nearest *rendez-vous node*. The rendez-vous node is located with a Distributed Hash Table (DHT). When a subscriber is interested in that information it contacts its local *rendez-vous node* which configures the *topology manager* to create the route between publisher and subscriber(s). Once the publisher is aware of the existence of

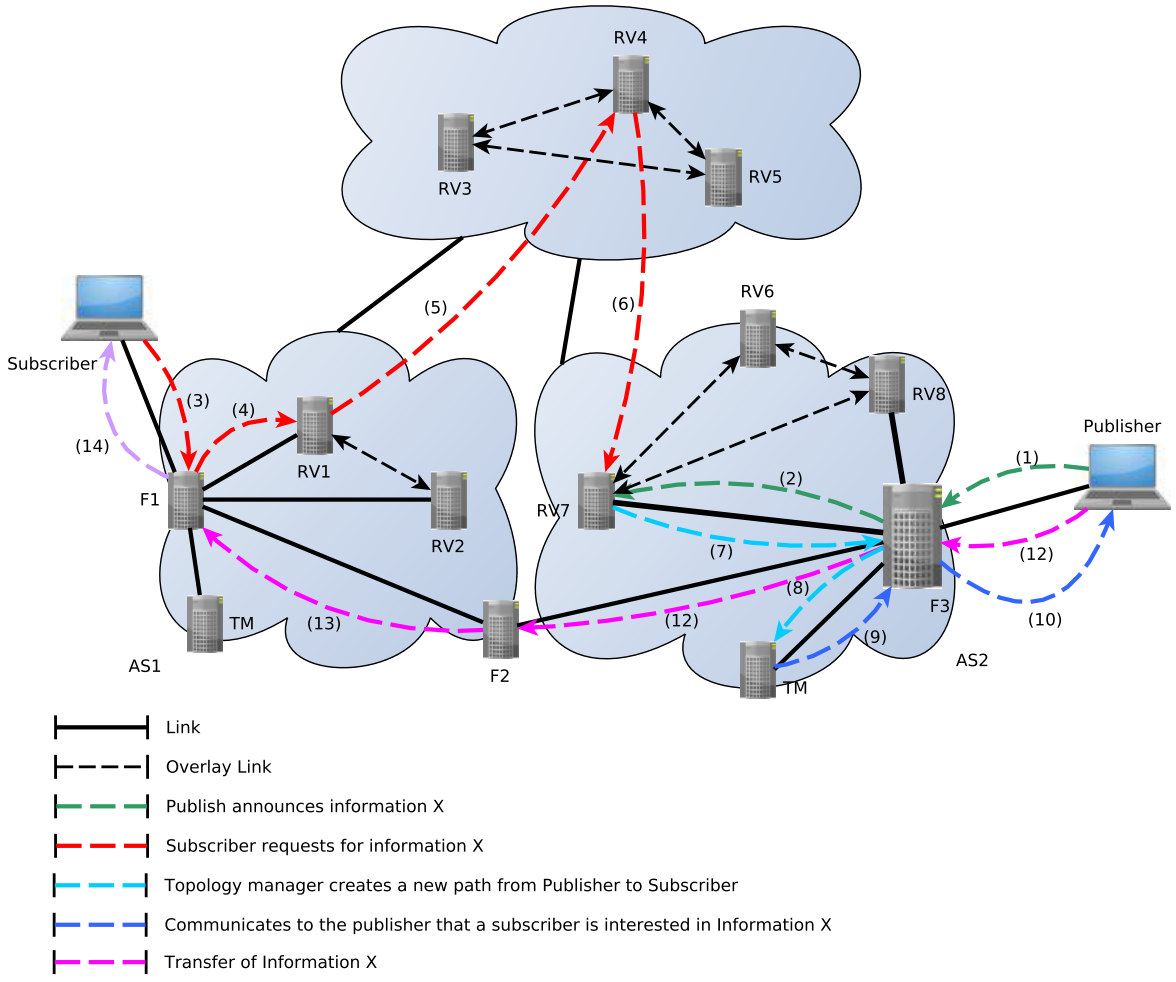


Figure 2.2: The PURSUIT Architecture

the new route, he instructs the *forwarding* function to transfer the information.

In Figure 2.2, we depict an example of the FP-7 PURSUIT architecture. The solid black lines represent links and the dashed black lines overlay links between nodes. The example shows the procedure to publish and retrieve Information X. First, the Publisher communicates his intention to publish Information X and sends a message to the nearest *rendez-vous node* (RV) as we can see in the green dotted lines tagged as 1 and 2. Later on, the Subscriber reports interest for Information X. This interest is translated into a communication to RV1 which contacts RV7 as shown with red dashed lines (3, 4, 5, 6) – through a communication among *rendez-vous node* using the DHT. Subsequently, the Topology Manager (TM) retrieves the best route from Publisher to Subscriber and communicates the new path to publisher in light and dark blue dashed lines (7, 8, 9, 10). Finally, the Publisher transfers Information to Subscriber following the path (F3, F2, F1) as shown with purple dashed lines (11, 12, 13, 14).

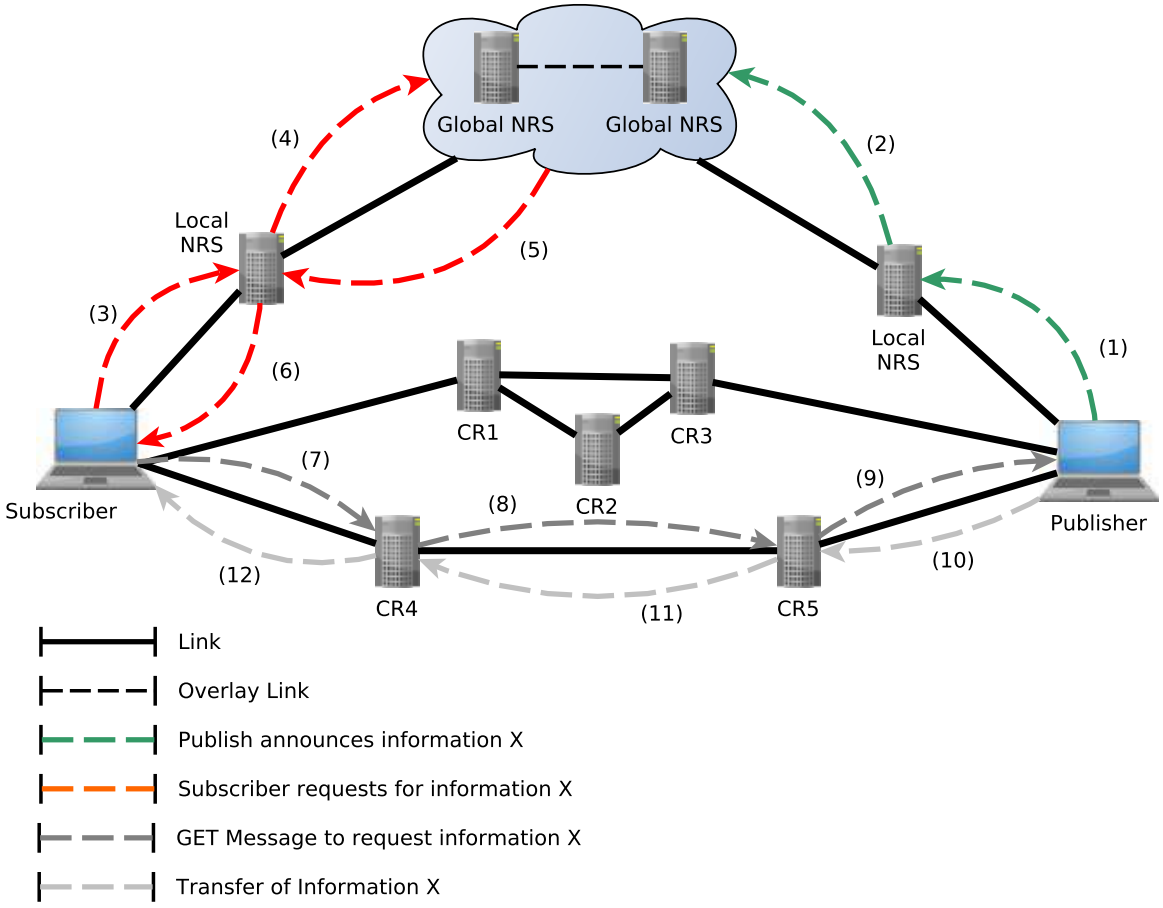


Figure 2.3: SAIL NetInf Architecture

2.2.3 SAIL NetInf

The *Scalable and Adaptive Internet soLutions* (SAIL) project investigates smooth transitions from the current to the future Internet. It combines elements of the previously explained CCN and PURSUIT architectures [20].

SAIL names are an hybrid combination of hierarchical and flat names. The first part is flat and describes the authority entity that created the object while the second part is hierarchical and describes the local denomination of the information. When we mention authorities, we refer to central entities. These central entities build the namespaces.

The SAIL NetInf architecture has a multilevel DHT organization [32] based on global name resolvers. The global name resolvers handle the resolution of the authority part of the name. Then, the rest of the name is resolved by a local name resolver. The subscriber sends a message to its local name resolver which consults the global name resolver to find the publisher. Once found, the publisher answers the message.

The SAIL NetInf architecture includes in-network caching capabilities at every node. It includes a hierarchy of caches, which a small number of caches are placed at root levels of the hierarchy. The higher the cache is located in the hierarchy, the more space it has for temporal storing of content [20].

In Figure 2.3, we describe an example of information retrieval –of Information X– in the SAIL NetInf framework. The publisher communicates that he owns the Information X to its Local Name Resolver (*Local NRS*). The *Local NRS* announces to the Global Name Resolver (*Global NRS*) the existence of the publisher. Both actions are shown with green arrows. The subscriber requests the Information X to its *Local NRS* which communicates to the *Global NRS* and returns information about the path to traverse (red arrows). Subsequently, the subscriber sends a request through the path indicated (light grey arrows) and the publisher answers with Information X (dark grey arrows). The *Global NRS* returns routing directions which are instructed in the sent messages in contrast to the CCN approach where decisions are taken according to the name of interest received.

2.2.4 Other Architectures

CCN, SAIL NetInf and FP-7 PURSUIT are important projects and represent strong contributions to ICN. Alternatives ICN architectures exist and they are described in this section.

DONA [18] creates an ICN network based on the current Internet architecture. DONA replaces the DNS system with a name resolution system based on flat self-certifying names. The importance of DONA relies on being the first to include an anycast name resolution which was included lately in the CCN architecture [18]. TRIAD is DONA's predecessor [19] and its main difference relies on the lack of centralized querying place. Every node has a routing table which indicates the next node to jump to, following an anycast routing approach.

COMET [24] shares concepts of name resolution and routing with DONA. Its main difference relies on the entities that handle name resolution and routing. Although they are coupled, different entities are in charge of solving the resolution and the routing. Independent modules gives more flexibility to the architecture and simplify its extension. Like the NetInf project, it offers entities organized in autonomous systems in charge of taking decisions. Its routing scheme may be seen as a fusion between PURSUIT and NetInf approach. Another interesting approach of COMET is that as NetInf, it may easily be incorporated into the current Internet architecture as an overlay over the IP protocol [24].

Mobility First is an US founded project [92], where mobile devices are at the center of the scene. The architecture of Mobility First separates the names for all the entities attached to the network in order to deal with mobility problems once mobile devices change their connection point.

2.2.5 Comparison of Information Centric Networks architectures

ICN architectures previously introduced share similar concepts and differ regarding to other features. We present the implementation of these features for every ICN architecture. In Table 2.2, we summarize the most important characteristics of ICN architectures. The Table 2.1 details naming choices in every architecture.

Naming

The structure of names for an ICN architecture may be classified as hierarchical, flat or a combination of both. CCN uses a hierarchical naming scheme while DONA and FP-7 PURSUIT are built over flat naming schemes. SAIL NetInf is the only solution to use a combination of both approaches: flat for inter-domain and hierarchical for the intra-domain part.

The strongest argument in favor of Flat names is that the location-independent property is easy to achieve. Flat names are usually calculated with deterministic functions. No matter

in-which node the function is calculated, its result will be exactly the same [35]. As a negative argument, flat names are difficult to aggregate.

In contrast, hierarchical names present completely different properties. CCN names are human-readable. CCN uses hierarchical names and offers the self-certifying features by appending a signature to the Data packets, calculated over the payload. As we pointed out in Section 2.1.1, names may be ambiguous and the location-independent property is difficult to achieve. However, having human-readable names permit content-awareness reactive solutions in the network and this is a radical change with respects to IP. For example, in Chapter 6, our socially-aware caching strategy is able to determine flows that belong to social networks based on content-awareness ICN.

Name Resolution and Data Routing

For an ICN architecture based on names, name resolution and data routing is essential. Solutions may be classified according to the type of the naming scheme used or depending on entities served to solve requests. On the one hand, flat names cannot be aggregated but they are globally unique. Flat names are resolved by global entities as it is the case in the FP-7 PURSUIT and SAIL NetInf approaches. On the other hand, hierarchical names can be easily aggregated and CCN proposes a solution without central entities.

Relying on central entities for name resolution is the solution currently used in the Internet. The Domain Name System (DNS) [96] is a hierarchical distributed naming system, which is used today to find IP addresses. DNS has been working since 1983. Why should not we use it for ICN architectures? ICN routing is based on named content. Named content must be location independent. However, the DNS system is based on the fact that IP addresses are assigned geographically. ICN architectures based on central entities for name resolution are NetInf and PURSUIT. Both approaches are based on Distributed Hashing Tables (DHT) which are managed by the central entities. PURSUIT bases itself on a hierarchical DHT to spread the load of a huge name space that cannot be aggregated. Resolution is managed by an independent topology manager module that calculates data routing paths with bloom filters [23]. Even if the PURSUIT mechanism does not scale to Internet size, in certain cases it may offer optimal delivery paths based on multicast [41]. SAIL NetInf manages name resolution with two level DHTs [32]. Locally NetInf resolves requests for global information. The information about the path of the packet is stored in the packet itself. Even when these mechanisms are currently working and have worked in P2P environments for several years, we believe that the challenge of using flat names on the Internet is minor and it does not offer many new features. In addition, as we are reusing the features of the current Internet, the same problems may arise in the near future.

Name resolution without central entities is the solution chosen by CCN. CCN names are hierarchical and may be easily aggregated. Thus, CCN proposes an *anycast* based routing approach. This *anycast* approach have been proposed in [16] but nobody has figure it out how to make it work. Current solutions such as OSPFN [30] are based on current Internet solution [31] and violate the feature that names should be location-independent.

Caching

Caching is a fundamental feature of ICN. Caches located everywhere or in particular locations may have a great impact on the performance of the network and on the management of the network. Caching solutions may be classified in on-path and off-path approaches. In on-path

	CCN	PURSUIT	SAIL NetInf
Names	Hierarchical	Flat	Flat & Hierarchical
Name Resolution	Anycast	DHT-based solution	DHT-based solution
Name Routing	Anycast coupled with resolution	Decoupled from resolution.	End-to-End
Name Aggregation	Yes	Scopes with local names	Authority with local names

Table 2.1: Comparison of ICN Architectures

caching, when a request is received it is responded to with a locally cached copy of the content in the path taken on the resolution. In off-path caching, caches regularly announce their information to the name resolution system to answer forthcoming requests. Off-path caching requires extra configuration and complicated selection techniques. Content Delivery Networks (CDNs) actively replicate information in caches located in particular locations, usually ISP facilities. CDN calculates newly and top requested content. It then pro-actively replicates copies of these pieces of content.

All ICN architectures support on-path caching. However, the decoupling of name resolution and routing complicates the use of opportunistic caching. Name resolution and data forwarding usually follow distinct paths, which complicates the act of storing content over a path that will be subsequently chosen to forward requests for similar content. In contrast a coupled name resolution and data forwarding means request and response share the same path.

NetInf and PURSUIT decouple name resolution and forwarding, which means caching-off-the-path. CCN has been considered as coupled name resolution and forwarding solution, which means on-path caching. However, many questions are still unanswered about routing solutions for ICN architectures. The cache announcement process and content discovery are still undefined. We believe both topics will lead to off-path mechanisms, which implies that, for example, CCN may eventually decouple name resolution and data forwarding.

The subject of caching is deeply analyzed in this thesis. Caching in ICN is a fundamental feature that will drastically impact performance of the whole network. Nevertheless, all current architectures offers similar features in terms of caching. We do therefore not consider caching as a major distinct feature and we believe advances in CCN caching can be easily extended to other ICN architectures and viceversa.

2.3 Choosing the ICN architecture: Content Centric Networking

So far, we have compared ICN architectures in terms of *naming scheme*, *name resolution and data forwarding* and *caching*. In this section, we motivate our decision of ICN architecture according to these topics that have tipped the scales towards CCN. In addition, we analyze in an informative nature other aspects (i.e. *simulation platforms* and *community decision*).

As we mentionned, CCN names content with hierarchical names. Bringing hierarchical names

Architecture	Naming	Name Resolution and Data Routing	Caching
CCN	Hierarchical, may contain publisher-specific prefix.	Routing protocol used to flood name prefix information. Coupled: routing state for data is established at routers during request propagation.	On-path caching at content routers. Off-path caching requires additional routing information.
PURSUIT	Flat, consisting of scope and rendezvous part. Scopes may be organized hierarchical.	DHT-based rendezvous network matches subscriptions to publications. Scopes can be used to limit publication/resolution. Decoupled: topology management and forwarding are separated from rendezvous	On-path caching difficult due to decoupled operation. Off-path caching requires additional registrations.
SAIL NetInf	Flat names consisting of authority and local part. Possible name aggregation for the same authority	Coupled: requests accumulate routing state during name resolution. Decoupled: DHT-based name resolution returns content locator. Hybrid: name resolution returns routing hints to assis coupled operation	On-path caching at content routers. Off-path caching requires additional routing information or registrations.

Table 2.2: Comparison of key functionalities partially extracted from [20]

to the network layer is a major change which opens a new range of possibilities to study awareness into the network. We therefore believe that CCN is a better candidate due to hierarchical name based naming-scheme. In this thesis, we assume CCN names are hierarchical and human-readable. Names express content of packets and the type of protocol being used. Our hypothesis is that every packet can unequivocally describe its content.

CCN couples name resolution and forwarding solution (i.e. on-path caching). However, decoupling resolution and forwarding is considered as a more wide and complete solution. As the final routing solution is not decided for an ICN architecture, we have chosen to couple name resolution and data forwarding in order to deploy solutions that are supported in all the ICN architectures. With regards to name resolution and data forwarding, all ICN architectures are good candidates.

The subject of caching is deeply analyzed in the rest of the thesis. Caching in ICN is a fundamental feature that will drastically impact performance of the whole network. Nevertheless, all the ICN architectures offer similar features in terms of caching. The differences relies in the implementation of mechanisms. We therefore do not consider caching as a major distinct feature and we believe advances in CCN can be easily ported to other architectures and viceversa. In addition to this argument, CCN implements caching in a simple and clean manner. We argue that CCN is the best candidate for caching due to simplicity of its mechanisms.

The next line of arguing is informative in nature. We have considered the available simulation

tools and the choice of ICN architecture in the literature.

According to simulation platforms, ICN studies have provided quality software. Most of the ICN architectures are jointly released with a prototype. Blackadder [111] and Blackhawk [112] have been proposed for FP-7 PURSUIT and [118] for MobilityFirst project. NetInf offers its OpenNetInf library [124]. CCN proposes the CCNx prototype [113]. Some simulators have been proposed as well. icnsim [117], NS3 DCE CCNx [119], ndnSim [33], ccnSim [64], ccnpl-sim [115] and Icarus [34]. All but the first simulate a CCN network. By the number of available simulators, the ICN community has chosen CCN as ICN architecture.

We have analyzed the papers published at the conference ACM ICN from 2011 to 2013, the only conference that targets exclusively ICN architectures. We have extracted the ICN selected architecture of every published paper. In the Figure 2.4, we show the frequency of appearances of every ICN architecture. There are three charts for ACM ICN 2011, ACM ICN 2012 and ACM ICN 2013 respectively. The x-axis shows the ICN architecture and the y-axis describes the relative frequency. As we can see, most of the articles had chosen CCN in every edition of the conference. CCN articles form 50% of the publications of the conference. Clearly, the community has chosen CCN as the ICN architecture.

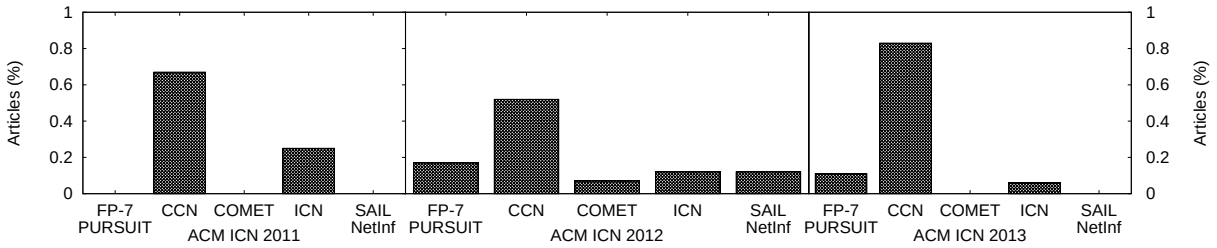


Figure 2.4: Selection of ICN architectures in the community

Projects such as DONA, TRIAD and COMET incorporate concepts that may be found in SAIL NetInf, FP-7 PURSUIT and CCN. Interesting research topics such as Mobility which Mobility First targets as main objective should be treated carefully, but once a basic understanding of the final decision is traced.

CCN is the best candidate for our thesis. Its name structure composed of human-readable keywords permits that the nodes become content-aware. Every CCN node can be enhanced to react according to context in the network. The CCN architecture couples name routing and name forwarding. For CCN, there is no clear solution about the problem of routing. However as we are not aware of the final caching strategy to be chosen, we believe this fact will not affect our studies. Once the final solutions are chosen we may decide whether to couple or decouple name resolution and name forwarding. The caching features of CCN are simple.

2.4 Summary

We have presented the limitations of the current Internet and a novel architecture that expects to overcome all these problems: Information Centric Network (ICN).

ICN is a clean-state design of the Internet. As today the Internet is used mostly to access information, the information is placed at the center of the scene. ICN includes many new features and we detailled them in terms of naming scheme, caching, security and mobility. We described the different alternatives of implementation which are chosen by the ICN architectures. So far,

many ICN architectures have been proposed, we describe the most important ICN architectures: CCN, Sail NetInf and FP-7 PURSUIT. We have described them with ilustrative examples and analyzed the implementations of their main features. Subsequently we have compared all of them in terms of the features. After the comparison, we came up to conclude that CCN is the most appropiate ICN architecture to perform our studies due to its content-awareness features, hierarchical names and in-network caching capabilities.

Bringing in-network caching capabilities to every node of the network becomes a major change to the Internet. It may have drastic impact on the Internet performance. We believe the efficiency of caches is going to have a major impact in the general performance of ICN. As we have pointed out, caching features in every ICN architecture are similar and the changes appear into their implementation. The findings on CCN may be easily ported to other ICN architectures.

3

Caching in Content Centric Networking

Contents

3.1 Caching in Content Centric Networking 36
3.1.1 Limitations of Caching in the Current Internet 36
3.1.2 Caching Properties of Content Centric Networks 37
3.1.3 Improving Performance of Content Centric Networking by means of Caching 38
3.1.4 Other Research Challenges on Caching in Content Centric Networking 39
3.2 Replacement Policies or Caching Strategies 42
3.2.1 Replacement Policies 42
3.2.2 Caching Strategies 43
3.3 Caching Strategies 43
3.4 Caching Metrics 48
3.4.1 Caching Performance Metrics 48
3.4.2 Traffic 51
3.5 Summary 53

A cache is a mechanism for the temporary storage of content such as web documents, images or videos. Cache aims at reducing bandwidth, user-perceived delay and server load. Web cache has its origins in the early 90's. It was originally used by companies giving Internet access to their users through a special HTTP server called a proxy. A proxy server processes requests by forwarding them to remote servers; it intercepts responses and handles the responses to the clients [14]. When the proxy servers contain temporary storage memory, they are commonly called caches. As a matter of fact, coordination of caches becomes a major issue. Alternative caching orchestration solutions were proposed such as hierarchical and distributed caches. Hierarchical caching solutions propose deployment of caches at every level of the network [29]: local, institutional, regional and ISP in such a way that requests travel up through the caching hierarchy [2]. Distributed caching solution sets a large number of caches at the lower level of the network, the edge nodes. Thus, requests stay at the edge of the network where there are less congestion [28]. Distributed caches gained a momentum with the arrival of CDNs. CDN caches content at the edge of the Internet, close to end-users. The Akamai CDN already delivers 15-20% of all Web traffic worldwide [44] and it is expected to delivers 50% of traffic by 2016. The Akamai platform comprises more than 61,000 servers [44], which is an enormous number. However, caches have never been used in every device connected to the Internet. The CCN

architecture propose in-network caching capabilities at every node of the Network. Thus, there will exist at least 3.7 billion of available caches for storing content [15]. As a matter of fact, finding solutions that can cope with this large number of caches is essential.

The contributions of this chapter are summarized as follows:

- We present the problem of caching in CCN, which includes in-network caching capabilities at every node. Therefore CCN acts as a network of caches. We describe the CCN main caching features and the caching research challenges. In particular, we describe the problems of improving the global performance of a network of caches.
- We then explain the concepts of replacement policy and caching strategy. We motivate the choice of studying caching strategies in this thesis.
- Subsequently, we describe the 5 major state of the art caching strategies proposed in the literature: Leave Copy Everywhere (LCE), Leave Copy Down (LCD), MAGIC, ProbCache and Cache "Less" For More.
- Finally, we summarize the metrics used in the literature to assess performance of caching in CCN. From the listed metrics, we selected those that are the most representative of the performance of caches in a network of caches.

In this chapter, we describe the principles of caching in CCN in Section 3.1: its main properties and its open challenges. These problems and challenges are divided in two parts: those problems related and unrelated to improving the performance of a network of caches. We then explain the difference between caching strategies and replacement policies in Section 3.2. We continue with the description of caching strategies proposed in the literature in Section 3.3. Subsequently, we analyze metrics of caching in Section 3.4. Finally, we draw some conclusions in Section 3.5.

3.1 Caching in Content Centric Networking

At packet level, the CCN communication is processed with two messages: *Interest* and *Data*. Every consumer report its intention to consume a content with an *Interest* packet while the content is sent with a *Data* packet. *Data* packets contain the content itself and may be stored in the caches. These *Data* packets may be re-used several times in the network. Every CCN node includes in-network capabilities. CCN is therefore a network of caches.

In addition, every CCN packet includes a CCN descriptive name. CCN names are organized in components which describe the content contained in the packet. CCN packets include encryption of raw content. However, the CCN name of the packet is visible to every intermediary node without encryption. Every intermediary CCN node is aware of packet being transmitted. CCN is a content-aware network.

These two characteristics induce new properties and open many challenges.

3.1.1 Limitations of Caching in the Current Internet

In this section, we describe the limitations of the current Internet in terms of caching:

- **Caches for every protocol.** Current cache systems target particular protocols such as HTTP or P2P. P2P applications use proprietary protocols, which make each P2P application an autonomous system. Caching between distinct P2P protocols is not possible [26].

HTTP caches follow a domain-based naming convention. Two different names are generated for two copies of the same content stored in different domains. Two HTTP caches, stored in different domains, can not interact among each other [93]. Thus, the same content is stored several times without possibility of being reused.

The same problem occurs with every protocol, the Internet has not internal mechanism to cache content spanning different protocols.

- **Caches for every application.** Current caching systems are configured for particular situations. For example, a caching server such as SQUID [125] may be configured to serve a newspaper website or a video streaming service. The configuration of these scenarios is completely different: the newspaper website may require to cache large images but the content is going to change every day or even many times per day. Video streaming services require large capabilities of caching storage, orders of magnitude larger than the newspaper, and videos are expected to be persistent.

  The current Internet approach requires to configure the caches for every situation. Furthermore, if the same server hosts many different applications, it requires many caching solutions with different configurations.

- **Caches at fixed places** Current Internet has predetermined cache locations, typically organized in a linear or hierarchical structure [14]. CDN networks have caches in fixed locations such as ISP. For example the free and open solution, CoralCDN, has between 300 and 400 servers running on the PlanetLab platform across 100-200 sites worldwide [48]. A comercial solution such as Akamai has 61,000 servers in 70 countries [44]. Many countries are still uncovered by these solutions. Commercial solutions are available for those users who can afford it. It means that the access to caches is limited by its fixed locations and for its high prices.

- **Lack of homogeneous naming scheme** In the Internet, there is a lack of homogeneous naming scheme. Similar content being transmitted through different protocols can not take advantage of caching capabilities. The names are decided by publishers and it is impossible to decide that both content are similar.

To sum up, the current Internet lacks a unified naming scheme and the caching software is built in the application layer. A unified naming scheme permits that distinct protocols or applications share the caching capabilities of the Internet while they are transferring the same pieces of content. As the caching software is built over the application layer, the software is built for particular applications and protocols. Every time a new protocol or application is released, the caching software must be adapted for this new solutions. ICN caching features permits to overcome these problems following certain properties. These properties are analyzed in the following section.

3.1.2 Caching Properties of Content Centric Networks

Transforming the current Internet architecture into a network of caches is not the only change that CCN brings to the network. There are other properties that CCN seeks and they are summarized in the following list:

- **Caching at Network Layer.** With the arrival of Content Centric Networking, the caching process is performed at the networking layer. All the content produced by every application

may be cached in CCN. P2P, HTTP or FTP traffic are treated equally in CCN. All the caching space is shared among the competitors.

- **Cache Transparency.** With cache transparency, end-users are not aware that content is served from caches. Thus, end-users receive the content from a nearest-located node while network resources are saved. CCN aims at creating more transparent caching systems which are independent of protocols and applications. All this content is going to share the same the caching capabilities and –now– it is up to CCN to determine how to manage the caching space.

- **Cache Ubiquity.** In CCN, caches are ubiquitous. Caching points are no longer fixed in particular points of the network, Caches are everywhere. Nowadays, there are more than 2.9 billion of users connected to the Internet [137] and the Internet is composed of more than 3.7 billion of hosts [15] If CCN is implemented at every node of the Internet and we suppose that every node will share, at least, $1GB$ of memory for caching. It means that $3.7EB$ will at least be available to temporary store information in the whole network.

- **Fine-Granularity of Cached Content** As mentionned, CCN names are hierarchically structured and made up of components. These components are of arbitrary length strings. Between the components, CCN stores segmentation and versioning of the packet. For example, the content `/video/Argentina/Tourism2014` may be composed of 130 chunks and 2 versions. The name `/video/Argentina/Tourism2014/124/2` refers to the chunk #124 of the second version of the video. Therefore CCN handles packets at a chunk level. As every single packet is named, every single chunk of content is named. In the current Internet, the segmentation and versioning was managed by every protocol. Bringing sequencing and versioning to the network layer opens a large number of possibilities for caching decision in the network.

To sum up, CCN brings caching capabilities to the network layer. Every single chunk is managed at Network Layer. It means that the Network Layer is aware of the sequencing and versioning of the content. In addition, CCN transform the Internet into a network of caches, placing caches everywhere. These caches are available for everybody in the network: we can store every piece of content and every user can use them. We argue CCN in-network caching features will drastically improve performance of the Internet. In the following section, we discuss the impact of performance in a network of caches and the issues to be consider in order to achieve it.

3.1.3 Improving Performance of Content Centric Networking by means of Caching

In order to achieve a good performance of the caches in a CCN network, there are many issues to be consider. These issues are summarized in the following list.

- **Replication of Content.** What and when content has to be prefetched is an important issue. It has been largely used in CDNs [47]. In CCN, prefetching and replication were studied together. These techniques were used to spread popular content to serve forthcoming requests such as Leave Copy Down (Section 3.3) or our proposed caching strategy Most Popular Caching strategy (Chapter 5). In this thesis, we tackle this issue. It is essential to discover the "best" pieces of content to be previously fetched. we believe prefetching

of content may be combined with opportunistic caching techniques in order to increase performance of caches, to ensure freshness, availability and to be reluctant against flash crowds.

- **Coordination of Caches.** Fricker et al. show that enormous cache memory would be necessary in order to achieve high caching performances [59]. Therefore it is essential to smartly decide whether to cache content and in case the cache is full, the element to be replaced respectively both in local, and distributed manners. Regarding local coordination solutions, many imported solutions from operating systems manage internal structures of the caches and are installed locally at every CCN node [97, 62] (Most Recently Used, Most Frequently Used, Last Recently Used, First-In First-Out). Regarding distributed mechanisms, MAGIC [70] proposes to maximize cache hit performance in the delivery path, by smartly selecting what element to insert and remove. ProbCache [53] seeks to insert new elements in the best place of the delivery path, having knowledge about the topological structure of the network. Thus, it is necessary to deploy solutions that coordinate resources of the network, combining internal structures of caches and the resources of all the nodes.

- **Information to be used** Caching strategies decide what to cache according to algorithms based on information about network, participants or content itself. Information includes a vast range of possibilities from popularity of content [69], content's name [61] to topological information [54]. Nevertheless, it is not clear what information may be used and what information should not be used to take caching decisions. In this thesis, we deal with this problem and we use information about popularity (Chapter 5) and we are the first to consider social networks information for construction of caching strategies (Chapter 6).

- **Caching Strategies.** The caching strategies perform the coordination of all the previously discussed issues. Caching strategies orchestrate the resources destinated to temporary storage of content. They decide the manner to manage the resources according to internal algorithms. These algorithms decide which information has to be used, the prefetching, the replication and the coordination.

In this thesis, we tackle the problem of improving the general performance of Content Centric Network by designing caching strategies. Performing caching strategies may allow to manage and organize network resources.

3.1.4 Other Research Challenges on Caching in Content Centric Networking

In addition to the previously presented issues for improving performance of Caching in CCN, there are other issues that the literature is dealing with. We organize the topics in the following list:

- **Topological Structure.** The topological structure of future Internet is expected to be just like today's Internet. As changes in infrastructure are motivated by economical incentives, it is unlikely that topological changes occur at the core of the Internet. The topologies used to evaluate the CCN caching strategies vary from k-ary trees [88] to complex ISP level topologies [68] or a combination of both [60]. However, it is not clear what is the best manner to organize the caches. Distinct topologies have shown different results [62]. It is unknown as well if particular structures are going to improve performance of CCN networks. Tyson et al. [55] analyse the behavior of P2P networks working over CCN and

show that caching at the edges of the network would allow serving 11% of the messages from the ISPs. Fayazbakhsh et al. [60] discuss the benefits of CCN. They claim that the benefits of CCN can be achieved by the simple implementation of caches in the edges of the network.

- **Organization into Autonomous System.** Similar to the current Internet structure, a future content-centric network is likely to be organized into autonomous systems (AS). AS are profit seeking entities and maintain client-provider relationships with adjacent ASes [39], which exchange information with neighbors. Pacifici et al. [40] study cooperation between CCN caches of AS. Wong et al. [43] propose a novel cooperative caching scheme to exchange content between caches. However, cooperation may be studied in a lower level without considering the existence of AS and one example of this subject is the proposal of Wu et al.[38] with their coordinated caching scheme.

- **Naming Scheme.** In CCN naming adopts a hierarchical, human-readable approach while other ICN architecture use self-certifying names. Ghodsi et al [35] have compared both approached with focus in the expected features for CCN: security, scalability and flexibility. It has not been proved if the naming scheme of CCN is the best alternative, even when we believe it is. However, the final naming scheme that all the applications should adopt has not been defined yet and only naming schemes for particular applications has been defined so far. J. Wang et al [49] defines a naming scheme for an instant messaging application. Jacobson et al. [50] show an example of NDN name space based on SIP, a VoIP protocol. Zhu et al. [37] implement an audio conference tool as well with support for ongoing conferences. However, the naming schemes proposed are specific to each application. There is not clear consensus about the right naming approach and –furthermore– it is not known if there exist an unambiguous manner to name content.

- **Named data or named functions.** Studies assume commonly that CCN is going to cache data, which represents fixed catalogs of Youtube [62], P2P traffic [55] or web traffic [60]. However, CCN caches may store other kind of information such as connection metadata or even computation. Tschudin et al [52] propose an ICN network should act like a computing machine, capable of not only caching content but also compute and cache processed and unprocessed results. Research of computation in the network appears as an interesting approach, but we are agnostic about its direct impact due to the appearance of un-deterministic situations in the network.

- **Multicast.** Current Internet does not support multicast. CCN incorporates multicast features that may have direct impact in caching performance. Saino et al. [61] revisit hash-routing techniques and forward content through alternative paths. Their research includes symmetric and asymmetric-multicast schemes which may provide substantial reduction in interdomain traffic at the cost of a limited increase in intradomain traffic. Yang et al. [41] create optimal delivery trees that use multicast to forward content. Xie et al. [42] replace the forwarding algorithm and implement a multicast reactive solution. However, Rossi et al. [62] affirm that multi-path routing may play against CCN efficiency.

- **Not cacheable data.** Many pieces of data may be stored in caches for certain time or not being cached at all. In the HTTP protocol, Adaptive TTL or Alex Protocol [36] handles the problem by adjusting a document time-to-lived based on observations from its lifetime: if a file has not been modified for a long time, it tends to stay unchanged. Adaptive TTL can keep the probability of stale documents within reasonable bounds smaller than 5% [14].

In CCN, every Data message is encrypted: its name is openly available while raw content is encrypted. The CCNx prototype incorporates a Freshness field in every data message that marks how many seconds a node should wait before marking the content as stale [114].

- **Homogeneous or heterogenous caches.** Nowadays, Internet interconnects thousands different devices. Such devices have different heterogeneous features and capabilities. It is expected that CCN is composed of heterogeneous devices with diverse capabilities. In terms of caching, many questions are open. Is there an optimal size for CCN caches? Does CCN work better with heterogeneous or homogeneous cache sizes? Rossi et al [58] study performance of heterogeneous and homogeneous cache sizes. Even when heterogeneous caches achieve better results than homogeneous caches, the gain is modest and it incorporates high complexity for managing and maintaining heterogeneous cache sizes.

- **Feasibility of implementation** Another subject of importance is the feasibility of whether today's technology is ready for CCN implementation. Arianfar et al [45] have analyzed practical features for building a CCN-capable router. Perino et al [46] evaluate suitability of existing hardware components in today's components for support of CCN and conclude that it is feasible to deploy at a CDN or ISP scale. However, components are not ready to support today a CCN network at the Internet scale. Fricker et al.[59] claim that enormous cache sizes (100 Petabytes) will be needed to achieve acceptable rates of performance in the caches. Even though, they are optimistic about the use of CCN for Video on Demand (VoD).

- **Best and wost case for CCN caches.** CCN aims at replacing IP in the future. In the future, new applications will be available and they may enjoy new CCN caching capabilities. Some studies have focused on showing and porting applications to CCN. VoCCN [50] is VoIP application which serves to show the caching capabilities of CCN caches for videoconference. NDN.JS [51] is a javascript client library that facilitates experimentation of CCN in web browsers. Wang et al [49] implement an instant messaging library to bring caching capabilities to CCN. However, the killing application for CCN has not yet been found and it is not clear if it would be possible to deploy new applications that are impossible to deploy it in the current Internet. Mathieu et al. [56] compare the use of social networks over the current IP architecture, CDN and CCN architectures. They highlight that CCN is a natural approach to develop social networks and make analytic studies on the performance of the caches.

In the hypothetical case that CCN is implemented at global scale, it is expected that the current Internet keeps the same infrastructure as today. It means that the Internet will stay organized into Autonomous Systems and its topological structures will remain the same as it is today with heterogeneous devices everywhere. Although all these are interesting problems is unlikely to change the topologies and the organization of the network due to the high economical costs.

Other problems such as definition of naming schemes and dealing with uncacheable data are extremely important for improving performance of CCN networks. However, it is out of the scope of this thesis and these subjects may demand a thesis by their own.

3.2 Replacement Policies or Caching Strategies

We study solutions to improve the general performance of CCN. The study of caches involves coordination of several caching entities and resources. It includes the physical organization of the caches, eviction policies, acceptance policies, information to be used and triggers. Structure of the caches refers to the internal algorithm of the CCN's Content Store. Eviction policy describes which piece of content should be replaced when the cache is full. Acceptance policy decides whether a piece of content should be stored in the cache. Triggers describe actions to be executed before and after caching new pieces of content. Information to be used refers to external information that helps to take decisions for acceptance, eviction or triggers. Structures of caches and eviction policies are grouped into replacement policies. Acceptance policies, triggers and information to be used are grouped into caching strategies.

In this section, we briefly summarize replacement policies and caching strategies available in the literature and explain the reason to study caching strategies in this thesis.

3.2.1 Replacement Policies

The replacement policies define the structure of the cache and the method to evict elements when space is needed. A number of replacement policies have been proposed and they focus on minimizing or maximizing various cost metrics, such as cache efficiency or perceived delay.

Replacement strategies are classified in three categories [3]: traditional replacement policies, key-based replacement policies and cost-based replacement policies.

Traditional replacement policies are largely used in common operating systems for management of memory pages or improving hard-drive accesses. *Last Recently Used (LRU)* removes the least recently referenced object. It is commonly built with double-linked lists. It is widely used in databases, memory management and hard-drive buffers [63, 65]. *Last Frequently Used (LFU)* stores the popularity values for each of the elements stored in the cache. The elements with lowest popularity values are the first to be removed. *First In First Out (FIFO)* organizes the elements in a list where every new element is attached in the front and the last element is dismissed. *Random Replacement (Rand)* removes arbitrarily content.

Key-based replacement policies are based on primary keys to evict content while ties are broken based on a secondary key. *Size* evicts the largest object. *Lowest Latency First* [6] evicts the piece of content with the lowest download latency first. *LRU-threshold* does not cache content larger than a certain threshold, *LRU-MIN* combines size and LRU policy [14].

Cost-based replacement policies implement a utility function calculated from several inputs such as time since last access, entry-time in the cache, transfer-time cost, etc. Greedy-Dual-Size [4] sets a cost function $\frac{cost}{size}$ where *cost* is the cost of fetching the document from the provider and *size* is the filesize of the object. This policy evicts the documents with the minimum cost function because it is easy and cheap to retrieve them again. Server-assisted Cache Replacement [5] uses a *price* function, which uses cost of fetching the document from the provider and the portion of caching space allocated by the object.

A great effort has been given to maximize metrics used in the replacement policies and to achieve high efficiency rates for caches, decrease average latency or maximize number of stored objects. However, performance of caches depends drastically on traffic characteristics [14]. No policy can outperform other policies under all traffic patterns. We need to look further, we can not search yield a high rate of a particular metric. In a network of caches such as CCN, caches have to cooperate and coordinate among them, smartly share information and adapt to changes in the traffic patterns. This is the aim of caching strategies.

3.2.2 Caching Strategies

Caching strategies are management policies commonly used to decide what elements must be cached. Although, their decisions include a large range of problems such as content placement, smart detection of traffic patterns, selection of information to decide and adapting to changes in traffic behavior.

The design of efficient caching strategies includes the resolution of several related problems, such as dimensioning of caches, intelligent content placement and accurate prediction of traffic patterns. In the context of CCN, the resolution of these problems involves a large number of users distributed across the planet, with diverse equipment at every node and adaptive requirements to users' needs. We argue that the construction of efficient caching strategies require orchestration of network resources. The selection of location to place content, information used to take decisions and adaptation to changes need to be coordinated.

3.3 Caching Strategies

Several caching strategies exist in the literature. They are presented in this section.

Leave Copy Everywhere (LCE)

Leave Copy Everywhere (LCE) is the built-in CCN caching strategy. In LCE, every CCN node stores a copy of every *Data* message that it processes. It has received many different names in the literature such as Ubiquitous caching [54], pervasive caching [60] and AllCache [69].

In CCN, *Interest* messages are generated every time there is a request. The *Interest* is answered with a *Data* message that follows the reverse path taken by the *Interest*. LCE stores copies of the *Data* message content in every node of the path.

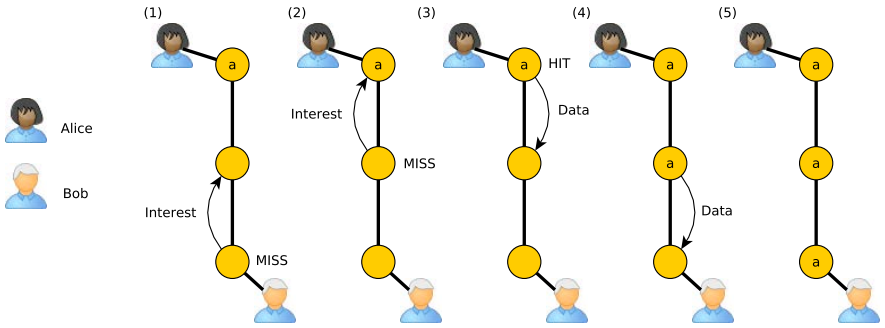


Figure 3.1: Leave Copy Everywhere (LCE) Caching Strategy

The Figure 3.1 illustrates the LCE caching strategy. As the user Bob requests the *content a*, the CCN network decides to use a path from Bob towards Alice. The *Interest* message passes by the first cache that retransmits the message to the following node after the cache reported a *miss* operation, because the content was not found locally. (Part (1)). In the next cache, the content is not present either and the *Interest* is forwarded to the third cache (Part (2)). The *content a* is found in the third cache and a *Data* message is generated which follows the reverse path taken by the *Interest* (Part 3). The strategy creates a copy in every node it passes (Part 4 and 5).

Leave Copy Down (LCD)

When a cache hit occurs, this caching strategy duplicates the element to the direct neighbor of the one where the data was found. LCD creates fewer copies than LCE and multiple copies are only produced when several requests are issued for the same content [93].

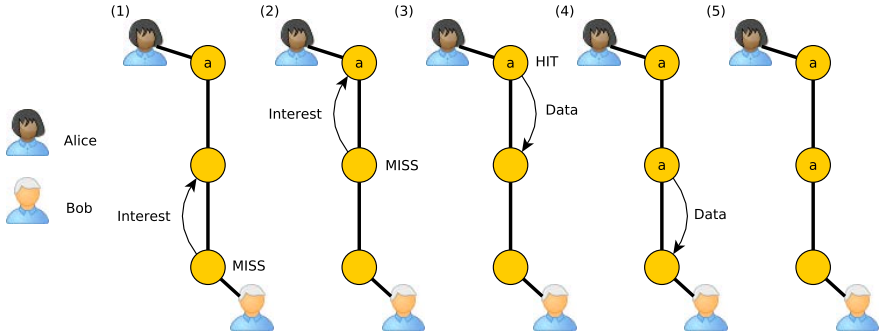


Figure 3.2: Leave Copy Down (LCD) Caching Strategy

The Figure 3.2 illustrates the LCD caching strategy. The user Bob requests the *content a*, which is located next to Alice. The *Interest* message arrives to the first cache that retransmits the message to the following node because the content is not found in the cache (Part (1)). In the next node, the content is not found either in the cache and the request is forwarded to the third cache (Part (2)). These *content a* is found in the third cache. A *Data* message is generated which follows the inverse path taken by the *Interest* (Part 3). The LCD strategy creates one copy in the immediate neighbor towards the issuer of the message (Part 4). The rest of the caches of the path stay without any change (Part 5).

MAx-Gain In-Network Caching (MAGIC)

MAx-Gain In-Network Caching (MAGIC) [70] aims at maximizing the local cache gain of inserting a new piece of content and minimizing the cache replacement penalty of removing an old piece of content from the caches. The local cache gain is the probability of adding this piece of content and that this piece of content will be requested in the future. The cache replacement penalty is the potential lost in future requests due to the replacement of the content in the cache.

As described by the authors, r_v^m is the request rate per second (the number of *Interest* messages per second at node v for content m). In MAGIC, every node records access counts of requested content in order to calculate request rates. h_v^m describes the hop counts from node v to the original server which hold content m. C_v describes the set of content in the cache v. The equation 3.1 represents the *Place Gain*, the potential gain of inserting the content m at node v. The potential loss of removing the less demanded content in node v, the *Replace Penalty* is computed in Equation 3.2. Finally, the *Local Gain* is the difference between the *Place Gain* and the *Replace Penalty* and represents the gain of inserting the content m at node v.

$$PlaceGain_v^m = r_v^m \times h_v^m \tag{3.1}$$

$$ReplacePenalty_v = \begin{cases} \min_{t \in C_v} r_v^t \times h_v^t & \text{if cache is full} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

$$LocalGain_v^m = PlaceGain_v^m - ReplacePenalty_v \tag{3.3}$$

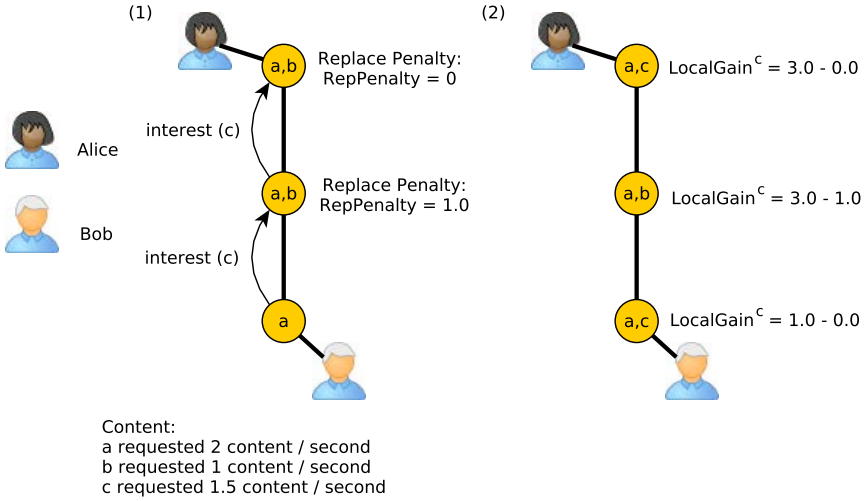


Figure 3.3: MAGIC Caching Strategy

Figure 3.3 presents an example of the MAGIC operation. In this case, there are three pieces of content a, b, c that are requested 2 times/second, 1 time/second and 1.5 times/second rates. Two of the caches are full with 2 elements, and one of them has space for one more element. *Interest* for content c is issued and it eventually arrives to the top node carrying the information. *Data* messages are forwarded through the same path and multiple replacements occur. The content c is stored at the bottom node because it has enough space to store it. In the top and intermediary nodes, MAGIC calculates local gain of inserting content c. Local gain is 3.0 at top node and 2.0 at intermediary node by replacing content b (replacing content a has implies a bigger replace penalty). As MAGIC maximizes the local gain, *content c* replaces *content b* because the local gain is bigger at the third node than at the first or the second node.

ProbCache

Probabilistic In-Network CAching (ProbCache) is a probabilistic algorithm for distributed content caching along a path of caches. ProbCache selects the best node to cache the content in a path of caches. It aims at leaving caching space for other flows and at distributing content through a path shared by several nodes [53]. ProbCache behaves differently using heterogeneous and homogeneous cache sizes.

ProbCache uses a special field into the *Interest* and *Data* messages to store distance. Every time an *Interest* message is issued, the special field is filled with a 0 that represents the number of hops travelled. When the *Interest* message is received by the second node, it increments the special field to 1. Once the content is found on a cache or the original provider, the *Interest* carry the number of hops travelled so far. Subsequently, The *Data* message is created and the number of hops is included into it. In CCN, the *Data* message travels through the requester following the reverse path taken by the *Interest* message. Every node that receives the *Data* remember its number of hop. The number of hop is distance from the origin node of the *Interest* to the current node. By dividing its number of hops by the total number of hops, it uses a random

number proportional to this value. Thus, ProbCache calculates a probabilistic index such that the probability of caching increases closer to the requester and it decreases while the distance increases.

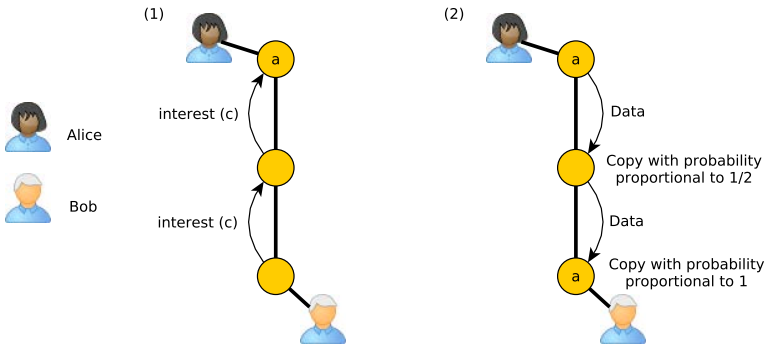


Figure 3.4: ProbCache Caching Strategy

We see an example in the Figure 3.4. The user Bob requests the piece of content a, which stored in a cache next to the user Alice. The distance from user Bob to the content a is of 2 nodes because the node will be found in the third node (i.e. distance of 2 nodes because the node has to travel through the first and second node). ProbCache calculates a probability of caching according to the distance to the node. Thus, in the first node the probability of caching will be proportional to 1 because the *Interest* has to travel through all the path to arrive to the content. The second node has to travel through the half of the distance to get the content which means it will cache with a probability value proportional to $\frac{1}{2}$. It is the same situation as shown before but this time the caching probability is calculated proportionally to the closeness to the origin of the request. At intermediary nodes the value is proportional to $\frac{1}{2}$ and is proportional to 1 at the bottom node.

Cache "Less" For More

Based on the premise that caching less content can achieve better results than caching everywhere, Chai et al.[54] proposes Cache "Less" For More. Cache "Less" For More is a centrality-based caching strategy that exploits the concepts of betweenness centrality to improve the cache gain and reduce redundancy.

Betweenness centrality is an algorithm to calculate the importance of a node within a graph [1]. It quantifies the number of times a node acts as bridge between other nodes. Betweenness centrality is applied to the topology graph and it is used to determine the best location to cache content in the path.

The equation 3.4 defines the computation of betweenness centrality. $\sigma_{s,t}$ is the number of shortest paths from s to t and $\sigma_{s,t}(v)$ is the number of shortest paths from node s to node t that passes through node v.

$$BetweennessCentrality(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \tag{3.4}$$

In Cache "Less" For More, every node is aware of its betweenness centrality value. Every time an *Interest* message is issued, the nodes appends their betweenness centrality value if it is

bigger than the currently carried in the message. Once the message has arrived to the provider or the content has been found into a cache, the *Data* message includes the maximum betweenness centrality value stored in the *Interest* message. As in CCN the *Data* message travels through the reverse path of the *Interest* message, every node checks if its betweenness centrality value is contained into the *Data* message and in case it does, the node stores the content into its cache.

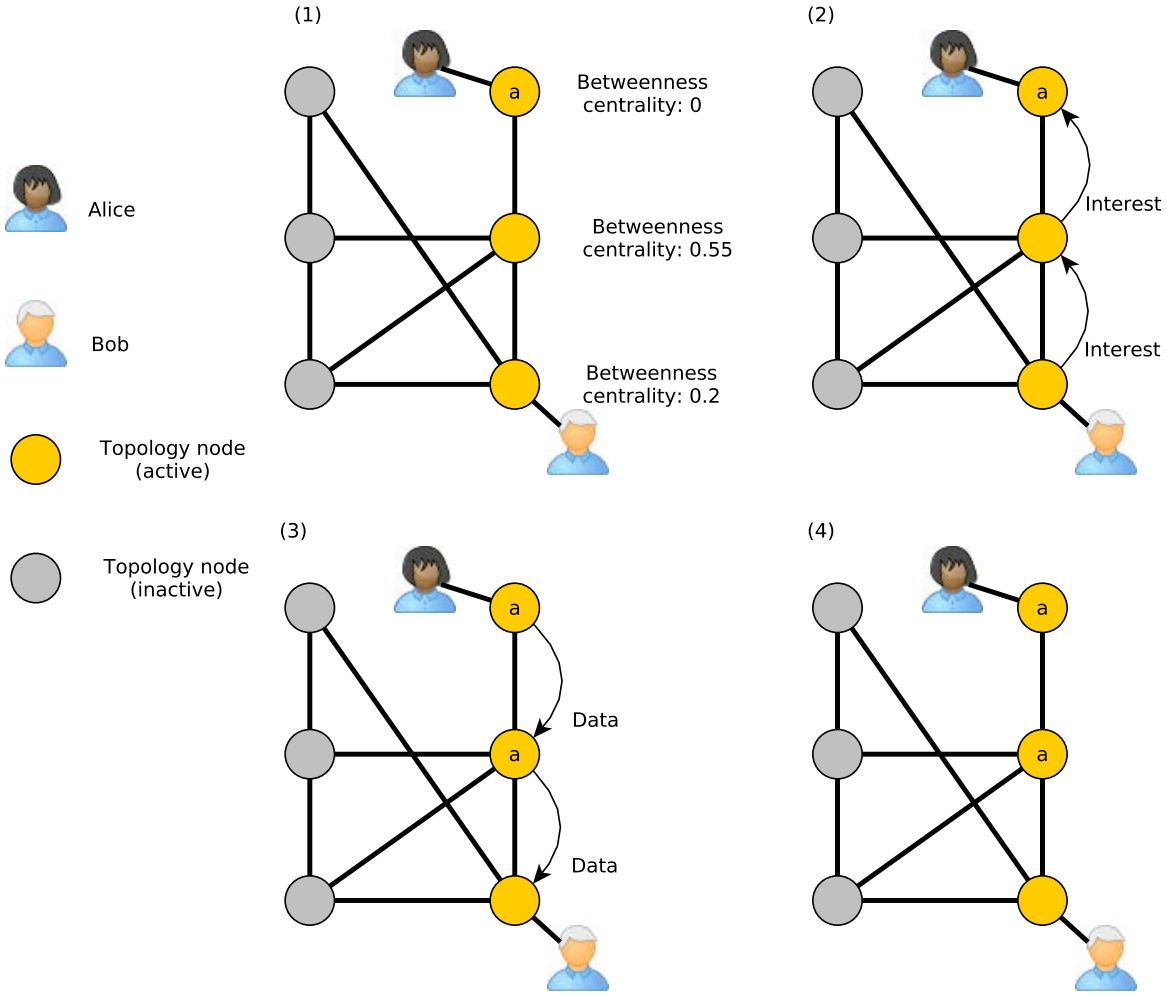


Figure 3.5: Cache "Less" for More.

The Figure 3.5 shows an example of application of this strategy. As Cache "Less" For More uses topological information, the example shows a Bob request for the content stored at Alice node. The yellow circles represent nodes that participate in the request while grey circles are nodes belonging to the topology. First, the betweenness centrality is calculated for every node in the topology. Alice's node has a centrality value of 0, while Bob's node 0.2 and intermediary node is 0.55. For the other nodes, the betweeness centrality values are 0.1, 0.05 and 0. As they do not offer important information, we hide them in the image. The intermediary node has the biggest betweenness centrality value, which means that according to the proposed scheme it will store the content. The user sends a request to retrieve *content a*. The *Interest* message arrives to a cache that stores the content and then issues a *Data* message response. The *Data* message

Metric	Alias		Equation
Caching performance metrics			
Caching Efficiency		[69]	3.11
Cache Evictions	Cache Replacement count	[53, 69, 70]	3.9
Cache Hit		[67, 68, 58, 62, 69]	3.8
Diversity		[62]	3.12
Request Cache Hit		[53, 54, 70]	3.5
Server Load	Publisher Load	[66]	3.7
Traffic metrics			
Average Link Load	Link Load, Link Stress	[69, 61]	3.20
Download Effort		[68]	
Fairness		[62]	3.17
Hop Reduction Ratio	Path Stretch, Hop Count, Average Hop Count	[67, 53, 54, 58, 62, 70]	3.13
Inter-AS Data Transferred	Inter-ISP Traffic	[55, 69]	3.18
Delay	Aggregated Network Delay	[66]	3.21

Table 3.1: Summary of the metrics available in the literature.

content is stored at the node with the biggest betweenness centrality index (In the example, the intermediary yellow node).

3.4 Caching Metrics

Due to the lack of any standard defining the use of metrics to evaluate the efficiency of caches in the network, many different metrics have been proposed and sometimes the same metrics have been re-defined with different names in different experiments and proposals. Even proposed simulators suffer from these problem and implement different versions of the same metrics [100].

In the Table 3.1, we summarize the metrics found in the literature. These metrics are classified in two categories: caching performance and traffic. Caching performance metrics describe the behavior of the caches: content found and not found in the caches, number of times the replacement policies are called, the variety of contents stored in the caches, etc. Traffic metrics measure the hops crossed, throughput among nodes, ISPs or Autonomous Systems (AS), etc.

3.4.1 Caching Performance Metrics

In order to evaluate performance of caching strategies, several metrics have been proposed.

All the metrics presented in this section use six parameters: $|R|$ represents the total number of requests (i.e., the summation of all the *Interest* messages sent by final users). w is the number of requests satisfied by the caches. $hits_i$ refers to the number of *Interest* messages that have been answered with content found in the cache of the node i while $miss_i$ refers to the number of miss operations in the cache of the node i. Finally, o_i is the number of caching operations performed at node i. C_i corresponds to the set of elements cached in node i. The notation is summarized in Table 3.2.

It is a common mistake to assume that $|R| = \sum_{i=1}^{|N|}(hits_i + miss_i)$. Assuming equality leads to wrongly compare different metrics. However, $|R| \neq \sum_{i=1}^{|N|}(hits_i + miss_i)$ and we show an example

Symbol	Description
w	Number of requests issued by end-users satisfied by caches
$\|R\|$	Total number of requests issued by end-users
$delayR_i$	Delay on request i in seconds
hit	Total number of hits
hit_i	number of *Interest* messages that have been answered with content found in the caches of the node i
$miss$	Total number of miss
$miss_i$	Total number of miss operations at node i
o_i	Number of caching operations at node i
$c_{j,i}$	Content j at cache i
C_i	All the content at cache i ($\bigcup_{j=1}^{\|N\|} c_{i,j}$)
D_i	Number of chunks (*Interest* and *Data*) that transit through node i.
$\|N\|$	Number of nodes in the network
$\|L\|$	Number of links in the network
$hops\_walked_i$	Number of hops from request i to where cache hit occurs
$total\_hops_i$	Hop count from user (request i) to server
$traffic_i$	Traffic forwarded through link i in MBytes

Table 3.2: Notation used for the metrics.

that highlights the difference in the caching performance metrics. In the Figure 3.6, Alice sends a single *Interest* message to retrieve content a. The *Interest* message passes through three CCN nodes. In the first and second CCN node, the content a is not found in the cache which means two miss operations. Finally, the content a is found at the third node (one hit operation). To summarize, $|R| = 1$ because one request for content was issued by Alice. $\sum_{i=1}^{|N|}(hits_i + miss_i) = 3$ because one hit operation occured at the third node and two miss operations occured in the rest of the nodes.

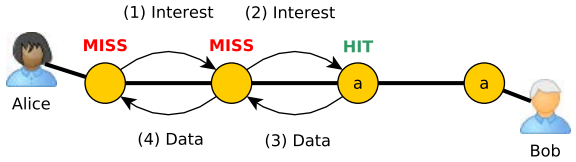


Figure 3.6: Example of CCN request to illustrate the metrics.

The first metric proposed is Request Cache Hit (RCH) [53, 54, 70] and it is shown in the Equation 3.5. SHRR refers to the percentage of requests that have been solved by the caches. In the previous example (Figure 3.6), the only request sent by Alice is solved by the third cache and $SHRR = 1$. Request Hit (RH) is related to the RCH. It refers to the number of requests solved by caches and it is shown in Equation 3.6.

$$Request\ Cache\ Hit = \frac{w}{|R|} \tag{3.5}$$

$$Request\ Hit = w \tag{3.6}$$

Server Load refers to the requests that have arrived to the server. In other words, requests that have not been resolved by the caches. It is simply represented with the difference of two numbers: total number of requests and number of requests solved by the caches. In other words, it is the complement of RH. Equation 3.7 shows the metric.

$$Server\ Load = |R| - w \tag{3.7}$$

The *Cache Hit* is the most used of the metrics to measure performance of caches. The *Cache Hit* is shown in Equation 3.8 with different forms to be expressed. In the example of Figure 3.6, there are one hit and two miss operations in the caches which means $CacheHit = \frac{1}{1+2} = \frac{1}{3}$

$$Cache\ Hit = \frac{hit}{hit + miss} = \frac{\sum_{i=1}^{|N|} hits_i}{(\sum_{i=1}^{|N|} hits_i) + (\sum_{i=1}^{|N|} miss_i)} = \frac{\sum_{i=1}^{|N|} hits_i}{\sum_{i=1}^{|N|} (hits_i + miss_i)} \tag{3.8}$$

The RCH is always bigger than or equal to the *Cache Hit* ratio. As the *Interest* message follows a path from source to the content, we can consider that it traverses a linear sequence of caches. The RCH metric does not reflect cache miss operations in a sequence of caches. If the request is resolved by the third cache, the miss operations occured in the first and second caches are not reported by the metric. The problem gets worst when the sequence of caches is longer and the request is solved by the last caches of the sequence. As we can see in the example of Figure 3.6, the user Alice requests content a stored in the third and fourth caches and miss operation of first and second cache are ignored by RCH and taken into account by Cache Hit. The Cache Hit gives $\frac{1}{3}$ while the RCH is equal to 1.

Another useful metric is *Cache Evictions*. It counts the number of times the replacement policy is called (Equation 3.9). In [70], they re-define *Cache Evictions*, which is called Average number of caching operations, and *Cache Evictions* is divided by the number of nodes. In this thesis, we use *Ratio of Cached Elements* (Equation 3.10). It is defined as a relative ratio of *Cached Elements* between two different caching strategies.

$$Cache\ Evictions = \sum_{i=1}^{|N|} o_i \tag{3.9}$$

$$Ratio\ of\ Cached\ Elements = \frac{Cache\ Evictions\ for\ Strategy\ \#1}{Cache\ Evictions\ for\ Strategy\ \#2} \tag{3.10}$$

When CCN nodes have just bootstrapped, caches are empty. While caches are being filled, there are no evictions in the caches. Rosenweig et al. [57] formally proved that caches initially empty may lead to particular results. Thus, it is always better to initialize randomly the caches and thus allows that the *Cache Evictions* is a similar number to cache insertions.

In [69], the *Cache Efficiency* metric is proposed and it is shown at Equation 3.11. It is a combination of cache operations and the averaged Cache Hit per node and it does not offer more insights than the other two metrics.

$$Cache\ Efficiency = \frac{\text{average hits per node}}{\text{sum over of all node of eviction operations}} = \frac{\frac{\sum_{i=1}^{|N|} hits_i}{|N|}}{\sum_{i=1}^{|N|} o_i} \tag{3.11}$$

Diversity is a useful metric to calculate the number of distinct elements stored at the caches. By definition, *Diversity* assumes that the cache sizes have an homogeneous size of N. *Diversity* expresses the ratio of unique content stored across all the caches. *Diversity* varies between $[\frac{1}{|N|}, 1]$. $\frac{1}{|N|}$ corresponds to caches that contain the same elements while 1 refers to all the caching space occupied by distinct elements.

$$Diversity = \frac{len(\bigcup_{n=1}^{|N|} C_n)}{\sum_{n=1}^{|N|} |C_n|} \tag{3.12}$$

To summarize, we have listed all the available metrics used in the literature to measure caching performance. *Cache Hit* is a good metric which allows to assess the efficiency of the caches. Other metrics such as RCH, *Request Hit* or *Server Load* reflect only partial information. As we consider global performance of caches, we are going to ignore them. *Cache Evictions* and *Diversity* are useful metrics to calculate the number of caching operations and how diverse the elements stored in the caches are. *Cache Efficiency* is a combination of *Cache Hit* and *Cache Evictions*. We consider that it does not offer better insights and we prefer instead to consider *Cache Hit* and *Cache Evictions* separetely. In this thesis, we have selected *Cache Hit* and *Diversity* for our experiments. *Cache Hit* reflects performance of the caches and *Diversity* describes the variety of content stored in the caches. In addition, *Ratio of Cached Elements* is used in Chapter 5 to compare *Cache Evictions* of two caching strategies.

3.4.2 Traffic

Relevant metrics of traffic in presence of caching nodes include hops crossed by the messages, the transmitted raw data and the delay.

The traffic metrics are made of several parameters: $hops\_walked_i$ refers to number of hops of request i from source to where cache hit occurs. $total\_hops_i$ refers to the hop count from user (request i) to server. $|N|$ and $|L|$ describes the number of nodes and links respectively. D_i is number of chunks (*Interest* and *Data*) that transit through node i. $traffic_i$ refers to traffic forwarded through link i in MB. Then, we serve of some notation as $src(x)$ and $dst(x)$ to describe the origin and destination of the request x and $AS(n)$ to determine the autonoumous server of node n. The notation is summarized in Table 3.2.

The *Stretch* and *Hop Reduction Ratio* are the most used metrics [67, 53, 54, 58, 62, 70]. *Stretch* refers to ratio of the complete path from client to server that the *Interest* message has travelled and it is defined in Equation 3.13. *Hop Reduction Ratio* refers to ratio of the complete path from client to server that the *Interest* message has not been crossed and it is defined in Equation 3.14. If we consider –one more time– the example of Figure 3.6, the complete path from Alice to Bob includes three links. But the *Interest* message find its content at the third node after travelling through two links. Thus, the $Stretch = \frac{2}{3}$.

$$Stretch = \frac{\sum_{i=1}^{|R|} hops\_walked_i}{\sum_{i=1}^{|R|} total\_hops_i} \tag{3.13}$$

$$Hops\ Reduction\ Ratio = 1 - \text{Stretch} \tag{3.14}$$

In [54], the *Instantaneous Hop Reduction* metric is used to calculate the evolution of hop ratio in relation with time.

Download Effort is proposed as a metric in [68]. It represents the ratio of walked hops to retrieve all the chunks of a content j over the total hops from the path between sender and

receiver. It is shown in the Equation 3.15. *Download Effort* may be seen as a Stretch for a particular content. If the number of elements in the catalog is large, its implementation becomes expensive and it does not offer information about general caching performance. We do not consider this metric in this thesis.

$$Download\ Effort_i = \frac{\sum_{i=1}^{|R|}(hops\_walked_i \times isRequestForObject(i,j))}{\sum_{i=1}^{|R|}(total\_hops_i \times isRequestForObject(i,j))} \tag{3.15}$$

$$isRequestForObject(r,j) = \begin{cases} 1 & \text{if r requests j} \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

Rossi et al. [62] propose a *Fairness* metric. *Fairness* is a metric that determines the presence of nodes that are significantly more loaded than others. We denote D_i as the number of chunks (*Interest* and *Data*) that transit through node i. *Fairness* is presented in Equation 3.17. Low values of *Fairness* indicate the presence of nodes that are significantly more loaded than others. As we are not concerned about congestion control, we do not use this metric along this work.

$$Fairness = \frac{(\sum_{i=1}^{|N|} D_i)^2}{|N| \times \sum_{i=1}^{|N|} D_i^2} \tag{3.17}$$

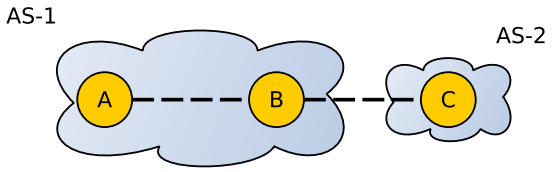


Figure 3.7: Inter-AS Traffic

Inter-AS Data Transferred refers to the data exchanged between different Autonomous Systems (AS). In the Figure 3.7, we can see an example of traffic between different AS. The nodes A and B belong to AS-1 while node C belongs to AS-2. This metric considers all the traffic sent through the link that connects the two AS –the link between B and C. We assume that for each node v that belongs to the topology there is an AS associated and we express it as as_v. Equation 3.18 is the summation of all the traffic sent from nodes that belongs to an AS to nodes that belongs to another. Equation 3.18 requires Equation 3.19. It returns the traffic if and only if the source and destination of the link belong to different AS. This metric is highly useful to measure data exchanged between AS within an ISP.

$$Inter\text{-}AS\ Traffic\ Transferred = \sum_{i=1}^{|L|} interAS(l_i) \tag{3.18}$$

$$interAS(l) = \begin{cases} traffic_l & \text{if } l \in L \text{ and } AS_{src(l)} \neq AS_{dst(l)} \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

The *Average Link Load* is shown in the Equation 3.20. It refers to the average traffic transferred through all the links of the network.

$$\frac{\sum_{i=1}^{|L|} traffic_i}{|L|} \tag{3.20}$$

Delay is a critical metric in quality of service [17]. In some articles, it has been used to show the impact of the caches in the perceived delay by the end-user. The delay is lower when the requests are solved by caches closer to the end-user. The equation 3.21 calculates the average delay perceived for every request.

$$Aggregated\ Network\ Delay = \frac{\sum_{i=1}^{|R|} delayR_i}{|R|} \tag{3.21}$$

Distinct metrics of traffic have been presented in this subsection. Most of them represent different results and they have to be chosen according to the problem to be tackled. In our thesis, we focus on caching performance and *Stretch* is the most relevant metric for us because it directly reflects efficiency of the caches: the smaller its value, smaller paths travelled by the node, which means better caching performance of the CCN network. *Fairness*, *Inter-AS Data Transferred* and *Average Link Load* are important metrics but we do not consider them because they do not represent efficiency of the caches. *Aggregated Network Delay* may be consider as a valid metric for caching performance, but we do not consider in our simulation environment because we aim at representing global performance of the network and we do not represent performance of individual users.

3.5 Summary

Nowadays, the current Internet uses largely caches to cope with its traffic. However, its architecture has not been designed for the large use of caches and it presents many limitations. Caches are designed for particular applications and protocols and these caches are located at fixed places.

The use of CCN will become the network into a network of caches. Regarding caches in CCN, the caching is moved from application layer to the network layer. Caches are available everywhere and for everybody. CCN presents a fine-granularity of the cached content.

Due to the importance of caches in CCN, cache performance is essential. In order to improve its performance, we have to consider many aspects. What, where and when store information, coordination of caches, replication mechanisms and information to be used. All these problems are managed by caching strategies and its the main subject of this thesis.

Subsequently, we described alternative caching strategies proposed in the literature. We present mechanisms based on frequency such as Leave Copy Everywhere and Leave Copy Down. Other mechanisms base on probability and fairness such as ProbCache. Targeting the same results but caching only one time the results such as Cache "Less" For More. Finally, maximizing the gain and minimizing the penalty of replacing an element in the caches with MAGIC. These caching strategies target different objectives and use different information. However, these caching strategies have never been compared before. We believe caching strategies should be compared thoroughly in order to determine best and worst cases for each strategy. Furthermore and once strategies are compared, we will be able to determine if new caching strategies are necessaries. Certainly, they are needed and in Chapters 5 and 6 we aboard the problem of designing new caching strategies based on popularity.

In addition, we have studied the metrics used to assess caches in a network. We have listed and curated the metrics in two categories: caching performance and traffic. From every category, we have selected those metrics that are the fairest to reflect performance of caches. For caching performance, we motivated the choice of Cache Hit, Diversity and Cache Evictions. For traffic metrics, we selected Stretch. These metrics are the most representatives for our problem and we use them all along this thesis.

Caching Strategies for Content Centric Networks

4

Comparing CCN Caching Strategies

Order and Progress

Brazil national motto

Contents

4.1 Survey of the Simulation Environments for Caching Strategies in CCN 58
4.1.1 Related Work 58
4.1.2 The Need for a Common Evaluation Scenario 60
4.2 Common Evaluation Scenario 61
4.3 Comparison of Caching Strategies 63
4.3.1 Simulation Environment 63
4.3.2 Results 64
4.3.3 Summary 67
4.4 Summary 68

CCN relies on in-network caching and multiple copies of the data are stored in the network. Any nodes having the data can serve future requests, and these multiple copies help reducing the load on servers and congestion in the network. Thus, many research works have proposed novel caching strategies in order to improve the performances of CCN, starting from the CCN default strategy *Leave Copy Everywhere* (LCE) [93], to more sophisticated strategies such as *Leave Copy Down* (LCD) [93], *ProbCache* [53], *Cache "Less" for More* [54] and *MAGIC* [70].

However, these caching strategies are usually evaluated within different simulation environments, using a wide variety of parameters such as different topologies, catalog, workload traces, content popularity models, cache sizes and so on. Furthermore, these proposed caching strategies have never been compared to each other, making it almost impossible to draw solid conclusions from their evaluations. In other words, it is still impossible to answer simple questions about caching strategies such as "which one really improves the overall performances of CCN in realistic scenarios", or "which strategy performs better than others with respect to the same comparison framework".

To this end, we evaluate and compare in this chapter the most relevant CCN caching strategies within a common evaluation scenario using the same simulation environment, parameters and metrics.

The contributions of this chapter are threefold:

- We first analyze the simulations scenarios commonly used in the literature and discuss the diversity of the parameters. We propose a common evaluation framework handling the most relevant parameters that are used in the evaluation process;
- We implement and evaluate using our simulation framework the five most relevant caching strategies and compare them with respects to common metrics;
- We compute the complexity of each strategy and discuss about feasibility into real implementation.

Finally, we provide our insights and finding for designing caching strategies for CCN.

The remainder of the chapter is organized as follows. We first survey the related work in Section 4.1 and emphasize on the different simulation environments and parameters used to evaluate the caching strategies. Based on this study, we propose in Section 4.2 a common evaluation scenario for comparing the caching strategies. The Section 4.3 presents the results of our comparison study and discusses the complexity of each strategy. Finally, we sum up the findings of the chapter in Section 4.4.

4.1 Survey of the Simulation Environments for Caching Strategies in CCN

4.1.1 Related Work

Since CCN was proposed in 2009 [16], CCN has become a very active research topic with a lot of new proposals published in major conferences. Some proposals deal for instance with the content namespace [35], the routing scheme [30] or the congestion control [12], but most of the works focus on the Caching Strategy as caching is the main milestone of CCN and its in-network caching capability.

Regarding caching strategies, authors usually evaluate their proposals using their own simulation environment, scenarios and select their configuration parameters at their own will. This makes it impossible to compare the different caching strategies among them.

In this section, we extract from the related work the diverse simulation environments in the area of CCN caching and we keep four main parameters of the simulations: topology, content catalog, content popularity model, and cache size. We then discuss the values of these parameters in the literature. Finally, we highlight the need for defining a common evaluation framework to accurately evaluate the CCN caching strategies.

Topology

In the literature, the topologies used to evaluate the CCN caching strategies vary from k-ary trees to complex ISP level topologies or a combination of both: Psaras et al.[53] uses a 6-level binary tree (i.e. 127 nodes) where all the requests are issued from the last two levels of the ISP and are served by the root node. Chai et al.[54] and Ren et al. [70] use the same kind of a k-ary tree ranging from 4 to 6 levels and the number of children per node varies between 2 and 5. Fayazbakhsh et al. [60] uses a mixed approach, where an ISP level topology is generated and a binary tree of depth 5 is built at each node of the ISP level topology.

Rossi et al. [62, 58] uses the educational network Abilene, and other ISP level topologies such as GEANT, DTelecom, Level3 and Tiger. Rossini et al.[68] considers a 4x4 torus topology and ISP level topology (GEANT). Wang et al.[67] uses RocketFuel to generate ISP level topologies

in particular AS1755 (eBone) and AS3967 (Exodus) and builds on BRITE to generate two other hierarchical topologies. Rossini et al.[68] and Saino et al.[61] only consider the GEANT topology.

There is no overall consensus on the topology to be used in simulation environments to fairly evaluate the CCN caching strategies. It is however a crucial point since the topology has a direct impact on the results [62].

Content Popularity Model

The content popularity model is a function that establishes the popularity of every piece of content, i.e., how often every single piece of content is going to be requested. Previous measurement studies show that users are attracted by only a few web sites, while they give little or no attention to millions of others. Therefore, the content popularity can be commonly modeled with a probability distribution function such as a power law.

Zipf and its generalization Maldelbrot-Zipf (MZipf) distributions (Equation 4.1) belong to the power law distribution and are commonly used in most of the models or simulations [71, 62]. Its α parameters refers to the slope of the distribution while q describes its plateau. While q tends to zero, we obtain a flatter distribution. In the literature, the (M)Zipf α parameter ranges from 0.6 to 2.5. For instance, the catalog of the PirateBay is modeled with $\alpha = 0.75$, DailyMotion with $\alpha = 0.88$, while the VoD popularity in China exhibits an α parameter ranging from 0.65 to 1.0 [59]. (M)Zipf is commonly used in different simulation environments such as ccnSim, or Content Delivery Networks (CDNs) [60] to model the load of web servers.

$$pmf(x, N, q, \alpha) = \frac{(q+x)^{-\alpha}}{\sum_{k=1}^{N}(q+k)^{-\alpha}} \tag{4.1}$$

Experiments based on MZipf distribution assign probabilities to a fixed catalog of content. Thus, every time a piece of content is requested, it is selected with a probability value. Depending on the configuration of the α parameter, some pieces of content are more likely to be requested than others.

In the literature, we found a wide range of values for the α parameter. Psaras et al.[53] uses a Zipf popularity model with α of 0.8. In [54], this parameter is 1.0 while Saino et al. [61] varies it between 0.6 and 1.1. In [68, 62], α ranges between 0.65 and 2.5. The α varies between 0.7 and 0.96 in [67] while Ren et al. [70] evaluate with an α between 0.7 and 1.1.

Other authors have resorted to real traces: Fayazbakhsh et al.[60] evaluates the caches with traces extracted from the Akamai CDN Asia; Ming et al.[66] resort to traces from wdklife.com, a Chinese entertainment feed.

As (M)Zipf distribution is the best candidate to model content popularity, there is no consensus for the value of its α parameter and we saw that this value can range from 0.6 to 2.5 and will have an important impact on the evaluation of the caching strategies.

Catalog

The catalog represents the entire collection of elements (i.e.: content) in the network. The number of requests for a certain piece of content depends therefore directly on the content popularity model. Existing work use a wide range of values for the catalog size: Rossi et al.[68, 62, 58] consider a Youtube-like catalog with 10^8 pieces (10 MB each). Ren et al. [70] use a catalog size of only 10^4 pieces. Differently, Ming et al.[66] use 8×10^4 pieces of content for its catalog and Chai et al.[54] uses only 10^3. Surprisingly, [53, 60, 61, 67] do not mention the size of the catalog; [53, 60, 61] only detail that they generated 10^5 requests for all the catalog.

Cache Size

The cache size determines the space available in every CCN node for storing temporally pieces of content. This size is usually expressed with an absolute value or a ratio with regards to the catalog size (i.e. 10 elements or 0.01 of the catalog size respectively). In order to compare works together, it makes more sense to use a ratio for the cache size as most of quoted works use different catalog sizes.

Again, there is a wide different size of cache in the literature, with size ranging from 2×10^2 to 10^5 elements. Saino et al.[61] experiment values of 0.2 and Chai et al.[54], 0.1. In [67], it ranges between 0.01 and 0.06 while Ren et al.[70] tests values from 0.0055 to 0.005. Rossi et al.[62] uses the larger difference between cache size and catalog with ratio of 10^5.

4.1.2 The Need for a Common Evaluation Scenario

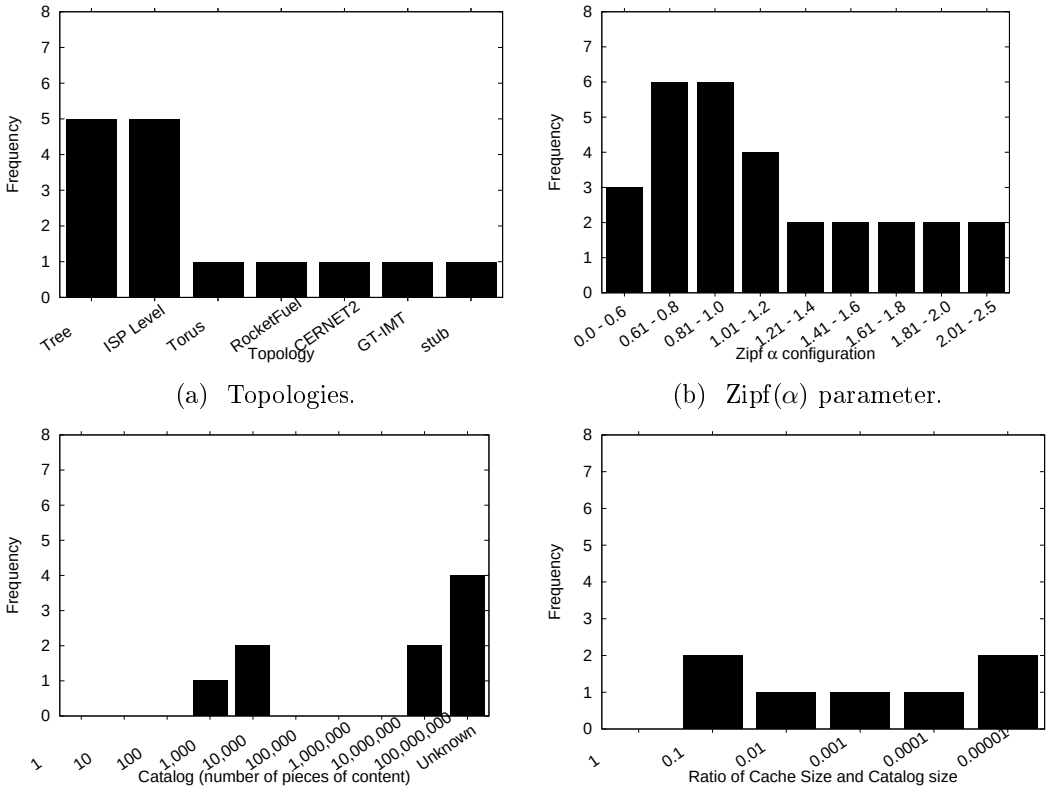
By presenting the related works on CCN caching, we point out that most of these quoted works rely on very different simulation environments and parameters (topologies, catalog size , cache size, content popularity).

In the Figure 4.1, we plot the histograms of the four parameters for all the presented papers: α parameter of the popularity model, topology, catalog size and cache size. From this figure, it is clear that there is no consensus for any specific parameter values. We observe instead a vast range of different values for all these parameters. More precisely, to assert diversity on the parameters, we resort to Gini-Simpson index. This statistical index is commonly used in Biology to represent diversity in a sample. Tuomisto[99] states the Gini–Simpson index as $1 - \frac{1}{2D}$ which quantifies the probability that "two individuals picked at random from the dataset do not represent the same species". By computing the Gini-Simpson index for the parameters, we obtain 0.84 for the Zipf α parameter, 0.76 for the topologies, 0.80 for the catalog and 0.76 for the cache size. Neither of these index is less than 0.75 which means most of the articles do not share the same values for the same parameter.

We can also correlate parameters for each paper. To this end, we compare catalog, cache size, topology and popularity model of every paper against each other. The Table 4.1 shows a correlation matrix where columns and rows indexes are the presented research papers. Every cell gives the number of common parameters for two papers. As we focus on the four main parameters, the cell value ranges from 0 to 4. For example, the cell value for the fourth and the fifth column is 2 ([60] and [70]): it means these two papers share two parameters. The Table 4.1 shows that only 2 papers share exactly the same simulation environment with the four parameters in common. Only 33% of papers share more than two parameters and the large majority of papers have only one or no parameter in common.

The papers presented in this related work do not share the same configuration of parameters. Most of these caching strategies have been proposed because they show good performances according to common evaluation metrics such as Cache Hit or Diversity, but it is impossible to compare these strategies between them. Indeed, their simulation environments are too different and a single parameter (topology, cache size etc.) can have an important impact on the overall evaluation. Even though these last 6 years led to considerable amount of work, it is still impossible to fairly compare caching strategies. Note that this work has already begun with the standardization efforts performed by the ICN Research Group at the IRTF. Indeed, they started providing guidelines for evaluating scenarios and in particular in case of mobility, routing and caching [87].

In this study, we follow the same trend and propose a general evaluation scenario for CCN



(a) Topologies. (b) Zipf(α) parameter.

(c) Catalog. (d) Cache Size.

Figure 4.1: Simulation environments in the literature.

(Section III). Based on this common framework, we perform a comparison study of the best caching strategies that have been proposed to date. This study will allow pointing out the main advantages and drawbacks of these caching strategies.

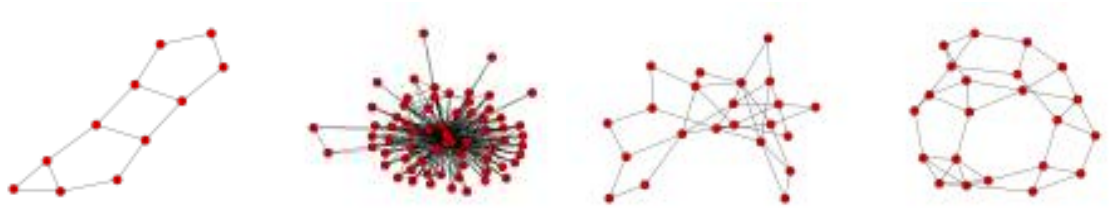
4.2 Common Evaluation Scenario

We define a common evaluation scenario that relies on the most common parameters used across the literature in order to compare all the caching strategies. Regarding the topology, we consider a Future content-centric Internet built on the CCN architecture. This new Internet will still be organized with ISPs and the topological structure of this Future Internet will be just like today's Internet. ISP topologies are therefore the best candidates for evaluating strategies compared with trees or torus topologies. We choose four ISP level topologies: Abilene, Dtelecom, GEANT and Tiger and we depict them in the Figure 4.2 [62]. Many research works use these topologies as seen in Fig. 4.1a

With regards to the content popularity model, we use a popularity model based on Zipf probability distribution. The α parameter of the Zipf varies between 0.65, 1.1, 1.5 and 2.0. The 0.65 value refers to a low popularity scenario when the probability of selecting a piece of content is close to a uniform probability function. The 1.1 value stands for a normal popularity scenario. Finally, we consider two high popularity scenarios with α equal to 1.5 and 2.0. These high popularity values are evaluated to be consistent with the literature. In the rest of this section, we refer to the scenario as *low*, *normal* and *high popularity* scenario.

Art.	[54]	[68]	[61]	[60]	[70]	[67]	[62]	[58]	[66]	[53]	[69]	[55]	[59]
Chai et al.[54]	N/A	1	1	2	3	1	1	1	0	1	0	0	0
Rossini et al. [68]		N/A	2	2	1	1	3	2	0	1	1	0	2
Saino et al. [61]			N/A	2	1	2	3	2	0	1	1	0	2
Fayazbakhsh et al. [60]				N/A	2	2	3	1	0	1	1	0	1
Ren et al. [70]					N/A	1	2	1	1	1	0	0	3
Wang et al. [67]						N/A	2	1	0	1	1	0	2
Rossi et al. [62]							N/A	4	1	1	2	0	3
Rossi et al. [58]								N/A	0	0	0	0	2
Ming et al. [66]									N/A	0	0	0	1
Psaras et al. [53]										N/A	1	0	2
Cho et al. [69]											N/A	0	2
Tyson et al. [55]												N/A	1
Fricker et al. [59]													N/A

Table 4.1: Cross-comparison of environments in the literature. The cell values can range only from 0 to 4 (number of parameters in common).



(a) Abilene. (b) DTelecom. (c) GEANT. (d) Tiger.

Figure 4.2: ISP-level topologies.

As catalog size, from Fig. 4.1c, 10^4 elements was the more frequently used value. However, this is a very limited catalog size. It is difficult to imagine the Internet with only 10^4 pieces of content. We thus consider in our experiments a catalog of 10^6 elements, as it is also a common value. The cache size will vary from 1 to 1,000 elements. We express the cache size as a division between the absolute cache size and the catalog size, and We call it *Cache Size Ratio*. The ranges of cache sizes between 1 and 1,000 correspond to 10^{-6} to 10^{-3} using *Cache Size Ratio*. We consider homogeneous cache sizes (all nodes have the same cache size). The Cache Replacement Policy (CRP) used in the comparisons is Last Recently Used (LRU) because Rosenweig et al. [57] show that CRP may be grouped into the same equivalent class, i.e., different CRP will exhibit the same performance.

Catalog	
Catalog	10^6
Popularity Model	MZipf($\alpha = \{0.65; 1.1; 1.5; 2.0\}, \beta = 0$)
Network	
Topologies	Geant, Dtelecom, Abilene, Tiger
CCN Cache Configuration	
Cache size ratio	$\{10^{-6}; 10^{-5}; 10^{-4}; 10^{-3}\}$

Table 4.2: Common Evaluation Scenario.

We summarize the chosen parameters of the scenario in the Table 4.2.

4.3 Comparison of Caching Strategies

4.3.1 Simulation Environment

In order to compare caching strategies for CCN under the same simulation environment and the common evaluation scenario, we implemented a discrete-event simulation tool written in Python. It is freely available at [130]. This simulation tool needed to be able to get all the parameters and scenarios we have presented throughout this chapter. Then we implemented the five previously mentioned caching strategies in our simulator.

A general overview of the simulator architecture is presented in the Figure 4.3. The simulator represents the interaction of users into a CCN network. It receives as input a synthetic trace (I.e. SONETOR trace). It generates the results by processing the interactions of users into a network of caches.

The simulator is composed internally by the following modules:

- The **Topology Manager** handles the topology of the CCN network and the position of the users. The topology manager creates a two-dimensional plane where the CCN nodes are located according to the configuration of the topology. According to the chosen trace file, the users are located into the 2D plane and attached to the closer node in the plane. This module handles as well changes in the topology and updates on the position of its users.

- The **Cache Manager** handles the caches placed in the topology manager. The cache manager sets the replacement policies at every node and installs and process the caching strategies to be used in the experiments.

- The **Engine** executes the activities of the trace in a discrete-manner. It advertise changes to the topology manager and reports the cache manager to publish and to retrieve pieces of content.

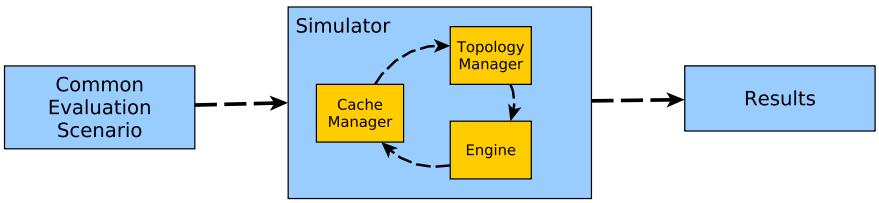


Figure 4.3: Simulator architecture

We then perform simulation experiments for the five caching strategies by varying the parameters of the common evaluation scenario (e.g.: cache size, content popularity, etc). For each simulation experiment, three runs are performed and confidence intervals are computed.

The next section presents the metrics to evaluate the performances of the caching strategies. The metrics used to assess the performances of the caches are *Cache Hit*, *Stretch* and *Diversity*. All these metrics have already been introduced in the Chapter 3. We summarize the simulation environment and the evaluation metrics in the Table 4.3.

Simulations	
#Runs	3
Simulated Time	1 day
Metrics	Cache Hit, Stretch Diversity, Complexity
CCN Cache Configuration	
Replacement Policy	LRU
Caching Strategies	{LCE, LCD, ProbCache, Cache "Less" For More, MAGIC}

Table 4.3: Simulation Environment

4.3.2 Results

Complexity

By implementing the caching strategies into our simulator, we also compute their complexity, analyze and compare their worst case performance. The complexity for the functions *insert*, *delete* and *lookup* in LRU implementations is $\mathcal{O}(1)$ using a hash table [129]. We analyze the CS complexity in a normal CCN interaction: an *Interest* message is sent through the network and a *Data* message is received as an answer to the message. We refer to m as the walked hops to retrieve the content and n as the cache size. The walked hop refers to the number of hops crossed to retrieve the content. The results are presented in the Table 4.4.

As stated before, LCE stores a copy of the *Data* message in every node it has passed. In the best case, the content is found in a CCN cache without forwarding the *Interest* and it's complexity is the cost of a lookup operation: $\mathcal{O}(1)$. In the worst case, the *Interest* travels through the whole path (length m) to the origin server. Its complexity is the cost of a lookup operation in every node and then the cost of a store operation in every node: $\mathcal{O}(LCE) = \mathcal{O}(m \times 1 + m \times 1) = \mathcal{O}(2 \times m)$. The LCE complexity is presented in Table 4.4 (first row).

LCD is a special case of LCE. Content is copied in the direct neighbor towards the requester which implies only one insert operation. In contrast to LCE, the cost is reduced to check all the caches in the path and storing only once in the direct neighbor where the content was found (see the second row in Table 4.4).

The MAGIC strategy aims at minimizing the cost of replacing an element. To do that, MAGIC needs to lookup all the caches on the path from the requester to the provider (m lookup operations) and then to traverse every element of every cache looking for the minimum replacement penalty (m operations of cost n, where n is the cache size) [70]. The Third row in Table 4.4 shows its computational complexity. As caches are expected to be bigger than 10^6 elements, the cost of MAGIC is expected to skyrocket and to decay its performance.

Cache "Less" For More uses information about the topology. This information is calculated off-line, and it has a cost of $\mathcal{O}(|L|^2)$. However, this is computed only once and then the complexity of the algorithm is the same than LCE adding the cost of looking for the best place to cache the content (a comparison made at every node in the path). The complexity is depicted in the fourth row of the Table 4.4).

In ProbCache, the distance from requester to provider is calculated during the transmission of the packet. At every hop, the distance increases by one. Probcache stores the content at the best location following a similar approach to the one in Cache "Less" For More. The complexity is similar to the previous one and can be found in the fifth row of Table 4.4.

Caching Strategy	Complexity
Leave Copy Everywhere (LCE)	$\mathcal{O}(2 \times m)$
Leave Copy Down (LCD)	$\mathcal{O}(m + 1)$
MAGIC	$\mathcal{O}(m \times (1 + n)$
Cache "Less" For More	$\mathcal{O}(3 \times m)$
ProbCache	$\mathcal{O}(3 \times m)$

Table 4.4: Computational Complexity for Caching Strategies

From Table 4.4, we observe that four caching strategies show a linear cost depending on the length of path to the content and MAGIC has the biggest computational cost. MAGIC is the only caching strategy, which depends on two variables (cache size and path). For example with caches of one million elements, MAGIC requires to run through m millions elements while other strategies requires approximately to perform only $3 \times m$ operations.

Figure 4.4 shows the comparison of the caching strategies in the common evaluation scenario. This figure is divided into 3 lines and 4 columns: Each line stands for three metrics to analyze the caching strategies (Cache Hit, Stretch and Diversity); each column stands for a different content popularity (Zipf α parameter). On each plot, the x-axis is the cache size and y-axis is the metric. For each value, each experiment has been performed three times (Table 4.3) and we show the confidence intervals.

Cache Hit

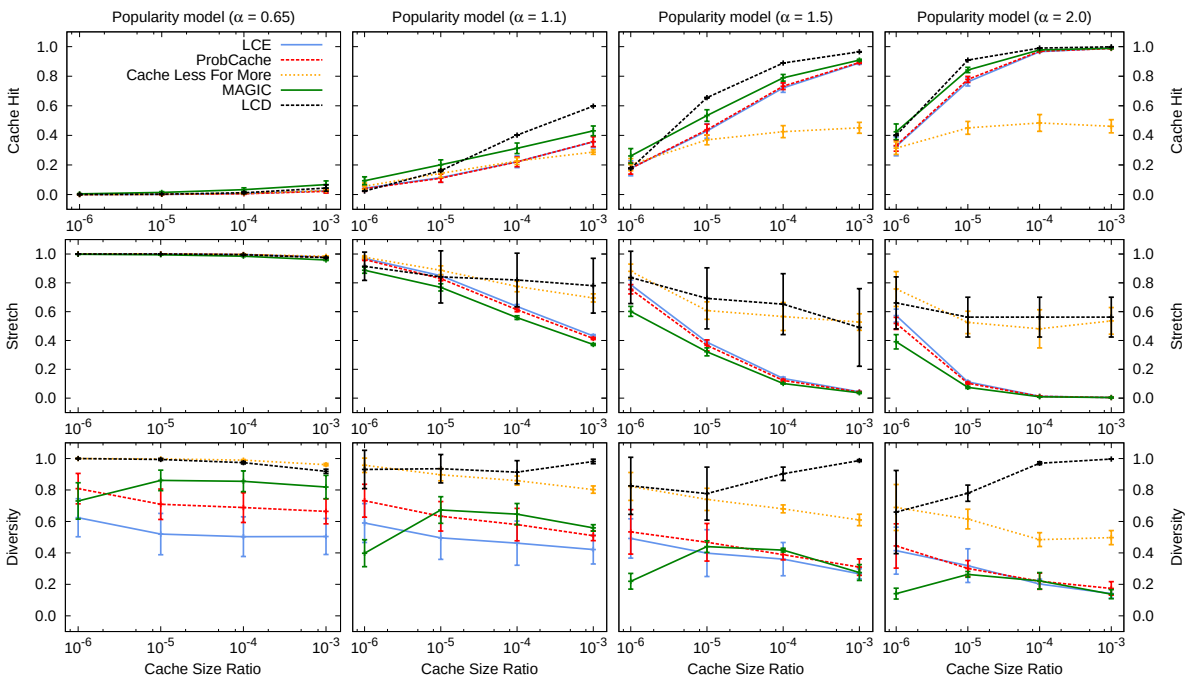


Figure 4.4: Comparison of the Caching Strategies

In the first line of the Figure 4.4, we compare the Cache Hit of the caching strategies in

different environments ranging from low to high popularity scenarios (α=0.65, α=1.1, α=1.5 and α=2.0). We should pay special attention to smaller cache sizes. The bigger the cache size, the easier to achieve good caching results. However, the goal is to minimize spent resources (cache size) and to maximize caching efficiency. For example, if we consider a catalog of 10^{13} pieces, a cache size of 10^{-6} pieces refers to a cache size of ten millions elements. While cache size ratio of 10^{-3} refers to ten billion elements.

In terms of Cache Hit, MAGIC and LCD show the best results and outperform the other strategies. In low popularity scenarios, MAGIC performs better with smaller cache sizes and it get overcome by LCD when the cache size increases. In the first column and first row of the Figure 4.4, the green line of MAGIC is over all the other lines. Its *Cache Hit* is superior to those of the other caching strategies. In the second column of the same row, the green line is placed over all the other lines for small cache sizes (10^{-6}, 10^{-5}). It means MAGIC cache size is greater than the others. It gets surpassed by the black dotted line of LCD when the size of the cache increases to 10^{-4}, 10^{-3}. For all the other cases, MAGIC and LCD always surpass the *Cache Hit* of the other caching strategies.

LCE and ProbCache show almost the same level of performance. The blue line of LCE and the red line of ProbCache are always tied together as we can see in the Figure 4.4. It means its *Cache Hit* ratios have always similar values. However, ProbCache performs fewer operations than LCE and is therefore a better candidate than LCE (computational complexity).

Cache "Less" For More strategy has shown different results. In the scenarios with low popularity ($\alpha = 0.65$ and $\alpha = 1.1$), especially for small-size caches (10^{-6} and 10^{-5}) Cache "Less" For More reaches the same level of performance as other strategies. It is difficult to observe in the first row and column that Cache "Less" achieves a good level of performance because all the lines are located close together. Cache "Less" For More is located between MAGIC and the rest of the caching strategies. In the second column, we observe that for 10^{-6} caches Cache "Less" For More performs better than other strategies and only MAGIC improves its *Cache Hit* ratio. The yellow dotted line is placed below MAGIC but over the other caching strategies. However, with high popularity and larger cache size, it performs badly compared to others ($\alpha = 1.5$ and $\alpha = 2.0$). As we can see in the Figure 4.4, the yellow dotted line of Cache "Less" For More is always below the *Cache Hit* results of the other caching strategies.

MAGIC and LCD have shown the best Cache Hit performances. Due to its high complexity cost, MAGIC may be used as a boundary function for other caching strategies (see Section 4.3.2). According to our results, LCD should be implemented as a replacement for LCE as the CCN default caching strategy. LCE and ProbCache present similar results, however ProbCache performs less operations. Finally, we consider that the Cache "Less" For More strategy is useful in scenarios with low and normal popularity and with cache sizes smaller than 10^{-5}.

Stretch

The Second line in the Figure 4.4 shows the Stretch. In low popularity scenarios ($\alpha = 0.65$), all the caching strategies performs badly. All the caching strategies traverse between 97% and 100% of the path to retrieve the content. In the figure, the *Stretch* results are stacked close to 1. Still, MAGIC performs better than the rest of the strategies but its performance is kept at low level. With a cache size of 10^{-3}, Stretch is at most 97%. It means that, by definition of the metric, the traveled path is only reduced by 3%.

In normal popularity scenarios ($\alpha = 1.1$), for cache size smaller than 10^{-5}, all the strategies show similar performances. For cache size larger than 10^{-5}, we can distinguish two classes of

strategies: MAGIC, LCE and ProbCache have the same level of performances and outperform LCD and Cache "Less" for More. Overall, regarding the Stretch, MAGIC is the best caching strategy and reduces the length of the path by 88% (cache size 10^{-6}) and to 33% (cache size 10^{-3}).

In high popularity scenarios ($\alpha = 1.5$ and $\alpha = 2.0$), the level of performance of MAGIC, ProbCache and LCE increases and tends to converge to exactly the same results. LCD and Cache "Less" For More still performs badly and are overcame by 40 percentage points with a large cache size (10^{-3})

MAGIC shows the best results for *Stretch*. In normal and high popularity scenarios, MAGIC outperforms all the strategies when the cache size is less than 10^{-5}. When the cache size increases, LCE and ProbCache reach the same level of performance as MAGIC.

Note that – as we stated in the results of complexity of every caching strategy – the complexity of MAGIC is proportional with cache size and its computational cost will be too expensive for large cache size.

Diversity

The third line in the Figure 4.4 shows the Diversity. The caching strategies can be grouped into two classes: High Diversity or Low Diversity. By its design, a caching strategy may generate multiple replicas of the same content, resulting in a Low Diversity into caches. Or, it may limit the number of replicas, resulting in the High Diversity into caches. Thus by definition, MAGIC, ProbCache and LCE belong to the Low Diversity class, whereas Cache "Less" For More and LCD are in the High Diversity class.

In the Low Diversity class, LCE shows lowest Diversity and can be considered as an appropriate caching strategy if low diversity is expected in the network. Once the popularity or the cache size increase, the gap between the three strategies of this class decreases and their Diversity tends to the same value.

In the High Diversity class, with the low popularity scenario ($\alpha = 0.65$), Cache "Less" For More outperforms LCD. Once the popularity increases ($\alpha \geq 0.65$), LCD becomes the strategy with the highest Diversity and it tends to the maximum value (1.0). In contrast, when popularity increases, Cache "Less" For More Diversity tends to decrease and does not follow the same behavior as LCD.

4.3.3 Summary

A rough analysis of the results may suggest that MAGIC is the best caching strategy according to Cache Hit and Stretch. However, the analysis of its complexity showed that MAGIC is an expensive caching strategy. MAGIC does not scale because the computational cost grows proportional to the distance of the path and the cache space. MAGIC can still be used as a boundary function for the further evaluation of future proposals.

Then, choosing the best strategy depends on the level of Diversity expected in the CCN network. In fact, the CCN network may contain diverse content or not: it depends on the needs of the network. For example in disaster scenarios [10], Low Diversity causes high number of replicas and it is an essential property to continue operating properly in the case of failures. Moreover, in some case, multiple replicas (i.e.: Low Diversity) may serve content that is originally located in an unavailable server, while in another case it can be considered as waste of cache space. Otherwise in case of low popularity of content ($\alpha \leq 0.65$), a High Diversity is more appropriate because users are more likely to request distinct content rather than the same content.

For a Low Diversity, ProbCache and LCE appear as good strategies. They also show similar results in terms of Complexity, Cache Hit and Stretch.

For a High Diversity, there are two cases: for small cache size and low popularity (up to 10^{-5} and 1.1 respectively), Cache "Less" For More is the best candidate as it achieves good performances in terms of Cache Hit and Stretch while still having a low complexity cost. Otherwise, we suggest resorting to LCD strategy, which overcomes all the other caching strategies.

4.4 Summary

In this chapter, we compared the most relevant CCN caching strategies. We have defined a common evaluation scenario to evaluate the strategies under the same environment. Then, we implemented the LCE, LCD, ProbCache, MAGIC and Cache "Less" for More strategies and computed their complexity.

We showed on a case-by-case basis what caching strategy is appropriate for each scenario. For instance MAGIC outperforms other caching strategy but its computational cost is very high and it can be expensive to be implemented in real CCN nodes. However, MAGIC could be considered as a boundary function for further comparison and evaluation. Even though LCD and Cache "Less" for More are good candidates to be used as caching strategy for CCN, LCE and ProbCache are more appropriate with low Diversity scenarios while LCD and Cache "Less" for More are better candidates with high Diversity. Selecting the best strategy depends on the objectives of the CCN network.

We have fairly compared the caches and determine the best case for every one of them. During this analysis, we have discovered that MAGIC and LCE overcome other caching strategies in terms of *Cache Hit*. However, based on some concepts of these caching strategies, better caching strategies that outperform current strategies can be designed. These concepts include the popularity count used within MAGIC and the use of replication of content found in LCD. Such new strategies are presented in the next chapters.

5

Popularity-based Caching Strategy for CCN

Contents

5.1 Most Popular Caching (MPC) Strategy 70
5.1.1 MPC Example Scenario 70
5.2 Experimental results and analyses 71
5.2.1 Simulation Environment 71
5.2.2 MPC parameters tuning 72
5.2.3 MPC vs. LCE 74
5.3 Discussion 76
5.4 Summary 77

In CCN, the content is not retrieved from a dedicated server, as it is the case for the current Internet. The premise is that content delivery can be enhanced by including per-node-caching features as content traverses its distribution path across the network. Content is therefore replicated and located at different points of the network, increasing availability for incoming requests.

From the comparison of the caching strategies in the Chapter 4, we have discovered that new caching strategies can be proposed. There are some concepts presented in the literature which have not been fully exploited such as popularity and replication of content. The MAGIC caching strategy uses the popularity of content to maximize the gain of inserting a new element into the caches. The LCD caching strategy uses replication of content to create copies of content found in the caches. These two concepts can be combined to create a new caching strategy.

The contributions of this chapter are the following:

- We propose Most Popular Caching (MPC), a caching strategy that purges the unpopular content from the caches. MPC achieves it by caching only popular content. MPC is based on the premise that if we store content more likely to be consumed, is unlikely to spend resources to remove content from the caches. By caching only popular content, MPC expects that CCN architectures are easier to manage by executing less caching operations.

- With a thorough simulation process, we tune the internal parameters of MPC and then, we evaluate the caching strategy against the previously presented LCE.

The reminder of the chapter is organized as follows. Section 5.1 describes MPC, our new caching strategy for CCN with an illustrative example. Section 5.2 presents the simulation environment, the tunning of MPC and the performances of MPC in several evaluations. Finally, we discuss the work and we draw some conclusions in Sections 5.3 and 5.4.

5.1 Most Popular Caching (MPC) Strategy

We present Most Popular Content (MPC), our new caching strategy designed for CCN networks. Instead of storing all the content at every node on the path, MPC caches only popular content. MPC caches less content than LCE caching strategy but still improves in-network caching performance while -at the same time- decreases resource consumption.

In MPC, every node counts locally the number of requests for each content name, and stores the pair *(Content Name; Popularity Count)* into a *Popularity Table*. Once a content name reaches locally a *Popularity Threshold*, the content name is tagged as popular and if the node holds the content it suggests its neighbor nodes to cache it through a new *Suggestion* primitive. These suggestion messages may be accepted or not, according to local policies such as resource availability.

As the popularity of a content can decrease over time, the Popularity Count is reinitialized after the suggestion process. This reinitialization consists in decreasing the popularity value to a pre-defined *Reset Value* and it serves to prevent flooding the same content to neighbors.

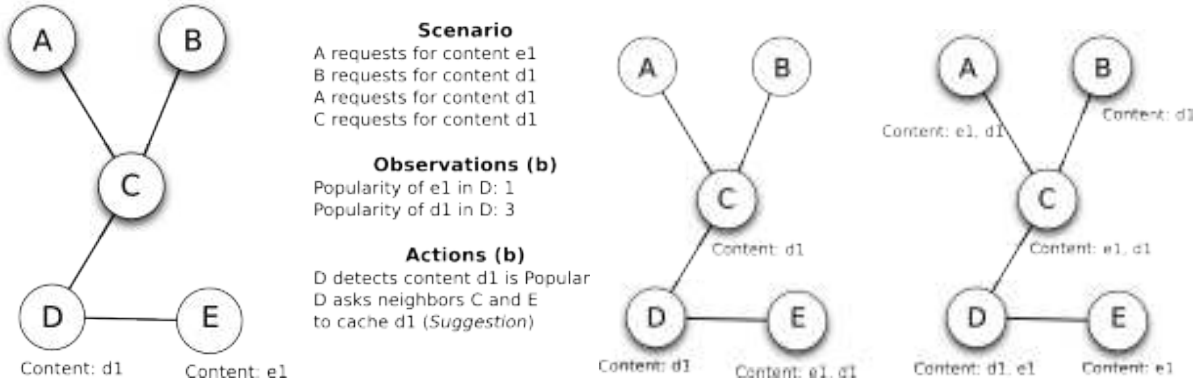
MPC strategy has a direct impact in the requirements of the CCN node. In addition to CCN cache space required, MPC needs an extra space to store the *Popularity Table*. For instance, keeping one million entries in the table means $1GB$ of RAM memory using $1023B$ per content name and $1B$ for the Popularity Count (we used fixed length for the name to simplify calculation). We expect to share dedicated caching memory with MPC tables in order to perform fair comparisons.

5.1.1 MPC Example Scenario

We present in this section an example to clarify the MPC workflow. In this example, nodes request content located around the network and we explain how the MPC strategy works in comparison with the default LCE strategy.

The example is depicted in Figure 5.1 and it is composed of three parts. Figure 5.1(a) describes the scenario and the sequence of content requests. Figure 5.1(b) and 5.1(c) show the final states after applying MPC and CCN caching strategies respectively. In our scenario, three nodes A, B and C request popular content d_1 initially stored at node D. Node A also requests unpopular content e_1 initially stored at node E.

Using the MPC strategy in Figure 5.1(b), when node A sends an *Interest* message for content e_1, e_1's popularity is increased in the *Popularity Tables* of every node along the path [A, C, D, E], i.e., e_1's popularity is set to 1 at nodes A, C, D and E. Similarly, when B sends an *Interest* message for content d_1, the popularity of d_1 is set to 1 along the path [B, C, D]. Since content d_1 is a popular one, A and C send in turn an *Interest* Message for d_1. The *Interest* message from A will increase d_1's popularity to 2 at C and D (through the path [C, D]). Finally, C requests d_1, increasing d_1's popularity to 3 at node C and D. At this moment, D is the only node to hold the content d_1. Assuming that d_1's popularity has reached the *Popularity Threshold*, D spreads *Suggestion* messages to his neighbors C and E. C and E will therefore cache the content and will be able to reply to upcoming requests and transmit the content d_1. For instance, if there are interested nodes close to A, their requests will be redirected towards C and not D, reducing the



(a) Initial state of the network and the scenario; (b) MPC final state; (c) LCE final state.

Figure 5.1: MPC Workflow example.

number of hops to get the content. After d_1 is spread to its neighbors, D will reset d_1's popularity at the *Reset Value* in order to prevent continuously flooding the same content to neighbors.

Figure 5.1(c) describes the CCN final state after using the default *LCE* caching strategy. When the first request is sent by A, content e_1 will be cached at every nodes along the path [A, C, D, E]. Then, according to our scenario, content d_1 will be cached along the path [B, C, D] after the first request from B. The request from A will be directed only through C and d_1 will be cached into A. Finally, node C stores already the content since it has been cached from previous requests.

To summarize this scenario, LCE counts three replicas of e_1 at nodes A, C and D and three of d_1 at nodes A, B and C (Figure 5.1(c)). MPC counts only two replicas of d_1 at nodes C and E, and e_1 has never been replicated (Fig. 5.1(b)). Clearly, by caching only popular content, our proposed strategy MPC is saving resources such as memory, bandwidth and the number of caching operations compared with the LCE strategy. The duplication of d_1 at node E has been opportunistic since E never requests for it (Figure 5.1(b)). However, node E may answer upcoming requests for d_1 since it is a popular content.

5.2 Experimental results and analyses

5.2.1 Simulation Environment

In order to evaluate our new strategy, we use ccnSim [62], a chunk-level CCN simulator, developed in C++ over the Omnet++ framework. The simulator was first designed to evaluate the CCN performance with different strategies and scenarios. To promote cross comparison between MPC and CCN, we detail the simulation environment, including ccnSim inherent parameters and network topologies.

The ccnSim configurable parameters are depicted in Table 5.1. For all our simulations, CCN nodes use the *Always* caching strategy along with the *LRU* caching replacement policy – the LCE caching strategy is called *Always* in ccnSim. From now and so on, we refer to this cache configuration as LCE in order to be consistent with the rest of the thesis. We choose these policies because they have shown the best performance for CCN [62] at the fall of 2012. Other parameters are presented in the table such as routing policy, chunk size or requests rate.

In ccnSim, the popularity of files has been modeled following a MZipf distribution function [59, 62]. In our experiments, we choose the same distribution function to model the file

Parameter	Value	Parameter	Value
Run/Simulation	10	Request Rate	50 req/s
Warmup Phase	True	Chunk Size	10 kb
Multipath	Disabled	Caching Replacement	LRU
Routing Strategy	Closest	Caching Strategy	Always

Table 5.1: ccnSim parameters setting

	Segment	N	δ	σ_γ	$\Delta[ms]$	D
Abilene	Core	11	2.54	0.19	11.13	8
Tiger	Metro	22	3.60	0.17	0.11	5
Geant	Aggr	22	3.40	0.41	2.59	4
DTelekom	Core	68	10.38	1.28	17.21	3
Level3	Core	46	11.65	0.86	8.88	4
Tree	Ref.	15	2.54	0.21	1.00	3

Table 5.2: Properties of the network topologies [62].

requests. For a Youtube-like catalog of 10^8 files, it means that 99% of the requests are concentrated approximately into the 6,000 most popular files.

As several network topologies are included into ccnSim, we use those topologies which are described in Table 5.2. All of them correspond to ISP level topologies and its number of nodes ranges from 11 to 68 nodes.

All along the experimentation section, we use two scenarios defined in Table 5.3. The first scenario S_{small} is the scenario commonly used for CCN evaluation [88] and it consists of a catalog of 10^4 files with 10^2 chunks in average per file; cache size is fixed at 10^3 chunks per node. The second one $S_{youtube}$ is a larger-scale scenario with a Youtube-like catalog containing 10^8 files with 10^3 chunks per file: approximately a catalog of $1PB$. The ratio of caches over total chunks has been set to $\frac{C}{|F|F} = 10^{-5}$; in this scenario, cache size is then set to 10^6 chunks ($10\ GB$).

Then, for each experiment, we randomly set one catalog and 8 requester nodes on the topologies. We performed 10 runs per simulation and provide the average value.

5.2.2 MPC parameters tuning

The MPC caching strategy and its results depend on several parameters such as *simulation time*, *Popularity Threshold*, *Reset Value* and *Popularity Table size*. The *simulation time* determines the time represented in our simulations. The *Popularity Threshold* set the trigger value to start suggesting neighbors to cache content. *Reset Value*, the value to re-establish the popularity

Parameter	Description	S_{small}	$S_{Youtube}$
F	avg. chunks per file	10^2	10^3
$\|F\|$	number of files	10^4	10^8
C	number of chunks stored per cache	10^3	10^6
α	MZipf exponent parameter	1.5	1.5
q	MZipf plateau parameter	0	0

Table 5.3: Simulation experiments scenarios

after the suggestion process. Finally, the *Popularity Table size* set the number of entries which popularity can be stored. The *simulation time* and *Popularity Threshold* are first determined and then, we continue to tune the other parameters.

The *simulation time* is an important parameter for simulating accurately the different strategies. The ccnSim simulator warms up placing chunks in the caches according to a statistical simulation of the strategy. As *Popularity Tables* are not warmed up, MPC needs also time to start caching popular content. We therefore evaluate the time MPC needs to outperform LCE. We varied simulation time up to 40,000 Simulation Time Units (STU). LCE and MPC *Cache Hit* results are presented on Figure 5.2.

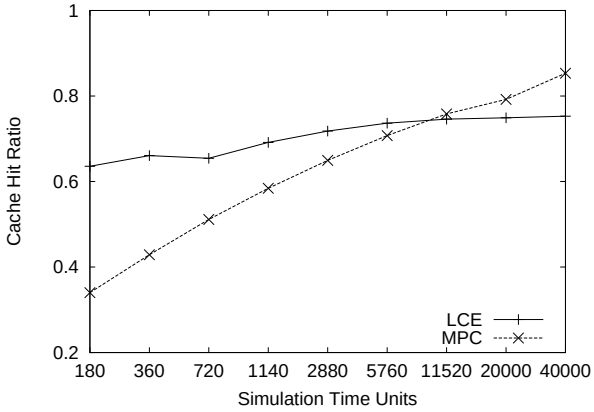


Figure 5.2: MPC parameters tuning: Simulation Time

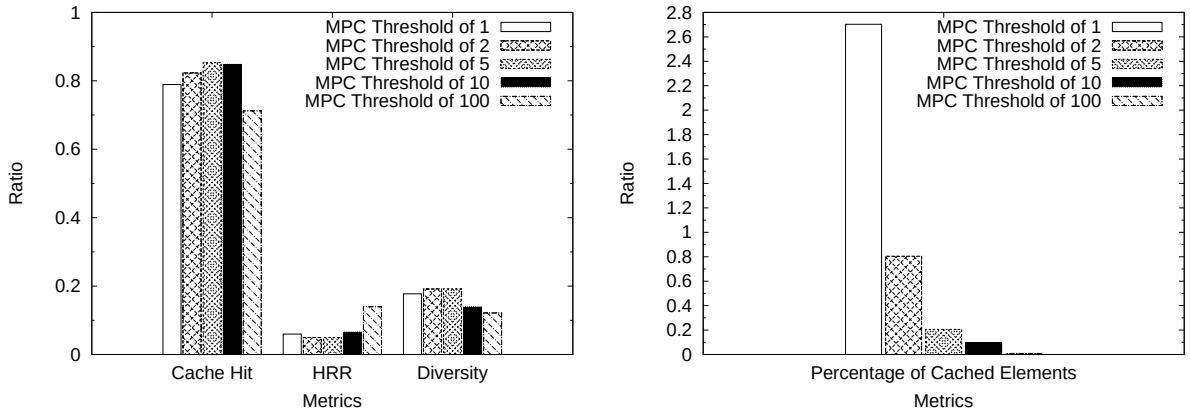
Clearly, for short time simulations and until 11,520 STU, LCE reached higher Cache Hit Ratio than MPC. Then the LCE's Cache Hit ratio remains stable while MPC's one is increasing and outperforms LCE. For instance, at 40,000 STU, Cache Hit Ratio is 85% for MPC and only 75% for LCE. In the rest of this Chapter, we focus on long time simulations and set simulation time to 40,000 Simulation Time Units.

Regarding the *Popularity Threshold* parameter, Figure 5.3a and Figure 5.3b show results with different threshold values. Clearly, *Popularity Threshold* set to 5 shows the highest Cache Hit, Diversity and lower Stretch than other tested values (Figure 5.3a). At the same time, the ratio of cached elements is kept very low at only 20% (Figure 5.3b). A lower value for this threshold (e.g. 1) may cache up to 2.7 times more than with the LCE default strategy.

The selection of MPC's parameters has been done according to an extensive simulation process over the $S_{youtube}$ scenario and the Tree topology.

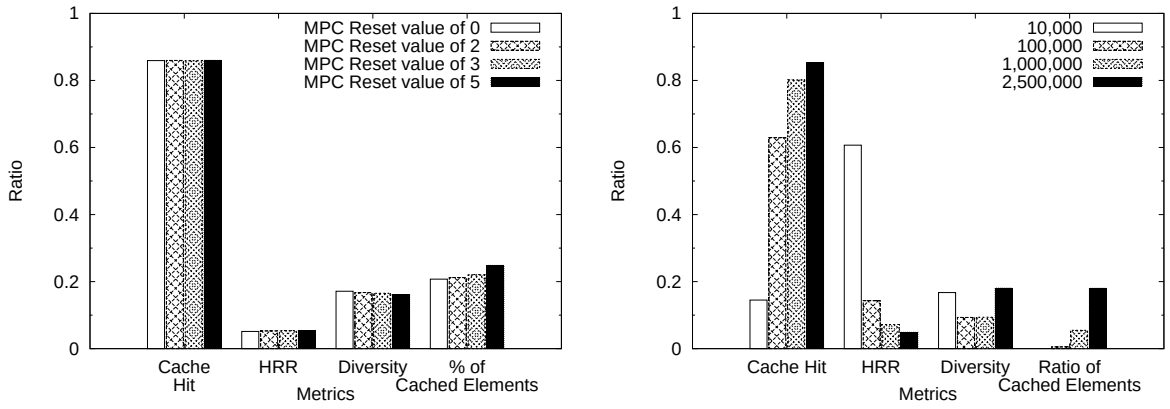
As explained before, when a content name reaches the *Popularity Threshold* and is spread to its neighbors, we need to reset this value in case of future requests for this content. As shown in Figure 5.4a, even though the cache hit is at the same rate for all the tested values, a *Reset Value* set to 0 is more appropriate. A *Reset Value* of 0 exhibits slightly higher diversity, less Stretch while reduces replications of the same content in the caches.

Since MPC uses a table to keep track of the content name's popularity, we need to choose an appropriate table size, without wasting the memory of cache nodes. Figure 5.4b shows that *Popularity Table* with 2,5 millions of entries has the highest Cache Hit, Diversity and keep the Stretch at a low value. By assuming that a content name is 1KB, it means 2.5GB of memory. Then, the cache-size space is shared with MPC requirements (*Popularity Table*), which stands for 25% of caching resources (2.5 GB in case of the $S_{Youtube}$ scenario of the total 10 GB for caches).



(a) (b)

Figure 5.3: Tuning of Popularity Threshold (5.3a and 5.3b).



(a) Reset value (b) Popularity Table Size

Figure 5.4: MPC Parameters Tunning

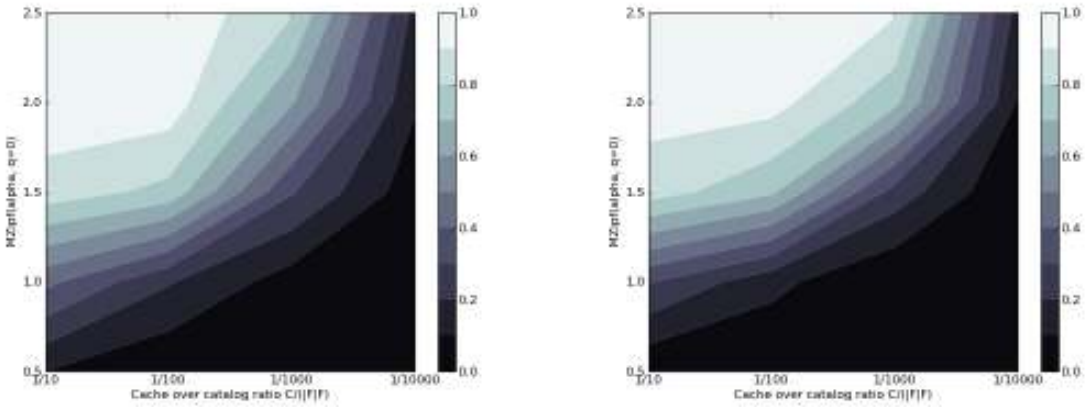
To sum up this evaluation part, we tune our MPC parameters for the upcoming experiments: simulation time is set to 40,000, *Popularity Threshold, Reset Value* and *Popularity Table size* are set to 5, 0, and 2.5 GB respectively.

5.2.3 MPC vs. LCE

In order to compare fairly MPC and LCE, we first performed the same experiment with the S_{small} scenario as in [62].

It consists in varying the ratio of the cache over the total number of chunks with different popularity distribution function (varying the MZipf exponent from 0.5 to 2.5) and observes the Cache Hit ratio. The results are shown on Fig. 5.5a and Figure 5.5b for LCE and MPC respectively. The similarity in the charts shows how likewise they behave for every configuration.

However, we compare the *Ratio of Cached Elements* for LCE and MPC. In LCE, content is cached by all nodes it passes by: the ratio for LCE would always be 1. It means one element cached per every node that a data message has passed. In MPC, the *Ratio of Cached Elements* is highly smaller than LCE as we show in the Table 5.4. For example, with an α value of MZipf



(a) LCE. (b) MPC.

Figure 5.5: Evaluation of MPC with a countour plot in a small scenario.

of 1.5 and ratio of cache $\frac{1}{100}$, MPC caches only 9% of content in comparison to the 100% cached by LCE. The ϵ value refers to values that tend to zero.

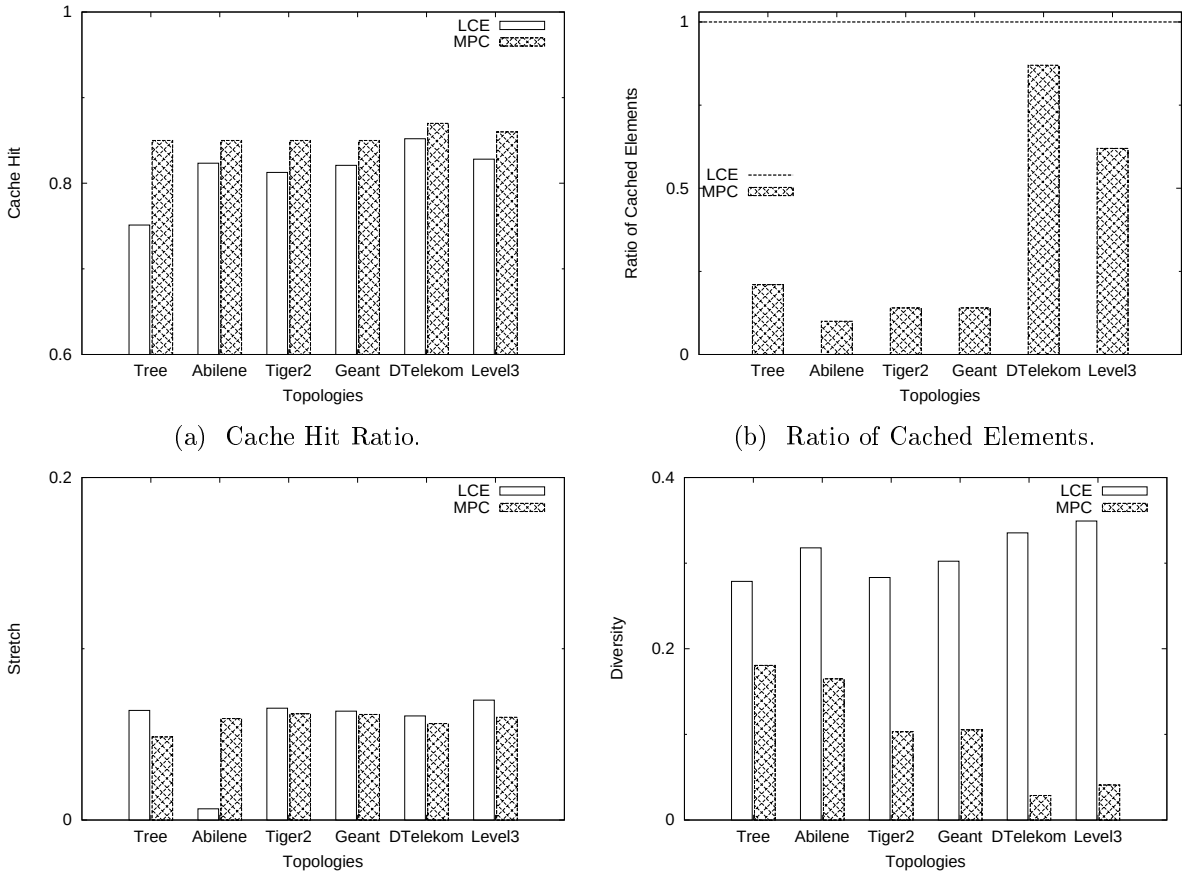
	Ratio of Cache			
MZipf α exponent	$\frac{1}{10}$	$\frac{1}{100}$	$\frac{1}{1000}$	$\frac{1}{10000}$
0.5	ϵ	ϵ	ϵ	ϵ
1	ϵ	ϵ	ϵ	ϵ
1.5	0.22	0.09	ϵ	ϵ
2	0.33	0.31	0.04	ϵ
2.5	0.37	0.38	0.15	ϵ

Table 5.4: Ratio of cached elements for MPC. This ratio is 1 for the LCE strategy. (ϵ is for value close to 0.)

We now compare MPC to LCE by using the $S_{Youtube}$ scenario with all the topologies. In Figure 5.6a, clearly MPC Cache Hit ratio is higher than LCE and above 85%. Even when LCE reaches its highest results with Level3 or DTelecom topologies, MPC still outperforms LCE. The *Ratio of Cached Elements* is presented in the Figure 5.6b. For CCN with the *LCE* strategy, content is always cached and it reaches a ratio of 100%. The MPC strategy caches much less content than LCE for the Tree, Abilene, Geant and Tiger topologies (approximately 20%), performing less caching operations and saving memory. MPC caches more content with DTelecom and Level3 topologies (80% and 60% respectively) but is still at lower rate than LCE. The increase of cached elements for these two topologies is due to the high connection degree of nodes (10 or 11 neighbors in Table 5.2). In such highly connected topology, MPC is caching *off-the-path* because it sends *Suggestion* messages to store content to more neighbors.

Figure 5.6c presents the Stretch metric. MPC and LCE exhibit similar results: the Stretch of LCE is about 10% for all the topologies and the MPC's one is slightly lower at 8%. By caching only popular content, MPC strategy is still able to cache content close to requesters.

The Diversity metric is presented into the Figure 5.6d. The LCE Diversity ranges from 28% to 35% for all the topologies and the MPC Diversity is much lower from 3% to 18%. Regarding Diversity, it was expected that MPC is less efficient than LCE since MPC has been designed in



(a) Cache Hit Ratio.

(b) Ratio of Cached Elements.

(c) Stretch.

(d) Diversity.

Figure 5.6: Evaluation of MPC with a countour plot in a Youtube-like scenario.

order to cache only popular content, limiting the diversity of the chunks in the cache of nodes. However, neither LCE nor MPC achieves high Diversity and further work is needed to improve Diversity in CCN networks.

Finally, our simulation experiments show that MPC strategy outperforms LCE strategy since it achieves a higher Cache Hit Ratio. Moreover, MPC caches less content, saves resources such as memory and reduces the number of cache operations.

5.3 Discussion

All along this chapter, we have presented and assessed the performance of the Most Popular Caching (MPC) strategy. This strategy has shown its ability to work in the proposed environment and it shows better results than the common used LCE caching strategy. In this section, we discuss our results and set some limitations of this work.

MPC improves Cache Hit and it reduces the number of eviction operations performed all across the network. It is important to note that the parameters have been tuned for a particular environment. MPC improves LCE in a environment with a popularity model of $\alpha = 1.1$.

MPC has shown similar results in several topologies in terms of Cache hit and Stretch. However, results are quite different once we analyze Ratio of Cached Elements and Diversity.

We assume that MPC is a topology dependent caching strategy and we expect to study in the future which are the best topologies for MPC. At the moment, there are insights that MPC works better with low connection degree. We expect to study different suggestion mechanisms in the future.

MPC is more complex to implement than LCE. Using LCE, we only need to run through a double-linked list to know if the content is in the cache. While using MPC, we need to go through two lists (i.e. the cache and the popularity table). However, we may implement popularity tables using mechanisms with complexity of $\mathcal{O}(1)$ for running through a table as it has being done in noSQL databases such as REDIS [123].

In the argumentation of the size required for the MPC strategy, we talked about the CCN names demands $1023B$. Some reviewers have argumented that names may be hashes. Smaller names implies that the space required for MPC is smaller and we have considered the worst case in our experiments.

5.4 Summary

In this chapter, we have presented the Most Popular Caching (MPC) Strategy. MPC is a caching strategy based on popularity of content. As principle, MPC only caches popular content. By purging caches of unpopular content, it saves space for other pieces of content and avoid to waste resources with barely demanded pieces of content.

We have replicated simulation environments proposed with ccnSim [62]. Rossi et al. use two simulation environments: a small catalog of 10^4 files and another with the size of a Youtube catalog (10^8 files). Once the simulation environment was fixed, We tuned empirically the internal parameters of MPC.

Then, we resort to compare MPC with the LCE caching strategy. We compared both strategies in two environments. A small catalog of 10^4 files and another of the size of the catalog of Youtube. Our simulation experiments showed that MPC outperforms the default-CCN LCE strategy. MPC achieves a higher Cache Hit and still reduces drastically the number of replicas of elements. By caching less data and improving the Cache Hit, MPC improves network resources consumption.

As future work, we expect to adapt MPC automatically to simulation environment. All along this work, we have tuned empirically its internal parameter. In addition, we believe that MPC access count information may be used to extract information and profile communities in the network. Thus, we may perform analysis of consumed content in the network and eventually adapt our mechanism to the new environment.

In addition, we expect to deploy MPC in a real testbed. In the following chapter, we are going to present experiments in the PlanetLab platform to evaluate the Socially-Aware Caching Strategy. As the environment is fully functional, we expect to deploy MPC in this testbed and to offer an implementation of MPC for the CCNx prototype.

6

Socially-Aware Caching Strategy for CCN

Contents

6.1 Introduction 79
6.2 Social Network Model 80
6.2.1 Model 81
6.2.2 SONETOR: Social Network Traffic Generator 82
6.3 SACS: Socially-Aware Caching Strategy 83
6.4 Experimental results and analyses 85
6.4.1 Simulation Environment 85
6.4.2 Experiments on the Inet Topology 86
6.4.3 Experiments on the common evaluation scenario 88
6.4.4 Experiments on PlanetLab 89
6.5 Discussion 91
6.6 Summary 92

6.1 Introduction

Over the past few years, Online Social Networks (OSN) have gained tremendous popularity on the Internet. Millions of users interact with each other through OSN such as Facebook or Twitter. Facebook announced one billion of users [126] while Twitter is massively used as a micro-blogging service. People create social relationships through OSN and exchange information about what they experience in their life within their community. As an example, the re-election picture of President Obama has become the single most-retweeted message in Twitter history with more than 800,000 re-tweets [128]. New ubiquitous devices (smartphones, tablets) appeared and include functionalities to instantaneously share information through OSN. Most if not all the Internet services improve the users' experience through the addition of social features to rapidly spread interesting content. Companies invest strongly into their Facebook pages to promote new products and benefit from user's feedback [75]. 90% of American hospitals use social media to attract new customers and one third has a formal social media plan [127]. Users follow friends and families and organize social events through the Internet. The Internet is becoming a social-oriented network.

In Chapter 4, we have compared the caching strategies proposed in the literature. In Chapter 5, we have built a caching strategy based on popularity of content. The caching strategies studies at the moment recur to popularity of elements to take caching decisions and others considers topological properties. Caching strategies have never been based on social networks information.

In this chapter, we propose to include social information in the design of a new caching strategy for Content Centric Networking. We conjecture a small number of users counts a huge amount of social relationships, dominates the activity and receives most attention from other users [74]. We call such users *Influential users*, and we argue that they produce content that is more likely to be consumed by others, and in consequence their content must be favored and replicated in priority. Our novel caching strategy is therefore prioritizing content from Influential users of the social network.

The major contributions of this chapter are :

- A model of social network that represents behavior of users in a social network environment. Based on this model, we present a tool to generate synthetic social traffic: SONETOR, the Social Network Traffic Generator. SONETOR is used all along the experimentation section in this chapter;
- Our social-network based caching strategy for Content Centric Networking;
- Extensive simulations of the interaction of users into two social environments with thousands of users for evaluation of the caching strategy;
- An implementation of our caching strategy in CCNx and a deployment on PlanetLab. The results shows that our proposed socially-aware strategy improves drastically the caching performances of CCN.

This chapter is organized as follows. Section 6.2 describes our social network model and SONETOR. Section 6.3 details our novel socially-aware caching strategy designed for CCN. Section 6.4 details the simulation environment, and emphasize strongly on the simulation parameters, the users' social interactions and the simulation tool. We also present the simulation experiments results and the benefits of our caching strategy for CCN. Section 6.4.4 details the implementation of our caching strategy into CCNx and the experiments on PlanetLab. We then sum up our findings and discuss our model in Section 6.5. Section 6.6 summarizes the contributions of this chapter.

6.2 Social Network Model

As the Internet is becoming a social oriented network, it is essential to model users behavior in a social-network fashion. This users behavior is going to be used all along this chapter.

Our model represents a set of users, which are connected among them according to social acquaintances. These users performs activities. The model is summarized in Figure 6.1. In the Figure, we have 6 users (A, B, C, D, E and F), which are connected to their friends (with red links). These users perform activities which are shown in the right side of the figure. In this section, we detail in-depth the model of social network and we propose an open-source tool, which implements the model.

In the rest of the section, we describe in-detail the social network model by analyzing: (i) a social network model capturing the social relationships between users, (ii) a users' interaction

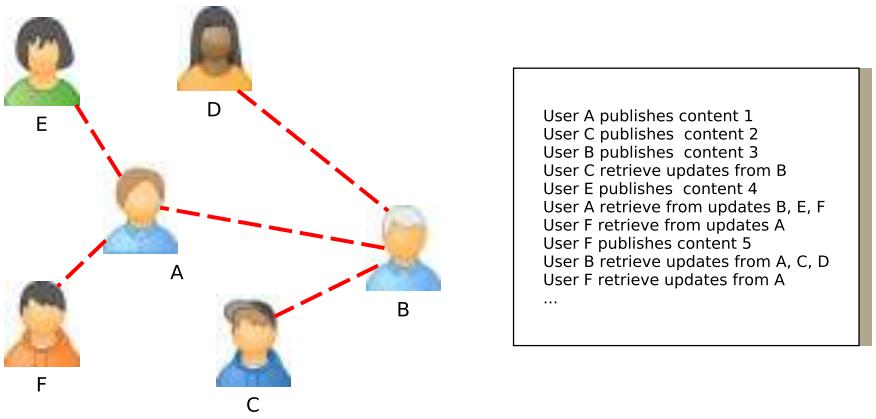


Figure 6.1: Social Network Model.

model capturing the activity schedule for each user, and (iii) a model for each type of activity and its dependent parameters. In order to emulate the users' activities in the social network, we present the users' interaction model. Our interaction model is based on research studies on social networks [72, 73].

6.2.1 Model

Activities

Online Social Networks (OSN) allow users to publish content at their own will and share it with their acquaintances (i.e., *friends*). Friends may always be updated through a news feed system. Each time a user finds interesting content (text, pictures or video), he may share it with his friends, spreading and expanding the visible scope of these information. Thus, we model a *social network* by a network where users can publish, retrieve and share information with their communities, according to their personal preferences. In our social model, each user has therefore two functionalities to interact with its community:

- *Publish*: the production of new content. After retrieving a content, users may share it again with their friends (i.e., *re-tweet* a message).
- *Retrieve*: this function allows users to receive the last content issued by all their friends. For example in Figure 6.1, the user A has friendship relationships (red dashed-line) with users B, E and F. Each time A issues a Retrieve message, A will obtain the content from all its friends B, E and F.

Social Relationships

As we model a social network, we have to represent the connections between users. They can be described by a social network graph. This social network graph may represent friend relationships as well as professional affiliation depending on the social network (e.g.: Facebook, Linkedin, etc).

Users' Interaction

Our interaction model is based on several major studies on social networks [72, 73]. From these studies we extract four parameters to model the users' interaction: the number of sessions NS,

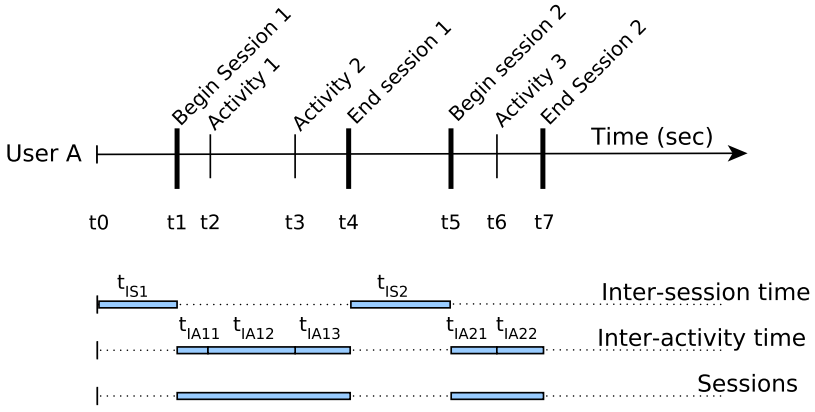


Figure 6.2: Users' interaction model. Users can perform several activities within sessions.

the inter-session time t_{IS_i}, the number of activities per session NA_i and the inter-activity time t_{IAij}.

The users' interaction model is depicted in Figure 6.2 and can be summarized as follows. A user may start NS independent and consecutive sessions. The interval time between each session is calculated with an inter-session time t_{IS_i} from the time origin $t0$ or the previous session ending time. In every session i, the user performs a finite number of activities, NA_i, such as publications or retrievals from friends. These activities are separated by an inter-activity time $t_{IA_{ij}}$. In the Figure 6.2, we illustrate an example of interactions in-which User A has two sessions ($NS = 2$); the first session starts after the inter-session time t_{IS_1} and contains two activities ($NA_1 = 2$), which are separated by three different inter-activity times $t_{IA_{11}}$, $t_{IA_{12}}$ and $t_{IA_{13}}$. The second session counts only one activity ($NA_2 = 1$) and is separated from the first session by a second inter-session time t_{IS_2}.

6.2.2 SONETOR: Social Network Traffic Generator

We aim at representing social network traffic, to be able to use it in our simulations. This social network traffic follows the previously presented model.

To this end we propose *SONETOR*, a SOcial NEtwork Traffic generatOR. It aims at offering an open-source tool to generate network traffic workload with OSN characteristics. SONETOR is able to generate the users' interaction model proposed in this chapter.

In the Figure 6.3, we represent the workflow of SONETOR. Following the presented model, SONETOR receives each parameter as an input organized into a configuration file. These parameters includes the social relationships graph and the probability laws that represent all the other parameters of the model such as NS, t_{IS_i}, NA_im t_{IAij} and other optional parameters such as popularity of content or mobility. As output, SONETOR generates a synthetic trace file, which is depicted in the figure.

A sample of SONETOR synthetic trace is presented in Figure 6.4. Each line is ordered by a timestamp and describes an activity performed by a user. Each activity can count several dependent parameters. It is noteworthy to mention that there are other options (opt_i) for further

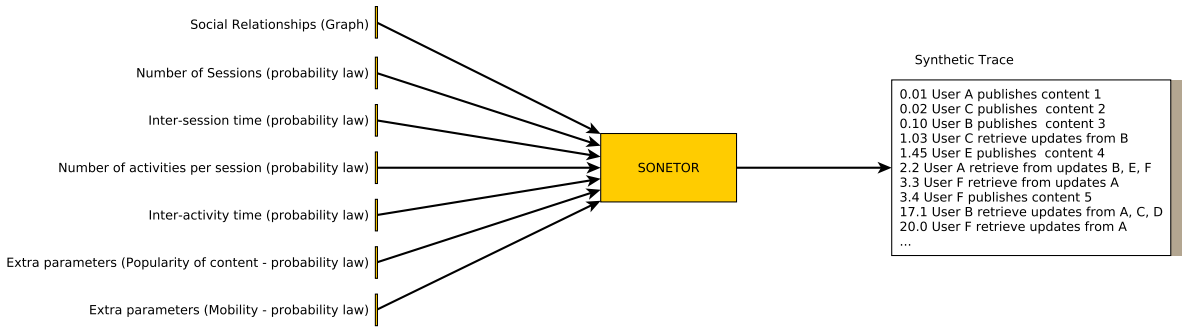


Figure 6.3: SONETOR Workflow.

extension of our generator tool. Such extensions can take into account mobility of users, or their geographical coordinates for more sophisticated scenarios.

```
Timestamp 1    User a    Activity(d 1, ..., d m)    [opt 1, opt 2...]
Timestamp 2    User b    Activity(d 1, ..., d m)    [opt 1, opt 2...]
Timestamp 3    User c    Activity(d 1, ..., d m)    [opt 1, opt 2...]
...
```

Figure 6.4: Format of Sonetor traces

SONETOR includes several graph models such as *small-world graph* or *random graph*. In addition, SONETOR also includes realistic social network graphs, extracted from social relationships from dataset publicly available of Facebook [84] and LastFM [83].

The tunning of the SONETOR model (NS, t_{IS_i}, NA_i and t_{IAij}), with realistic values extracted from research studies, and a use case showing the full potential of SONETOR are presented in [85].

SONETOR is already publicly available [131].

6.3 SACS: Socially-Aware Caching Strategy

We proposed a novel caching strategy for CCN based on the social context of users, which refers to information gained form social relationships. Our socially-aware caching strategy gives priority to content issued by Influential users and cache it pro-actively into the CCN network. Indeed, users with more social relationships (*popular users*) are more influential than users with fewer relationships, and they produce content that is more likely to be consumed by others. For instance, when a popular Twitter user sends a message, it may count much more re-tweets than a message from a regular user. In the rest of this Chapter, we refer to our socially-aware caching strategy as *SACS*.

As an hypothesis for our strategy, we assume that the users follow a social network model such as the one previously presented. These users are placed into a CCN topology. It means that the social networks users are mapped into CCN nodes and their activities, transformed into

requests, are performed from the CCN node. The Figure 6.5 represents the model assumed by our caching strategy. In the example, we have 6 different users placed into 9 CCN nodes. The black solid-lines represent the linkes between CCN nodes and the red dashed-lines represent the users' social relationships.

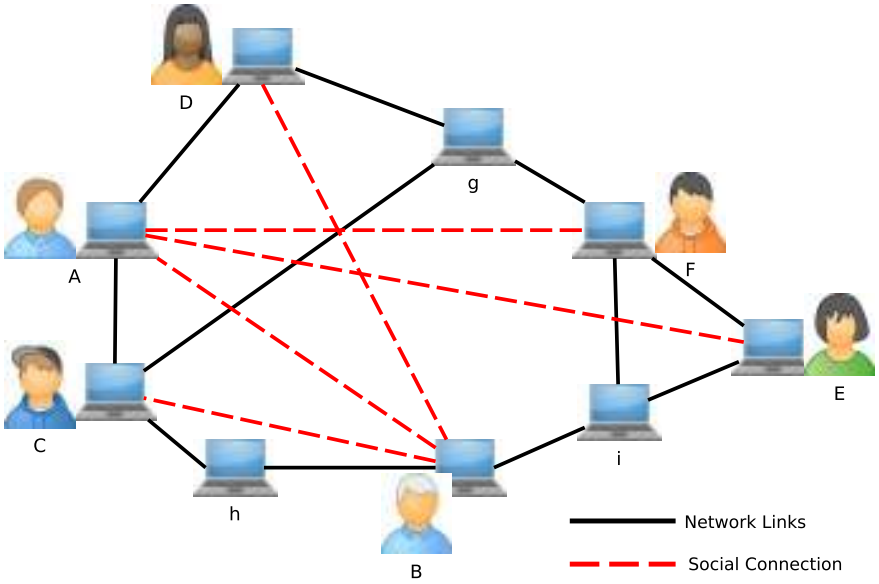


Figure 6.5: Example of a social network on top of a CCN architecture.

In our novel strategy, the content published by Influential users is pro-actively replicated into the shortest path towards their social neighbors before it is requested by them. It improves the availability of Influential users' content and reduces the number of *Interest* messages in the CCN network. Thus, *SACS* privileges content issued by Influential users. It is therefore an important matter for *SACS* to detect the influential users in the social network.

Influential Users Detection

We detect the influence of users within a social network by using the Eigenvector and PageRank centrality measures. These measures allow the computing of a score for every user in the network according to their *importance*. More details on the computation of these centrality measures can be found in [76] and [77] for Eigenvector and PageRank respectively. For instance, we list into the Table 6.1 the score of users from the previous example (Figure 6.5) by using both the Eigenvector and PageRank centrality measure. We then define a user as being *Influential* if its score is greater than the average score of the overall social network. In case every user has the same number of friends, our influential metric will fail. However as we represent social interactions, it is unlikely that every user in the network share the same number of friends.

From table 6.1, A and B are Influential users because their scores are greater than the average one ($0.58 \geq 0.38$ and $0.29 \geq 0.17$ respectively) while users C, D, E & F are non-influential users. Thus, whenever Influential user A publishes a new message, it is proactively cached into nodes along the shortest paths towards its social relationships B, E and F ($[A, C, h, B]$, $[A, D, g, F]$ and $[A, D, g, F, E]$). When F publishes a new message, it is not pro-actively cached as F is not an Influential user.

	Eigenvector		PageRank	
Users	Score	Influential	Score	Influential
A	0.58	Yes	0.29	Yes
B	0.58	Yes	0.29	Yes
C	0.29	No	0.11	No
D	0.29	No	0.11	No
E	0.29	No	0.11	No
F	0.29	No	0.11	No
Avg.	0.38	N/A	0.17	N/A

Table 6.1: Scores of the centrality measures computed from the users' social relationship on Figure 6.5.

6.4 Experimental results and analyses

6.4.1 Simulation Environment

We consider a future content-centric Internet built on the CCN architecture. Besides the caching capabilities at each node, the topological structure of this future Internet will be just like today's Internet and we believe ISP providers are going to adapt its structure to CCN. We believe that current Internet devices will be replaced with ccn-capable devices. However, the cable and wire infrastructure will remain the same. Thus, topologies are not likely to be replaced from those used in the current Internet. We consider two topological structures for our experiments. First, we resort to *Inet* [98], a common tool to generate Internet topologies that we use to model the CCN topology. We use the *Inet* default parameters as presented in Table 6.2 and the network topology we use throughout this Chapter counts 3,037 nodes. Second, we study ISP level topologies and impact of SACS in the common evaluation scenario presented in Chapter 4.

Social Network

In order to model the social relationships between users, we resort to two publicly available data sets : (i) LastFM data set [83] and (ii) Facebook data set [82]. LastFM is a music recommender system, where users share their musical preferences with their community. From the LastFM data set, we extract a social graph counting 1,896 users, 12,717 bi-directional relationships and each user counts on average 13 relationships. Facebook is the most popular OSN and the data set consists of 4,039 users, 88,234 friend relationships and each user counts in average 44 relationships. Our simulation experiments are therefore performed with two different social network models.

The users of the social network are randomly mapped into the network topology.

Users' Activities

Now that we model social networks from publicly available data set over a CCN topology through *Inet* and common evaluation scenario, we model the users' activities and interactions, i.e., the behavior of users and how they publish or retrieve pieces of information from their social relationships. Remember that in our model, users perform activities within sessions (Fig 6.2) and that they have two activity functions within the social network: *Publish* and *Retrieve* (Section 6.2.1). We therefore extract from previous studies [72, 73] the distribution parameters to model the

Social Network Activity Traces	
Number of sessions NS	Zipf ($\alpha = 1.792, \beta = 1.0$)
Number of activities NA_i	Zipf ($\alpha = 1.765, \beta = 4.888$)
Inter-session time t_{IS_i} (s)	LogNormal ($\mu = 2.245, \sigma = 1.133$)
Inter-activity time $t_{IA_{ij}}$ (s)	LogNormal ($\mu = 1.789, \sigma = 2.366$)

Caching Configuration	
Replacement Policies	LRU
Centrality Measure	PageRank
Cache Size	$\{10^{0..3}\}$

Social Network Topology	
LastFM	
#Users	1,896
#Links	12,717
Avg. Degree	13
Facebook	
#Users	4,039
#Links	88,234
Avg. Degree	44

Table 6.2: Simulation parameters of the experiments.

users' interactions such as the number of sessions, the number of activities by session, the inter-session time and the inter-activity time. From these measurement studies, we also found out that on average, 5% of the user's activities are publications. This is consistent with the fact that most of the users only consult their timelines in the OSN and they barely publish new content such as status updates or share multimedia content. These distributions and their parameters are presented in Table 6.2. Then, for each user from the two data sets (about 1,900 and 4,000 users respectively), we generate a sequence of activities (Publish and Retrieve) and sessions. These activities are timestamped and we obtain a realistic sequence of social activities for each user. By merging all the sequences of users' activities, we obtain synthetic social network activity traces for the LastFM and Facebook data sets. Following this procedure, we can generate as many traces as we need and for the next experiments we generate 20 different synthetic traces. On average, each trace counts 56,000 activities for the Facebook data set, and 25,547 activities for the LastFM data set.

6.4.2 Experiments on the Inet Topology

The Figure 6.6 presents results of *Inet* simulation environment. This figure consists in fact of 8 figures: the four figures on the left column (Fig. 6.6(a)-(b)-(c)-(d)) are the results obtained with the LastFM data set, while the four on the right column are results with Facebook data set (Fig. 6.6(e)-(f)-(g)-(h)). Four rows of two charts depict each of our metrics: Cache Hit (Fig. 6.6(a)-(e)), Stretch (Fig. 6.6(b)-(f)), Expired Elements (Fig. 6.6(c)-(g)) and Diversity (Fig. 6.6(d)-(h)). All the figures share the same axis: the x-axis is the cache size ranging from 1 to 20 elements; The y-axis is the probability. For clarity reasons, we show the x-axis only on the two bottom figures ((Fig. 6.6(d)-(h)), and the y-axis on the figures of the right column (Fig. 6.6(e)-(f)-(g)-(h)). For each simulation experiment, we performed 20 runs of each simulation using different social network activity traces and provide the average value and the confidence intervals.

Figures 6.6(a) and 6.6(e) show the Cache Hit performances of CCN with or without using *SACS*, our socially-aware caching strategy (for Eigenvector and PageRank measures). Without our strategy, the Cache Hit of LCE achieves low values and it barely reaches 5%. *SACS* in-

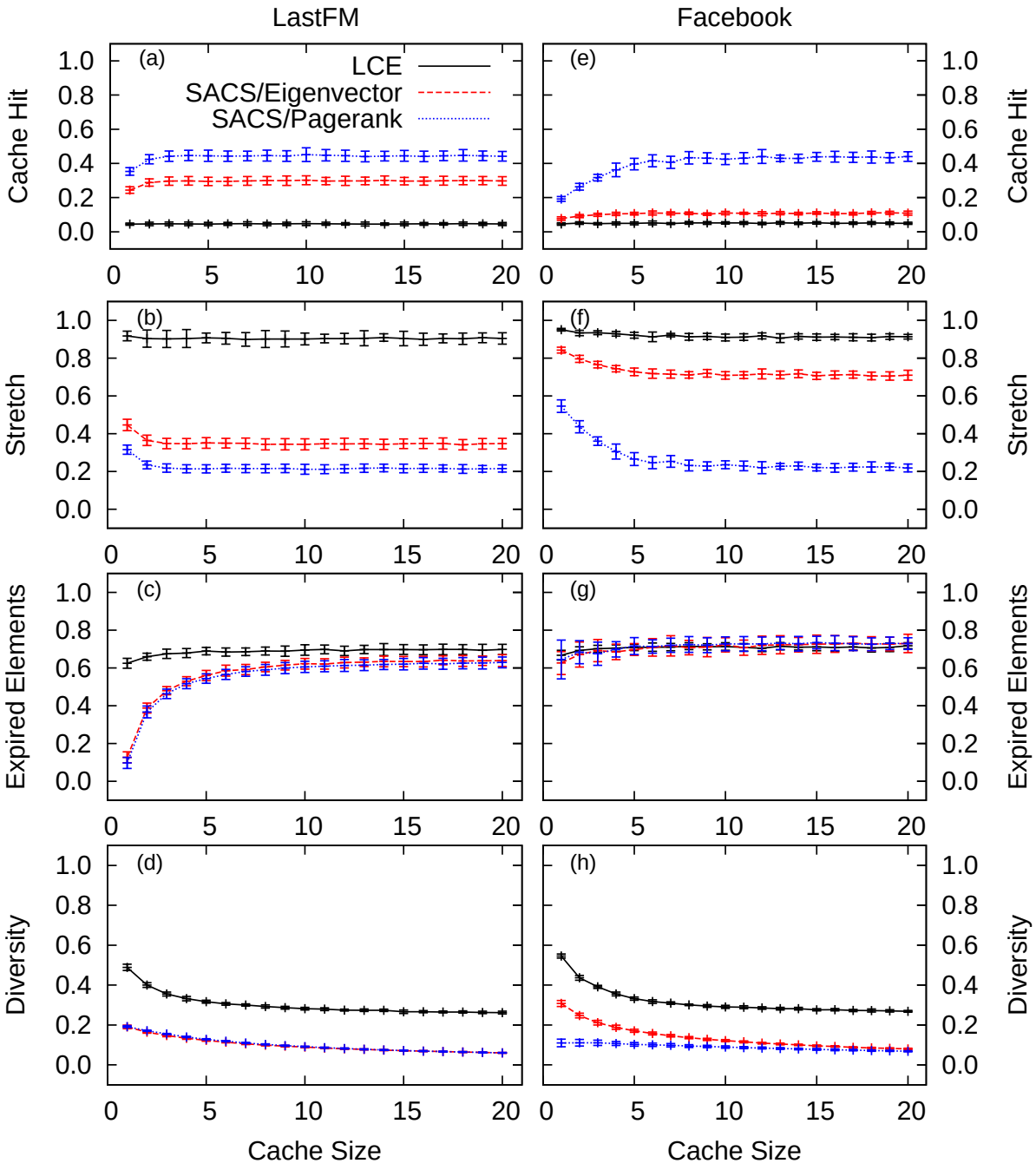


Figure 6.6: Evaluation of *SACS* in *Inet* Evaluation Scenario.

creases significantly the Cache Hit and reaches 30% for *SACS/Eigenvector* and up to 40% for *SACS/PageRank* with the LastFM data set, while it reaches 10% and up to 40% using Facebook data set.

The low performances for CCN are due to the use of large-scale and realistic social and network topologies. Indeed, the long routes traversed by the content affect the Cache Hit and its computation works as follows. In every hop passed by an *Interest* message, the Cache Hit gets updated with a *Hit* or a *Miss*. If the requested content is present at hop n , we obtain a

single *Hit* and there have been $n-1$ hops without the requested content (i.e., $n-1$ *Miss*). As the Cache Hit metric is the ratio of *Hit* with regard to the number of *Miss*, the longer the path to find the content, the lower the Cache Hit is. In our case, our strategy pro-actively caches content from Influential users in the network paths. The *SACS* strategy succeeds to make the content more available and improves drastically the caching performances of CCN.

Stretch is presented in Figures 6.6(b) and 6.6(f). This metric is in direct correlation with Cache Hit. Without our strategy, the Interest messages traverse 90% of the shortest path to get the content. As expected, the distance to get the content has greatly decreased with *SACS*: *Interest* messages traverse only 35% and 72% of the shortest path with *SACS/Eigenvector* and 22% and 26% of the path with *SACS/PageRank* (LastFM and Facebook data set respectively).

Figures 6.6(c) and 6.6(g) show the ratio of Expired Elements in the caches. This metric is important to verify if the content stored in the caches is still requested or not (i.e., outdated). In OSNs, users publish content into timelines. These timelines are ordered sequences of content. When users timelines of their friends, they receive last friend publications. The old publications are never shown, unless somebody explicitly request them. These pieces of content no longer requested are called stale or expired elements. In our experiments, one could have expected that LCE performs better than our socially-aware caching strategy because *SACS* favors the content from Influential users. However, as seen on Figures 6.6(c) and 6.6(g), *SACS* achieves the same level of performances as LCE. Indeed, our strategy gives priority to content from Influential users, and these popular users produce content that is more likely to be consumed by others. Thus, content cached by *SACS* is still mainly requested and it keeps the level of expired elements at the same level of LCE.

Last, the Diversity metric is presented in Figures 6.6(d) and 6.6(h). Since this metric stands for the number of distinct content stored across all the caches with respect to the total caching space, Diversity while the cache size grows. There is less diversity in the caches with *SACS* and this result was expected since our strategy discriminates content from Influential users. Hence *SACS* deliberately creates multiple copies of Influential's content, reducing the variety in the caches at the same time. However, even though *SACS* does not provide as high Diversity as LCE, it still drastically increases the caching performances in CCN.

It is noteworthy to mention that larger cache sizes have no impact on the Cache Hit. In Fig. 6.6(e), the Cache Hit for *SACS/PageRank* remains stable from a cache size of 6, while it is stable at any Cache size for *SACS/Eigenvector* or for the other data set (Fig. 6.6(a)).

Regarding both data sets, the number of users has no impact on the performances of our strategy. Indeed, Facebook data set counts almost twice the number of users as the LastFM one (4,039 and 1,896 respectively, Table 6.2), and our strategy achieves high levels of performances with each social environment. It is especially the case with the PageRank centrality measure (blue-dashed line on Fig. 6.6(a)-(e)). With the LastFM data set, *SACS* shows similar performances for both centrality measures. For the Facebook data set, *SACS/PageRank* reaches high performances while *SACS/Eigenvector* is lower and slightly improves LCE. The average degree can explain this trend (13 for LastFM and 44 for Facebook, Table 6.2) and the PageRank measure exhibits better performance than Eigenvector with a highly connected social environment. As the two centrality measures succeed to detect influential users in the network, the PageRank measure is a better choice for our strategy according to empirically tested experiments.

6.4.3 Experiments on the common evaluation scenario

In this section, we aim at homogenizing results with the rest of the thesis and to discover new insights in the common evaluation scenario. We expect to validate previous results and to

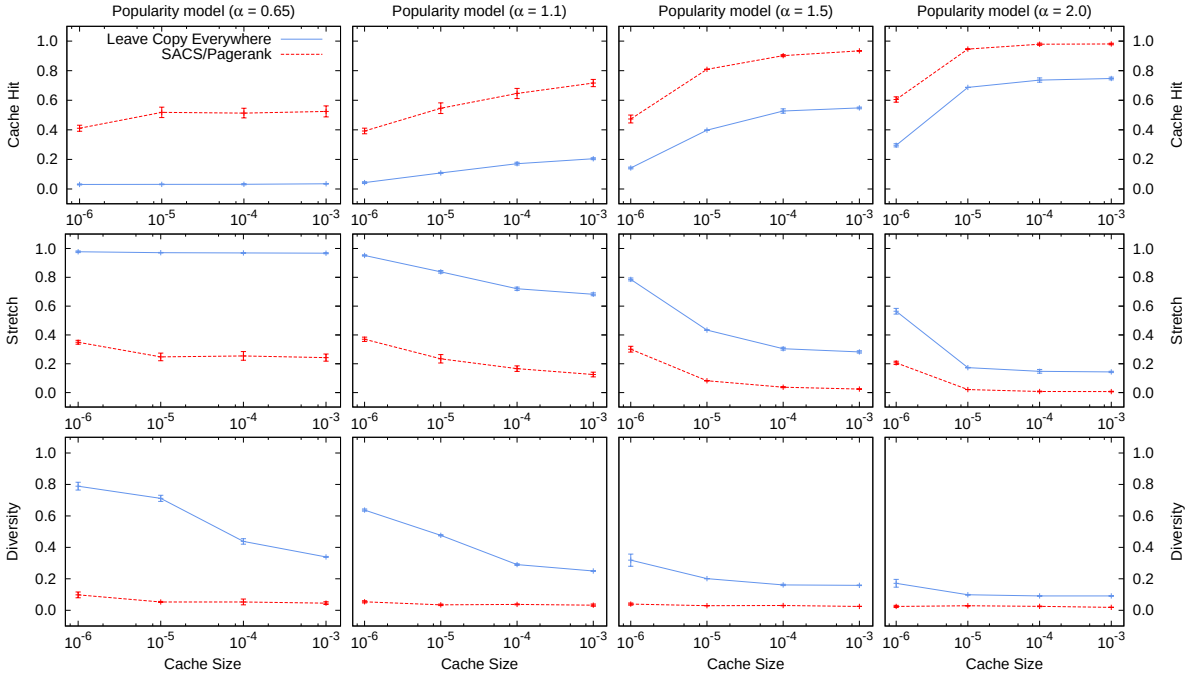


Figure 6.7: Evaluation of *SACS* in Common Evaluation Scenario.

experiment if SACS outperforms LCE in other popularity scenarios.

Figure 6.7 shows the comparison of the caching strategies in the common evaluation scenario. This figure is divided into 3 lines and 4 columns: each line stands for three metrics to analyze the caching strategies (Cache Hit, Stretch and Diversity); each column stands for a different content popularity (Zipf α parameter). In each plot, x-axis is the cache size and y-axis is the metric. For each value, each experiment has been performed three times in distinct topologies (Table 4.3) and we show the confidence intervals.

In a glimpse over the Figure 6.7, we observe that, in terms of Cache Hit and Stretch, *SACS* always overcomes LCE: it keeps a higher Cache Hit while it reduces the Stretch. It means that caches of the CCN network are going to answer more requests and to reduce the distance traveled to get the content.

The weak point of *SACS* is its low diversity. *SACS* diversity barely reaches 10% with a popularity scenario with an α of 0.65. In all the diverse popularity scenarios, *SACS* diversity tends to the minimum value, $\frac{1}{N}$ (i.e., see definition of the metric in Section 3.4). As we consider Diversity is an important aspect, we expect to develop this issue in the near future.

With regards to topologies, *SACS* performance barely changes in the distinct popularity scenarios. We remark this fact due to the small standard deviation presented in the results. As standard deviation tends to be smaller than 0.03 between ISP level topologies, *SACS* is expected to show the same performances with different topologies.

6.4.4 Experiments on PlanetLab

Besides simulation experiments, we evaluate *SACS*, our socially-aware caching strategy, on a real testbed. To this end, we use CCNx [113], the most advanced prototype of CCN, and we perform experiments on PlanetLab, a planetary-scale testbed platform that supports the development of new network services [121].

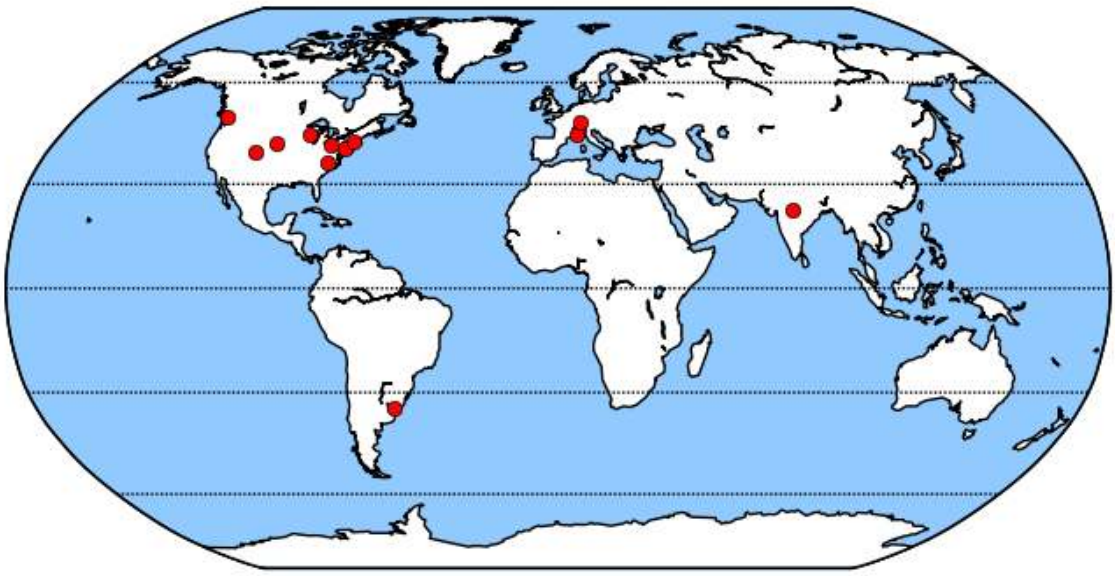


Figure 6.8: Location of PlanetLab nodes.

We implemented our socially-aware caching strategy into CCNx v0.7.1. and the PageRank centrality measure to detect Influential users as it has shown the best performances in the simulation experiments. Our modified CCNx prototype has been deployed into 14 PlanetLab nodes geographically distributed. During our experiments, as PlanetLab is a global research network, we could not handle more stable nodes at the same time. The locations of the PlanetLab nodes are indicated in the Figure 6.8.

For these experiments, we used the LastFM data set presented in Section 6.4.1 that counts 1,896 users and for which we obtained synthetic social network activity traces to model the users' activities. Each of the 1,896 users was assigned to one of the 14 PlanetLab nodes. The built CCN topology consists of a fully connected topology between the 14 PlanetLab nodes. We perform three runs of the experiment for each cache size with different synthetic traces. We evaluate the Cache Hit performances of CCN with regards to our strategy. The results of the experiments

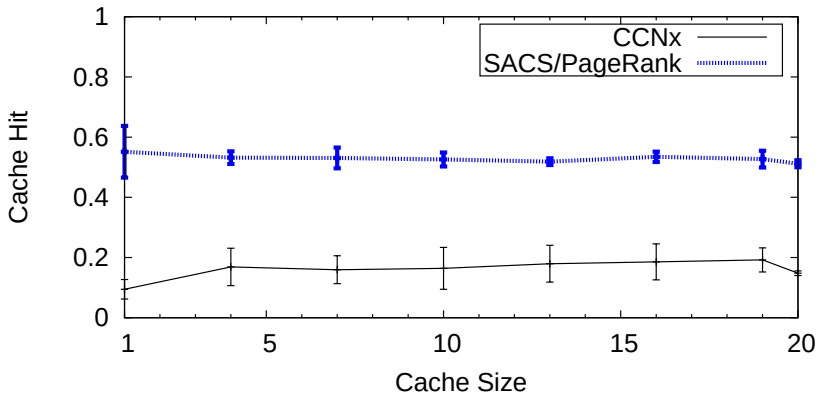


Figure 6.9: Evaluation of SACS in PlanetLab.

are presented in Figure 6.9 where we provide the average value and the standard deviation.

In the previous section, we used additionally Stretch, Diversity and Expired Elements. Due to the high complexity to implement these metrics in the PlanetLab platform, we decide to only implement Cache Hit and to use it to assess the previously shown results. To implement Stretch, we require to control every node by-which the messages passes to be able to count every hop. CCNx is implemented over IP, messages from node A to node B travel through IP. CCNx acts as a proxy server: after sending the message into node A, the message is transformed into IP; the IP message travels the IP network until it arrives to node B where it is reconverted into a CCN message. Even when the message is sent over CCN nodes, the message travels through IP. As we do not have control over the intermediary IP nodes, we can not calculate Stretch. For Expired Elements, we require to synchronize the clocks and Diversity experiences the same problem than Stretch.

Our socially-aware caching strategy improves drastically the performances of CCNx. CCNx Cache Hit barely reaches 20%, while it reaches about 50% with our strategy. *SACS* enhances the performances of the default caching strategies in CCNx by 2.5 times.

The planet-scale experiment results match the previously obtained simulation results (Section 6.4.2 and 6.4.3), and it confirms that our socially-aware caching strategy *SACS* improves the caching performances of CCN. It also points out that the use of social information (i.e.: the importance of users in a social network) improves the performances of a Future Internet based on the Content Centric Networking architecture.

6.5 Discussion

In the previous Chapter, we proposed a common evaluation scenario and compared five caching strategies. During the evaluation of SACS, we have only compared it with LCE. We expect to have the results against other strategies in a near future.

It is important to note that results are not the same in the previous chapter and in this chapter. The traces are not the same and in the previous chapter we have used client-server traces where every request is made from a source node to a target. While the traces in this chapter were created with SONETOR and represent traffic from a social network.

SACS has shown good results in terms of Cache Hit and Stretch. It achieves the same level of number of Expired Elements with regards to LCE as well. Although, SACS performs poorly in terms of Diversity, we expect to keep working on its proactive mechanisms. SACS creates copies across the path towards the friends of the Influential publishers. This fact provokes many replicas of the same content to be created and reduces the Diversity of content available in the caches.

From our previous experiments, we show that *SACS* improves drastically the caching performances of CCN and naturally reduces the distance to get the content. In addition, our PlanetLab experiments demonstrate that *SACS* enhances the caching performances of CCN by 2.5 times. Besides, we discuss here some open issues regarding our caching strategy.

SACS discriminates the content from popular users and caches it pro-actively in the network. There is a trade-off between the caching performance and the diversity of elements in the caches. By privileging content from important users, *SACS* reduces the number of sources of content, and thus the number of distinct elements stored in the network. However, regarding the number of Expired Elements, *SACS* shows the same level of performances as LCE. With *SACS*, the replicated content is the one from popular users. These pieces of content are still requested frequently by other users and multiple copies into caches does not induce larger waste of space

than using LCE.

Our strategy also requires detecting Influential users. It is therefore essential to compute accurately the centrality measure in the social network. In order to avoid a centralized computation of this measure and to have a complete knowledge of the social network, several studies [78, 79, 80] address the topic and in particular, in [78], they propose a decentralized version of the Eigenvector algorithm in which scores are calculated locally by every node and normalized by exchanging messages. Avrachenkov et al. [79] and Acharyya et al. [80] also show that it is feasible to compute a good estimation of the PageRank without the entire knowledge of the network.

Another matter of importance is the privacy of users. Computing a centrality measure involves the exchange of users' relationships. Thus, users are exposing sensitive information and partially unveiling their privacy. As the management of privacy is a subject of high interest, it has been investigated in other research areas. For instance, in the context of databases, Differential Privacy [81] hides sensitive information while it still allows answering queries on databases. Such kind of mechanism can also be used with our strategy in order to preserve users' privacy.

For large bulks of data (such as video), congestion control mechanisms could be put together with our strategy. For instance, Cho et al. [69] algorithm decides the number of chunks to be cached according to the content popularity. *SACS* may use congestion control mechanisms along with its proactive content replication functionality in order to avoid overloading the network.

6.6 Summary

We proposed, in this Chapter, *SACS*, a Socially-Aware Caching Strategy for Content Centric Networks. *SACS* is a caching strategy based on social networks information. *SACS* uses social information to privilege Influential users in the network. SACS privileges influential users by pro-actively caching the content they produce.

Based on extensive simulation experiments, we showed that our caching strategy improves significantly the caching performances of CCN with regard to the Cache Hit and Stretch, while it keeps a similar level of performances for the Expired Elements. Furthermore, we implemented *SACS* on CCNx and performed experiments on PlanetLab. *SACS* improved the caching performances of CCN by 2.5 times in a real testbed. These results point out that social information is a valuable knowledge's base to improve a future content-centric Internet.

As future work, we expect to work on two aspects of *SACS*. First, *SACS* is a caching strategy designed to operate into an environment with social networks. It is not clear how *SACS* will operate into a mixed scenario with client-server traffic and social networks traffic. Second, we pretend to improve the pro-active mechanism by other techniques that may achieve same level of performances but increase Diversity of the strategy. Finally, the inclusion of social information should not be limited to CCN caching related issues. Routing alternatives may as well consider social features to select the best path. Increasing the cache diversity is also a topic to be investigated.

7

General Conclusion

Contents

7.1 Contributions summary 94
7.1.1 Fair-Comparison of the State of the Art Caching Strategies 94
7.1.2 Popularity-Based Caching Strategies 94
7.2 Perspectives 95
7.2.1 Comparing against current Internet methods 95
7.2.2 Model of Social Networks 95
7.2.3 Popularity-Based Strategies 96
7.2.4 Keep digging into social network information 96
7.3 List of Publications 97

During the last two decades, Internet usage skyrocketed and diversified in a manner never seen before. Internet passed from a small experimental network to a world wide network.

The original Internet architecture has been altered throughout the time with incremental changes to provide all the services that Internet offers today. It passed from serving non-profit scientific and military activities to be an important part of the world economy. Indeed, Internet accounts today for 3.4% of GDP in G8 and BRICS countries. It adapted from exchange of files and messages to serve hours of digital video and multi-party video conferences. Nevertheless, the communication paradigm has remained unchanged and the network operation is based on an end-to-end principle.

In this context, Information Centric Networks have been proposed as a new architecture to replace the outdated Internet model. ICN changes the end-host approach for content centric approach: content is placed at the center of the scene. Several ICN architectures have been proposed and CCN has received most of the attraction from the community.

CCN offers caching capabilities at every node and therefore CCN acts as a network of caches. Dealing with a network of caches involves many research challenges. All along this thesis, we have focused on increasing performance of CCN network caches. We have evaluated different mechanisms and we have experimented the use of social network information to improve general CCN performance.

In this chapter, we first summarize our contributions in Section 7.1. Second, we describe the scientific perspectives of our work in Section 7.2. Finally, we list all the publications produced during the achievement of this doctoral thesis in Section 7.3.

7.1 Contributions summary

7.1.1 Fair-Comparison of the State of the Art Caching Strategies

During our research, we have found difficulties to cross-compare the caching contributions for CCN. A vast number of simulation environments were found in the literature. We have studied parameters ranging from popularity models to topologies passing from alternatives configurations of caches. Once all this information was recompiled, first, we aimed at homogenizing results of caching strategies in a common evaluation scenario and discovering the best caching strategies for every situation. Thus, depending on the environment we may adapt our caching strategies to improve network performance. Based on the work of standardization groups [25], we revisited various simulation parameters. For popularity models, we propose four alternative configuration of the Zipf model with four α values ranging from 0.65 to 2.0. For the topologies, we have selected four ISP-level topologies (Abilene, GEANT, DTelecom and Tiger) because we expect that ISPs are the target for the change of Internet architecture. With regards to the catalog sizes and cache sizes, we have set a catalog of 10^6 and cache sizes relatives to the catalog size. Second, we compared all the metrics available in the literature. We cross-compared them and found correlations between many of the metrics and we have chosen the most appropriate ones for evaluations. In particular, we highlighted three metrics: Cache Hit, Stretch and Diversity. These metrics are used all along the thesis because they reflect efficiency of caches and level of different content stored across the caches.

With a common evaluation scenario set, we have worked in assessing the mechanisms proposed in the literature. Their understanding and its subsequent implementation led to cross-comparing the evaluations. We were the first to compare the proposals and discovered flaws and worst-case scenarios on the CCN community studies. We summarize the results and show in case-by-case basis which caching strategy is appropriate for each scenario. For instance MAGIC outperforms other caching strategies but its computational cost is high and it can be expensive to be implemented in real CCN nodes. However, MAGIC could be considered as a boundary function for further comparison and evaluation. Even though LCD and Cache "Less" for More are good candidates to be used as caching strategy for CCN, LCE and ProbCache are more appropriate with low Diversity scenarios while LCD and Cache "Less" for More are better candidates with high Diversity. Selecting the best strategy depends on the objectives of the CCN network.

This comparison enables us to discover performances of caching strategies in particular scenarios. However, we discovered that other caching strategies may be used to improve global performance in CCN. In particular, caching strategies based on popularity have not been fully exploited.

7.1.2 Popularity-Based Caching Strategies

Many alternatives can be designed to improve performance of Content Centric Networking. In this thesis, we have proposed two caching strategies based on popularity: MPC and SACS. MPC is a caching strategy based on popularity of content while SACS bases on popularity of users.

Most Popular Caching Strategy

Based on popularity of users, we have presented the Most Popular Caching (MPC) Strategy. MPC caches only popular content and by purging caches of unpopular content, it saves space for other pieces of content and avoids the waste of resources with barely demanded pieces of content.

Regarding the results of MPC, we configure a similar simulation environment to the one proposed in [64]. With the simulation environment defined, we tuned empirically the internal parameters of MPC.

Then, we resort to compare MPC with the LCE caching strategy. Before MPC, LCE caching strategy showed the best results. We compared both strategies in two environments with different catalog size. A small catalog of 10^4 files and another of the size of the catalog of Youtube (10^8). Our simulation experiments showed that MPC outperforms the default LCE strategy. MPC achieves a higher Cache Hit and still reduces drastically the number of replicas in the network. By caching less data and improving the Cache Hit ratio, MPC improves network resources consumption.

Socially-Aware Caching Strategy

We proposed in this thesis *SACS*, a Socially-Aware Caching Strategy for Content Centric Networks. *SACS* uses social information and privileges Influential users in the network by proactively caching the content they produce. To the best of our knowledge, SACS is the first caching strategy that uses social information to improve the performance in CCN.

Based on a thorough simulation process, we proved that our caching strategy improves significantly the caching performances of CCN with regard to the Cache Hit and Stretch, while it keeps a similar level of performances for the Expired Elements. Furthermore, we implemented *SACS* on CCNx and performed experiments on PlanetLab. *SACS* improved the caching performances of CCN by 2.5 times in a real testbed. These results point out that social information is a relevant piece of information to improve a future content-centric Internet.

7.2 Perspectives

7.2.1 Comparing against current Internet methods

During our research work, we have analyzed several research aspects of caching in Content Centric Networks. In particular, we have set common ground for evaluation approaches in CCN and we have proposed two popularity based caching strategies. We have discovered what are the best methods to be applied in different environments. However, comparisons against current Internet approaches may be needed in order to discover the real impact of the approaches. Several research challenges may be addressed. The real impact of replacing parts of the current Internet with CCN is not clear: it is not clear neither where the Internet components will be replaced by CCN ones. CCN may be used by an Internet Service Provider or at local scale in home-made networks. There are still many questions opened about major advantages of using CCN. Or even other research topics about ecological or economic aspects. The energy consumption required by CCN nodes: it is not clear if the improvement in performance will justify the increase in the resources needed by CCN nodes.

7.2.2 Model of Social Networks

In the Chapter 6, we presented a social network model and SONETOR, a social network traffic generator. The represented model needs to be completed in many aspects. Currently, the model operates with two types of operations: publish and retrieve. The transition between one activity and the following is performed statistically with markov chains. In [86], we have studied the patterns of publication in the Pinterest social network. This contribution permits to represent

a social network model where user publishes following the behavior of Pinterest. However, we have only considered publications. We expect that this contribution permits us to contact OSN companies to improve the social network model.

7.2.3 Popularity-Based Strategies

Popularity of Content

In Chapter 5, we presented a caching strategy based on popularity of content: MPC. MPC uses a suggestion mechanism for replicating content: it pushes popular content towards every direct neighbor. This mechanism may be improved according to the configuration of the topology graph to avoid bursts of replications the same piece of content. According to the results, MPC has shown better results in certain topologies. there are insights that MPC works better with low degree topologies.

In Chapter 6, we have evaluated SACS not only in simulation environment but in a real testbed too. As the experimentation platform for SACS is similar to the platform for MPC, we expect to experiment MPC in the PlanetLab platform in the near future.

Furthermore, MPC was tuned in a previously fixed environment. However, this tuning of parameters must be deployed dynamically: while the popularity of content changes, MPC must adapt to the new environment. Having an adaptative MPC mechanism, we expect to combine it with the results of other caching strategies in Chapter 5. In this way, we may deploy an alternative caching strategy that alters between the state of the art mechanisms according to the scenario.

Popularity of Users

During the evaluation of our socially-aware caching strategy, we detected that the Diversity of SACS was low. It means that most of the caches store the same pieces of content many times over. This fact is due to the applied replication mechanism. SACS creates copies of Influential users towards the path to their friends. Thus, many similar copies are created across the network and it decreases the diversity. We expect to use a smarter solution such as the one propose in Cache "Less" For More into SACS with the goal of reducing Diversity.

Another aspect to keep working about this caching strategy is that the evaluation was performed using a social networks environment. SACS works only when users interact in a social-oriented manner. We seek to evaluate first SACS in a mixed environment with a client-server and social networks traffic. In a second step, SACS may be combined with a caching strategy from previous chapter to react selectively according to the type of traffic (social or client-server).

7.2.4 Keep digging into social network information

The social networks have changed the way we interact over the Internet. In many countries, many users check their Facebook pages several times a day and stay informed on their friends updates immediately. Social Networks have changed the Internet and they may keep changing it. To the best of our knowledge, we are the first to study the use of social network information to improve caching strategies in CCN. During this study, complications arised rapidly in the simulation environments due to the source of the social information. This social information was extracted from many datases available publicly. As they are public, most of the data is anonymized. It means that we were not able to extract every possible correlation from the information and we have to adapt the analysis to the data presented by the authors of the datasets. The way of

studying social networks and using its information requires collaboration between private and public institutions. We expect that publication of SACS may allow to start discussions with private OSN companies in order to retrieve information and continue the study.

In addition, sociological theories and studies should be revisited to understand behavior of users and create innovative mechanisms. These mechanisms may range from caching to name routing problems, passing from quality of services and economical issues such as the discovery of trends.

7.3 List of Publications

International peer-reviewed conferences

- **César Bernardini**, Thomas Silverston and Olivier Festor. *MPC: Popularity-based caching strategy for content centric networks.* In Proceedings of the IEEE International Conference on Communications (IEEE ICC 2013), 2013, Budapest, Hungary.
- **César Bernardini**, Thomas Silverston and Olivier Festor. *SONETOR: a Social Network Traffic Generator.* In Proceedings of the IEEE International Conference on Communications (IEEE ICC 2014), 2014, Sydney, Australia.
- **César Bernardini**, Thomas Silverston and Olivier Festor. *Socially-Aware Caching Strategy for Content Centric Networking.* In Proceedings of the IFIP Networking Conference 2014, 2014, Trondheim, Norway.
- **César Bernardini**, Thomas Silverston and Olivier Festor. *A pin is worth a thousand words: Characterization of publications in Pinterest.* In Proceedings of the IEEE International Wireless Communications & Computing Conference 2014 TRAC Workshop, 2014, Nicosia, Cyprus.
- **César Bernardini**, Thomas Silverston and Olivier Festor. *On Caching in Content Centric Networking: a Comparison Study.* Submitted to the IFIP Networking 2015 Conference, 2015, Toulouse, France.

Workshops

- **César Bernardini**, Thomas Silverston and Olivier Festor. *Cache management strategy for CCN based on content popularity.* In Proceedings of the International Conference on Autonomous Infrastructure, Management, and Security (AIMS 2013) PhD Workshop. 2013, Barcelona, Spain.

Posters

- **César Bernardini**, Thomas Silverston and Olivier Festor. *Using Social Network Information into CCN.* Presented at IEEE INFOCOM 2013 NOMEN Workshop, 2013, Turin, Italy.

Summer-Schools

- **César Bernardini**, Thomas Silverston and Olivier Festor. *Towards Popularity-Based Caching in Content Centric Networks.* In Proceedings of RESCOM 2012 Summer School, 2012, Vittel, France.

- **César Bernardini**, Thomas Silverston and Olivier Festor. *Using Social Information into ICN*. In Proceedings of RESCOM 2013 Summer School, 2013, Île de Porquerolles, France.

A

Impact of OSN into Content Popularity Model

During the last years, Online Social Networks (OSN) started gaining popularity to become today into one of the most visited websites in the Internet [134]. With the introduction of OSN, users changed their interaction manners from client-server requests to request on last friends updates. This scenario was represented with a social network model detailed in Chapter 6. This model was instantiated with a tool: SONETOR, a social network traffic generator.

Every time we simulate a network where users request pieces of content, we use popularity models. These popularity models decide what and how often pieces of content are requested. We aim at discovering the impact of social network into the popularity model. Say, the changes provoked in the popularity model by the use of social networks traffic.

The contributions of this Annex are the following:

1. We show an use-case of SONETOR. We use the SONETOR tool to generate traffic that are injected in our experiments.
2. We evaluate the changes produced by a social network model into the popularity model. This evaluation comprises two parts:
   (a) We simulate an scenario where all the traffic is social network traffic. After the execution of the scenario, we evaluate the changes produced into the popularity model. We show the evolution of the probability distribution functions used to simulate the traffic.
   (b) We then continue with a more complex scenario where regular (client-server) and social network traffic are mixed together. The evaluation are similar to those previously presented but this time into a more realistic scenario.

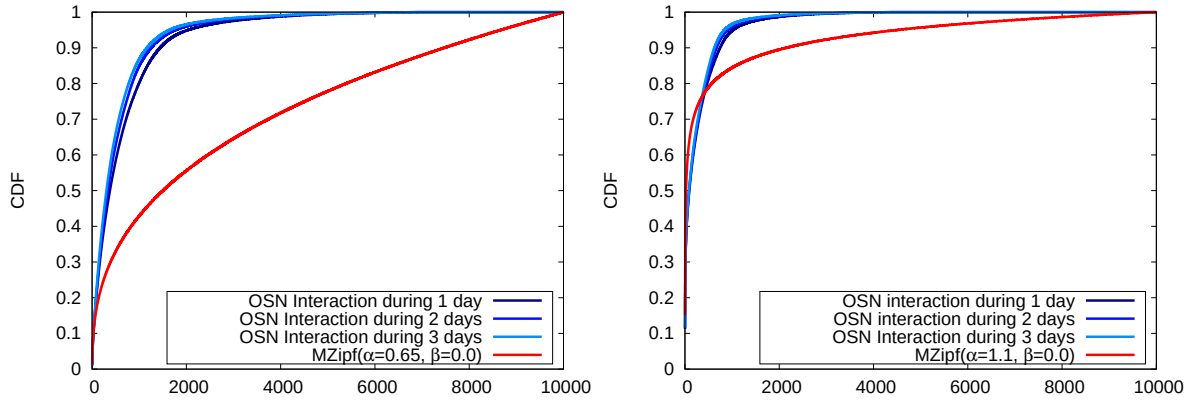
This annex is organized as follows. The Section A.1 describes the simulation environment used for the experiments. The Section A.2 details the results of the evaluation in the stand-alone scenario while the Section A.3 presents the results of the evaluation in the mixed scenario. Finally, the conclusions of the annex are shown in Annex A.4.

A.1 Simulation Parameters

In this section, we describe the notation and simulation parameters that we use for our experiments. All the simulation parameters are summarized in the Table A.1.

Social Network Activity Traces	
Number of sessions NS	Zipf ($\alpha = 1.792, \beta = 0.0$)
Number of activities NA_i	Zipf ($\alpha = 1.765, \beta = 4.888$)
Inter-session time t_{IS_i} (s)	LogNormal ($\mu = 2.245, \sigma = 1.133$)
Inter-activity time $t_{IA_{ij}}$ (s)	LogNormal ($\mu = 1.789, \sigma = 2.366$)
Catalog Configuration	
Size	10^4 Pieces of Content
MZip Parameters	$\alpha = \{0.65, 1.1\}$
	$\beta = 0.0$
Social Network graph	
#Users	4,039
#Links	88,234
#Avg. Degree	44

Table A.1: Simulation Parameters



(a) MZipf ($\alpha = 0.65$, $\beta = 0.0$) (b) MZipf ($\alpha = 1.1$, $\beta = 0.0$)

Figure A.1: Stand-alone Scenario: impact of the OSN on the content popularity model

All along this annex, we contrast results using different content popularity models with and without social networks traffic. From now and so on, the scenarios with content popularity model based on social networks will be called *OSN* traffic while scenarios with popularity model based on a MZipf distribution will be called *regular* traffic. All the experiments are performed with a catalog counting 10,000 pieces of content.

In order to model the social relationships between users, we resort to a Facebook data set publicly available [82]. The data set consists of 4,039 users, 88,234 friend relationships and each user counts in average 44 relationships.

In the following, we analyze the impact of social networks into the content popularity model. First, we investigate the probability distribution found on a stand-alone OSN traffic scenario. Second, we study a mixed scenario where OSN and regular traffic coexist. And last but not least, we show the presence of flash crowds in the stand-alone OSN traffic scenario.

A.2 Stand-alone Scenario with OSN Traffic

We aim at discovering the changes on the content popularity model provoked by the interaction of users within social networks. We start the assessment on a stand-alone scenario where all the traffic comes from social networks. In this experiment, we reproduce *SONETOR* traces to analyze the impact of social networks into the content popularity model. We model content popularity with a MZipf distribution ($\alpha = 1.1$; $\beta = 0$). The consumption of content is realized by the users; users receive an update of their friends' publication. The users do not consume all the content from their friends but only a small subset based on influence model [89]. Then, we bound the generated synthetic traces to 1, 2 or 3 days. Once the experiments execute the traces, we build a probability distribution with the consumed content. We proceed to draw the probability distribution obtained and to fit the curve with a new power-law distribution.

In Figures A.1b and A.1a, we show charts with distinct MZipf configurations. The plotted curves correspond to the Cumulative Distribution Function (CDF) of the content probability distribution. The charts are built with the number of requests for every piece of content. The red line represents the original content popularity model (*input*) while the blue lines stand for the obtained content popularity model (*output*) after 1, 2 or 3 days of social network interaction.

Using a popularity model MZipf ($\alpha = 0.65$; $\beta = 0$), the 90% of most requested content consists of 7,466 pieces of content. While using OSN, the 90% of most requested content get reduced to only 1,165 pieces. It means the subset of most popular content get reduced to 16% of its original number of pieces. It is important to remark as well that in the OSN case, only 6,992 pieces of content were consumed: it means 30% of the content is completely ignored.

Those results are confirmed using the model MZipf($\alpha = 1.1$; $\beta = 0$), the introduction of OSN produces that the 55% of content is ignored and the subset of most popular content is reduced to 29% of its original number of pieces.

From these figures, we observe how the ratio of popular elements got decreased. the viral effect of OSN provokes many users to consume less diverse content and it strengthens the importance of popular elements. As we see in both cases, the subset of most popular contents is reduced in important proportions. The social networks provoke the creation of *super-popular* content. These pieces of content are highly demanded and may have an important impact on the general behavior of the network. In the OSN, there are many pieces of content that are completely ignored and never consumed. This fact reveals that many of the content are irrelevant to most of the users. Even more, in the social network, the number of published content is a subset of all the pieces of content found in the popularity model. In other words, the content found in social networks is subset of all the content found in the Internet.

In the Table A.2, we present the obtained distribution parameters. We observe that the alpha parameters have grown significantly, which means a few popular content increases its popularity while most of the other stay unpopular.

A.3 Mixed Scenario with OSN and Regular Traffic

We have already shown the impact of social networks in an environment where all the traffic is produced by social networks. Now, we are interested in mixed scenarios where traffic is composed of two types of traffic: social network and regular traffic. OSN and regular traffic are going to coexist in the near future. We argue that the penetration of social networks into the overall traffic will incur changes in the content popularity model. These changes are analyzed in this section.

Original Content Popularity Model (*input*)			Obtained Content Popularity Model (*output*)	
Distrib.	Parameters	Period	Distrib.	Parameters
MZipf	$\alpha = 0.65$	1 day	MZipf	$\alpha = 5.27$
	$\beta = 0.0$			$\beta = 2114.89$
MZipf	$\alpha = 0.65$	2 days	MZipf	$\alpha = 5.59$
	$\beta = 0.0$			$\beta = 2009.75$
MZipf	$\alpha = 0.65$	3 days	MZipf	$\alpha = 5.19$
	$\beta = 0.0$			$\beta = 1664.63$
MZipf	$\alpha = 1.1$	1 day	MZipf	$\alpha = 2.43$
	$\beta = 0.0$			$\beta = 181.23$
MZipf	$\alpha = 1.1$	2 days	MZipf	$\alpha = 2.55$
	$\beta = 0.0$			$\beta = 191.67$
MZipf	$\alpha = 1.1$	3 days	MZipf	$\alpha = 2.59$
	$\beta = 0.0$			$\beta = 189.62$

Table A.2: Distribution parameters for the stand-alone scenario

To this end, we simulate scenarios with different ratio between *regular* and *OSN* traffic (i.e., 100%-0%; 90%-10%; 80%-20%; 50%-50%; 0%-100%). We then present in Figure A.2 the CDF of the content popularity obtained with all the mixed scenarios. We represent the execution of the social network activity traces in a three-days period. For lack of space, we only show the chart for regular traffic with MZipf ($\alpha = 0.65$), still in the Table A.3 the other configurations are shown.

We fit the mixed scenarios with a MZipf function. We summarize the obtained MZipf parameters in the Table A.3. As seen in Fig. A.2, the curves for regular traffic and for the mixed scenario with 10% of OSN traffic seem similar. The mixed scenario with 10% of OSN traffic has an apparently minimal impact in the MZipf obtained MZipf parameter: the α parameter passes from 0.65 to 0.71, which means the 20% of the most popular content passes from 151 pieces of content to 142. If we consider the 90% of all the content, it passes from 7,458 to 7,264 pieces of content: a reduction of 9% and 3% in the most popular contents respectively. The same phenomenon is also observed for the OSN traffic ranging from 20% to 100% but at higher scale. For instance, for the 50% OSN traffic, the α parameter passes from 0.65 to 1.14.

From this experiment, we observe that OSN traffic has an impact on the content popularity model. While we increase the ratio of OSN traffic, we observe the growth of the α parameter (Table A.3). A higher value for the α parameter implies a smaller subset of *super* popular content (i.e. the shape of the curve tends to the left side). Then with OSN traffic, there is a small subset of *super* popular content that is highly requested while the other content stay unnoticed.

This observation can have a significant impact for the Caches of the Internet. Indeed, with the rise of caching architectures such as Content Delivery Networks (CDN) and Content Centric Networks (CCN), in-network caching has become an important issue for the Internet. By reducing the subset of *super* popular content, it reduces the number of relevant content to be stored into Caches. It then alleviates the load on the Caches, saves the resources (e.g.: memory) and improves their performances.

Original Content Popularity Model (*input*)	Obtained Content Popularity Model (*output*)		
MZipf0.65 + 10%OSN	MZipf	$\alpha = 0.71$	$\beta = 10.57$
MZipf0.65 + 20%OSN	MZipf	$\alpha = 0.81$	$\beta = 38.90$
MZipf0.65 + 50%OSN	MZipf	$\alpha = 1.14$	$\beta = 123.81$
MZipf0.65 + 100%OSN	MZipf	$\alpha = 5.19$	$\beta = 1664.63$
MZipf1.1 + 10%OSN	MZipf	$\alpha = 1.11$	$\beta = 0.40$
MZipf1.1 + 20%OSN	MZipf	$\alpha = 1.16$	$\beta = 1.78$
MZipf1.1 + 50%OSN	MZipf	$\alpha = 1.38$	$\beta = 16.09$
MZipf1.1 + 100%OSN	MZipf	$\alpha = 2.59$	$\beta = 185.92$

Table A.3: Distribution parameters for the mixed scenario

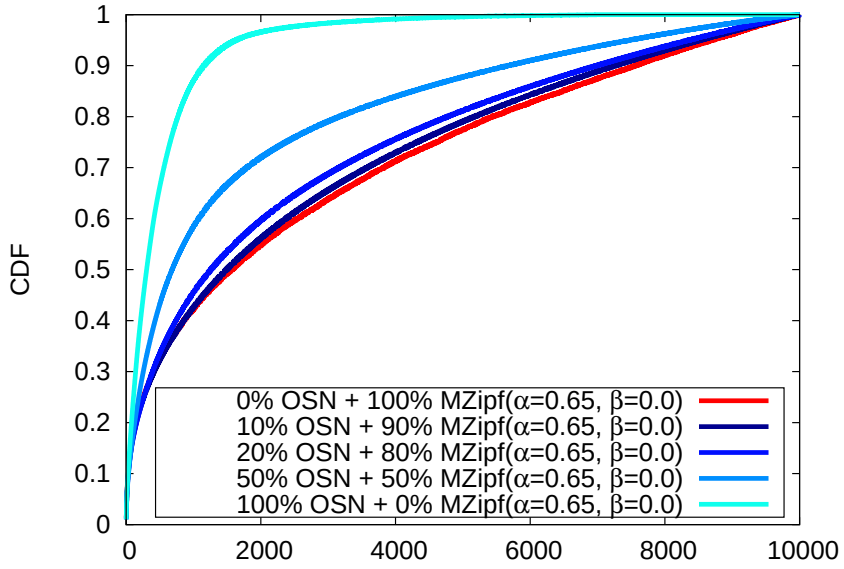


Figure A.2: Mixed Scenario: impact of the OSN on the content popularity model

A.4 Conclusion

In this Annex, we present an use-case of *SONETOR* and study the impact of social relationships in the content popularity. The social relationships between users enforce that many pieces of content become popular and are spread massively throughout the OSN while many others passed unnoticed. It has a major impact on the content popularity model traditionally used in the Internet. With SONETOR, we showed that OSN privileges a subset of super-popular content. By reducing the number of popular content, network caches can improve their performances with the same storage capabilities. It is an important result as in-network caching is nowadays largely studied with network architectures such as data-centers, Content Distribution Networks or Information-Centric Networks.

B

Characterization of Publications in Pinterest

During this thesis, we have highlighted the importance of social networks in the current Internet. We have presented a social network model and a tool that implements it in Chapter 6. In the Annex A, we shows an use-case for the SONETOR tool by showing the impact of social traffic in the popularity model. In this annex, we crawl the Pinterest social network in order to feed the social network traffic simulator.

Pinterest is already ranked among the top 30 most popular websites in the world and is ranked in the top 15 just for the United States [135]. Even more, Pinterest has just raised $225 millions to fund its international expansion [136] and is expected to become a major actor in the OSN landscape in the near future. In Pinterest, all the users share images (called *pins*) within its community and one of its main features is the ease to retransmit an image (i.e.: *repin*). At every moment, the user watches at least ten images with the possibility to *repin* them. In this OSN, users are highly attired by the new images and their main activity consist on repining images.

In this annex, we analyze and characterize the fastest ever growing social network in the world: Pinterest [132]. To this end, we performed an extensive measurement of the Pinterest website. Indeed, we crawled Pinterest during several months and collected 18 GB of metadata, 600,000 users' profiles, 450,000 images and 22.5 millions of activities [2]. We leverage all this relevant information and we provide an extensive characterization of users' publications in Pinterest. The type of activities performed by social network users, which includes a representation of sequenced activities under the social network. We then give an insight into the importance of the Pinterest architecture to leverage traffic into external websites. From the image collection, we provide information on the filesize and popularity and we classify images into categories illustrating the nature of Pinterest. We assess this result with a lexical analysis of the words used in the description of the images.

The rest of the annex is structured as follows. Section B.1 presents the Pinterest OSN. Section B.2 describes our measurement methodology and introduces our dataset. In Section B.3, we study the users' publications. Section B.4 details the related work. Finally, we conclude the annex in Section B.5 by summarizing our contributions and presenting future works.

[2]The dataset will be freely available upon requests.

B.1 Pinterest Overview

Pinterest is an alternative Social Network to Facebook and Twitter where the images are at the center of the scene. Pinterest allows its users to manage photo collections in different boards. Its users may follow any other user, and the followed user does not need to follow him back. As in most of the OSN, every user has a time-line where last friends publications are shown. Following a user implies that his last publications appear in the time-line. As all the publications contains photos, the time-lines are composed mostly of images and little descriptions.

A *pin* is an image users add to Pinterest. A *pin* may be accompanied by a text description and a link pointing to some web-address.

Every *pin* may be re-published. Once the image is re-published, it becomes a *repin*. The users classify its new *repin* into a *board* of his own and he may personalize the text description. Any pin may be repinned, and all *pins* link back to their source.

A *board* is a collection of *pins* (and *repins*) where users manually organize their pins. Boards may be public -by default- or private. Users may invite other users to publish on their boards.

B.2 Measurement Methodology

We crawled information from Pinterest starting from 15th January until 30th March, 2013. We rely on scrapy -an open source web scrapping framework- to collect profiles, pins, repins, boards and its inherent information. Additional, we collected recently performed activities on the social network.

B.2.1 Data Collection

We developed a web crawler to first collect popular pins. Thus, we gathered a seed set of 2,000 pins. We then opted to use a snowball sampling starting from popular pins and their authors and then following author's followers and followees. We detail all the extracted information at every step.

A Pinterest user keeps a short profile about himself. This profile includes a short name, an optional location, a list of other social network accounts, personal web-pages and a personal picture. In addition, we found the number of followers and followees, number of published pins (which includes repins from other users) and the number of boards. We collected 595,562 user profiles.

A Pinterest user has a collection of boards in-which all their pins are stored. These pins may be personal publications or repins found in some user board. Every board has a descriptive name, a number of pins and a number of followers and a URL. We collected 1,135,645 boards. We count an average of 98 pins per board with a high standard deviation (942).

Every pin contains an image, publication date, a URL, the number of likes, comments and repins. Every pins keeps a record of the original author for the pin and from whom it was pinned. Let's imagine user A publish a pin, user B repinned from A's board and user C repinned from B's board. The record from the repin in user C's board will contain A as original author and B as from who it was repinned. We collected 4,411,161 pins. From these publications, 10% (456,959) are pins and the rest 90% are repins. All the images were stored in our dataset and this represents 18GB of file storage.

In every user profile, there is a list of the user' activities. This list contains the information from published pins, repins, liked boards or pins and whom the user is following or whose board the user is following. We retrieved 22,574,642 activities from 444,750 users.

# Followers	# Following	# Pins	# Boards
12,722,935	152	8383	82
8,536,656	1209	85093	249
8,463,394	141	4777	38
7,985,952	215	4592	32
7,798,015	206	22836	96

Table B.1: Top five users in Pinterest.

# Followers	# Following	# Pins	# Boards
1843	200	2415	28

Table B.2: Average Number of Activities for the top 1,000 users.

In February 2013, Pinterest counted 48 million users [138]. As we collected information from 600,000 users' profiles, we can estimate that our sample represents 1.25% of the social network. We estimated the total number of pins and boards in 115 billion pins and 1.3 billion boards.

B.2.2 High Level Characteristics of Pinterest Dataset

In this section, we give an insight into the main characteristics of the OSN: the connection between users, the number of publications per user and the most popular users.

Figure B.1 shows the complementary cumulative distribution function (CCDF) of pins, followers and followees.

We first describe the number of pins. 90% of the users has less than 5,200 *pins*. Only 13 users have being found to have more than 100,000 pins.

Regarding to the number of following users, 90% of the users has less than 350 following users. Only 1% of the users follows more than 2,000 users.

With regards to number of followers, 90% is being followed by less than 550 users while only the 0.01% by more than 1,000,000.

We mentioned only a small subset has more than 100,000 followers. We now focus on the 5 most important users in Pinterest and we present them in the Table B.1. We consider the five most followed users in our dataset. All of them have more than 7 millions of followers when the number of people they follow barely surpass 200. We discover that five most followed users are girls. In the rest of the network we have found that 79% of users are women and only 21% are men. This fact has been studied in the literature [101, 102] and it is out of the scope of this study.

We consider the 1000 most popular users in Pinterest and we compare their number of pins, repins, boards and followers in Table B.2. We are surprised that most of the popular users does not follow more than 200 users, and this fact is correlated with the top five users analysis. Users are actively publishing pins counting in average for 2,400 pins per user which are stored in an average of 28 boards.

B.2.3 Who publishes in Pinterest?

We were interested in discovering who publishes in Pinterest. We broke the population using the ratio $\frac{followers}{following}$. If the ratio is 1.0 means that the user follows and is followed by the same number of users. The higher the ratio, the user is more popular. While the ratio tend to zero, the user is dedicated in watching others.

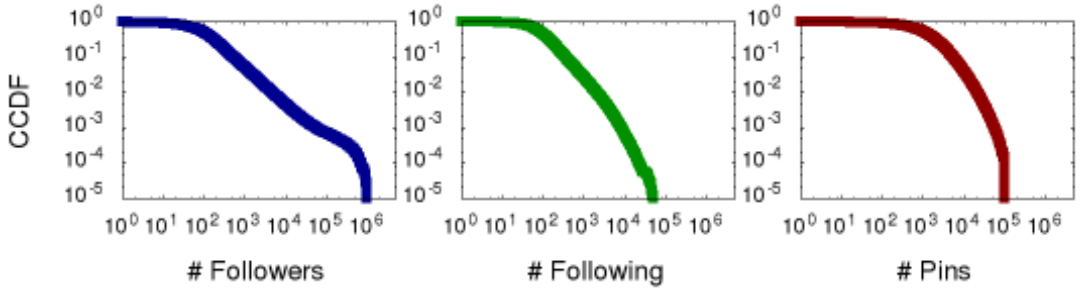


Figure B.1: Probability distribution for pins, followers and followees (CCDF).

In the Figure B.2, we present a CDF of the number of pins (y) with respects to the ratio $\frac{followers}{following}$. We split the chart according to the value of the ratio showing the most popular users in the upper side and the users dedicated in watching the others at the bottom. We found four groups of users. The users with ratio $\frac{followers}{following} < 0.2$ are rarely publishers (4% of the pins) and they represent 11% of the population. The group of $0.2 <= \frac{followers}{following} < 1$ includes 39% of the population while they publishes 32% of the content. $1 <= \frac{followers}{following} < 8$ correspond to 43% of the population and produce 51% of the content. The last group gather the most popular users with a ratio bigger than 8, they are active publishers 11% of the pins and they only represent the 5% of Pinterest users.

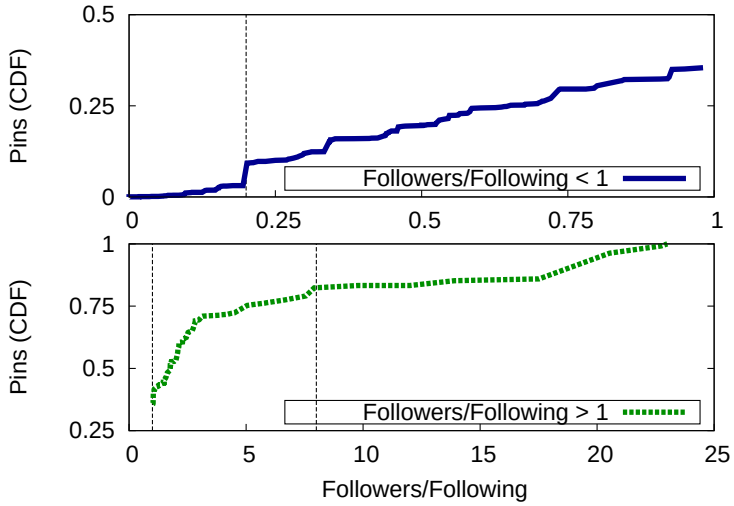


Figure B.2: Pinterest's Pinners.

B.3 Study of Users' Publications

In this section, we study and characterize the publications of Pinterest: we begin by studying the type of activities in the OSN. We give an insight into the impact of urls into Pintest which have a major impact on Pinterest's importance. We then concentrate on images of Pinterest and we study filesize, popularity and category of images. We end up with a lexical analysis of publications description.

Activity	Pin	Repin	Like	Follow
ratio	8%	64%	20%	8%

Table B.3: Activities in Pinterest.

B.3.1 Activities: Users' Publications

Based on the almost 23 millions of activities collected from Pinterest, we study the type of activities performed in Pinterest.

Pinterest has a section Activity in every user profile. This section contains a list of the user's activities in chronological order. The website shows only four activities (*pin*, *follow*, *like*, *repin*). *follow* and *like* include following/liking a pin or a board and other activities such as *to comment* were not listed in the website. The *repin* and *like* activities are considered retransmission activities because augment the visible scope of the publication. *like* adds the pin to user profile's likes section and *repin* to a specific board.

Pinterest website is designed with a concept of *infinite scroll* which means the website automatically loads content as the user goes toward the bottom of the page. As every user may have thousands of activities, we limited our crawler to gather at most 50 items per user. This number may seem small but we preferred collecting more activity transitions from several distinct users to collecting more transitions from a small subset of users.

In the Table B.3, we depict the type of activities captured in our dataset. Of the total of activities, only 8% are new publications (pins). The follow activities have the same proportion with 8% too. Most of the captured activities were likes and repins with 20% and 64% of the total dataset. As their sum represent 84% of all its activities, we can conclude that Pinterest is a retransmission network. In comparison with Twitter where users share their opinions about a trend and eventually shares somebody else's opinion (i.e. retweet), Pinterest users tend to repin most of the times.

One of the main reasons of Pinterest growth and interest, it is due to this characteristic of the network. Pinterest take advantage of the retransmission capabilities proposed by the OSN. Artists, enterprises are hungry for spreading their latest productions and Pinterest seems to have found the right manner to accomplish it. Unlike other social networks such as Facebook where users are eager to participate on their friends publications, Pinterest users are mostly interested in retransmit images they found interesting or fun.

In order to characterize Pinterest publications, we examine the sequence of users publications as a Markov Chain in Figure B.3. We represent every activity as a State. Transitions correspond to the probability to pass from one activity to another. We calculate the transition from State A to State B as follows: we count the number of times every user performs an activity B immediately after an activity A and we then normalize this value. The sum of all outgoing transitions from each state is 1.0.

In almost all states, self-loops have the biggest probability. For example, *pin*, *repin* and *likes* shows probabilities of 0.54, 0.82 and 0.64 respectively. The only exception is *follow* in-which the probability of *repin* is bigger than an immediate subsequent *follow*. In Pinterest, users are constantly being suggested to *repin*: the *pin it* button is present everywhere. Indeed, from *pin*, *follow* and *like* states the probability of a *repin* immediately is high: 0.33, 0.43 and 0.28.

These facts highlight the nature of Pinterest: Pinterest is based on the retransmission of images. We have seen that most of the activities found where *likes* and *repins* which are exactly retransmission activities. Then, we have seen that users tend to *repin* or do a *repin* immediately followed of the activity they are performing.

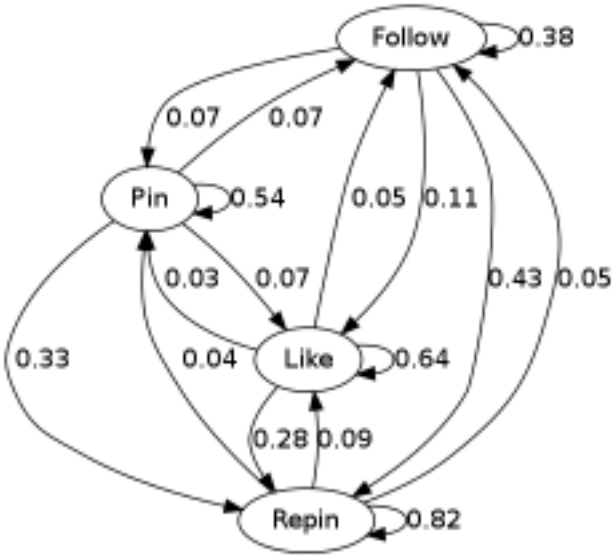


Figure B.3: Model of users' activities in Pinterest.

This chart could be useful to represent the behavior of users in a OSN environment.

B.3.2 Links to external websites

As we previously explained, every published *pin* includes a link and every *repin* links back to their original *pin* and to the original posted link as well. We retrieved 4,441,161 pins which points to 1,867,477 distinct urls: only the 42% points to different links.

We first study the web domains where these pins point out. We discovered that the pins link to 205,613 different domains from which the 94% are pointed only once. It means 4.2 millions of pins links to $\sim 10,000$ domains. We analyze this fact in the Figure B.4 where a CDF of the domains shows 40 domains that concentrates 73% of the links. Most of the domains corresponds to photo storage services such as Tumblr or Flickr, photo search engines such as Google or Yahoo and multimedia services oriented to specific communities such as Polyvore (fashion and decoration), Deviantart (art) or Imgur (humor & quotes).

What is the importance of the links? Pinterest relies on the attraction of images and due to its viral architecture retransmission of content is extremely easy and natural. As we said before, every *repin* links back to the original post link. If a *pin* is *repinned* only one time, the two *pins* are going to point to the same url: the original *pin* and the *repin*. Once the *pin* is *repinned* several times – and the *repins* may be *repinned* as well–, the pins pointing to the original url grow exponentially. This fact has a major effect on search engines such as Google. Google measures importance of websites in the Internet with PageRank. A study on PageRank [109] concludes that a web page benefits from links inside its web community. In this manner, it increases its rank and position in Google results. If we consider Pinterest and its retransmission nature (Section B.3.A), an exponential number of links are going to point out to the original *pin*'s url. This fact implies that the original url benefits from links inside its community which means this url position is going to be clearly affected in Google search results.

To assess the hypothesis, we select arbitrarily 1,000 of pins that have never been repinned and the 1,000 most repinned pins. As we previously mentioned, all the pins have an url pointed to a user-selected website. We compare these urls and we evaluate its PageRank using the

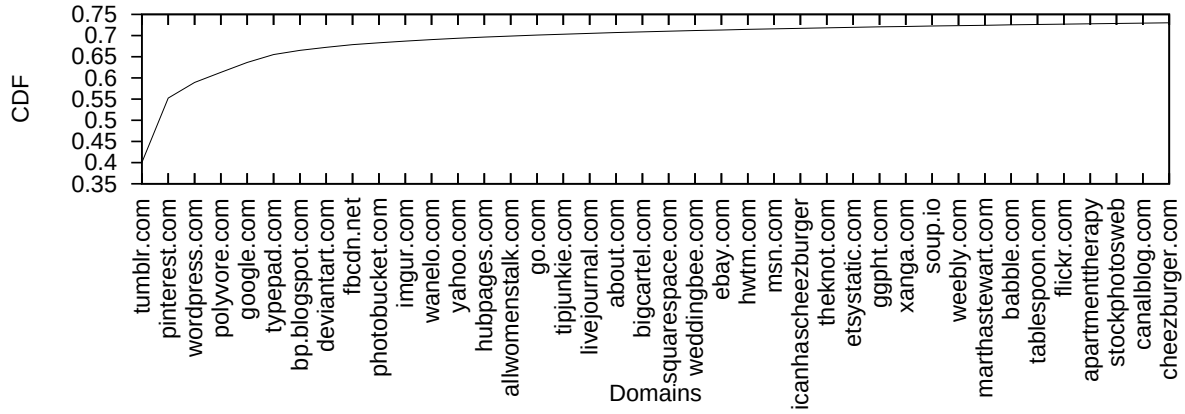


Figure B.4: Most linked domains on the pins.

GoogleToolbar information [133]. PageRank values range between 0 and 1. The Higher the PageRank value is, the more important the website is. The results show that pins that have been never repinned have an average PageRank of 0.21+/-0.01 while the most repinned images have an average PageRank of 0.32+/-0.01. Thus, the most popular pins affect the PageRank of the URL where they point to.

B.3.3 Filesize of Images

Due to the importance of filesize in terms of network transmission, we characterize the filesize of images published in Pinterest publications (i.e. pins and repins). We consider the image shown in the pin's pages. All these images are encoded by Pinterest with the JPEG/JFIF format.

In the Figure B.5a, we show the distribution of filesizes with a Cumulative Distribution Function (CDF). In the x-axis, we observe the filesize in bytes while the y-axis shows the cumulative probability distribution. The chart shows that 70% of published images has less than 50KB of filesize and 90% of the images are less than 80KB. Bigger file found was about 2.5MB and the expected value of 35KB.

The representation of filesize in Pinterest is useful to reproduce common traffic patterns in social networks. Thus, we fitted the distribution with a gamma distribution and we show it in the same Figure B.5a with a dotted green line. The sum of squares prediction errors (SSE) give us a 4×10^{-6} value which affirm the closeness from two functions. The parameters are $\alpha = 2.691$ and $\beta = 68.23$ for the gamma distribution. in order to use it, x values should be normalized before with the function $y = f(x) = \frac{x-426}{205}$.

B.3.4 Popularity of Images

We have currently shown the type of activities and the filesizes found in the Pinterest OSN. In this section, we study the popularity of the images found in pins and repins. Every pin contains an image and every repin is a re-publication of a pin. We aim at finding the probability distribution of images extracted from pins, repins and both of them.

During the crawling process, we downloaded the available image in every pin/repin publication and we calculate the checksum of the image. As every repin had a link pointing to the original pin, we were able to determine the original pin and the original publication of the image.

In the Figure B.5b, we show the Cumulative Distribution Function (CDF) of pins, repins and

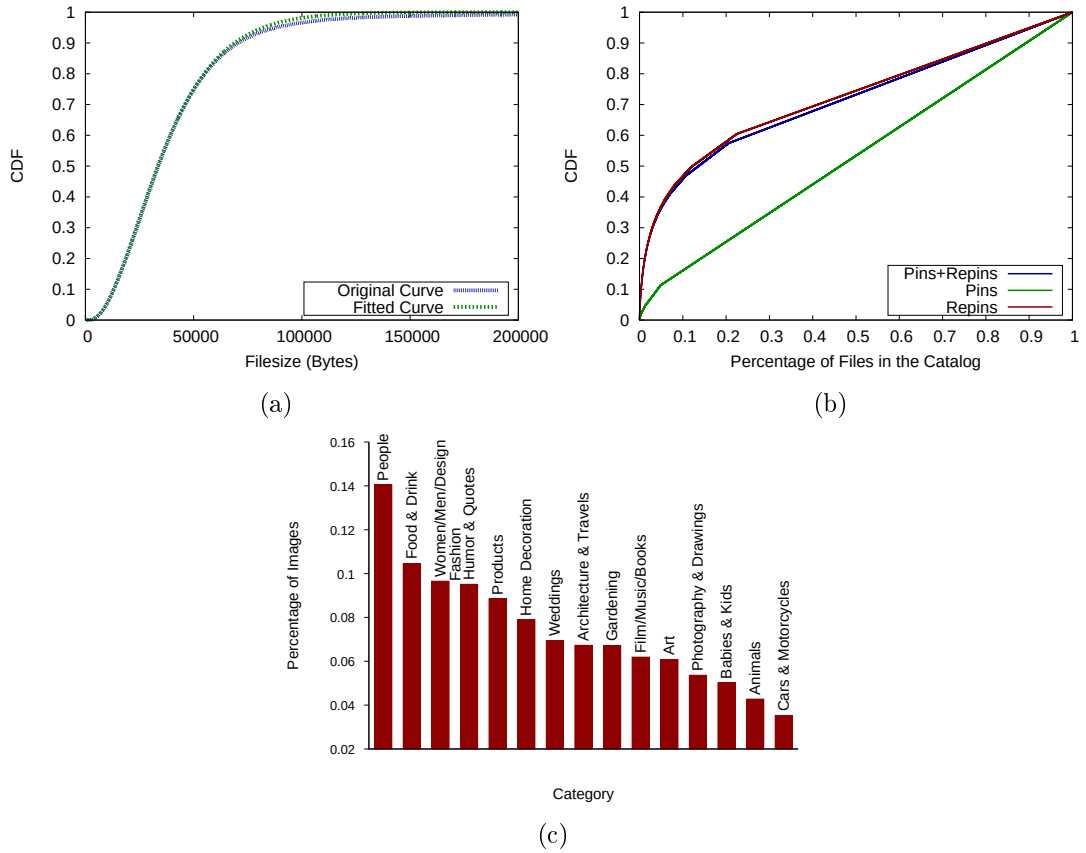


Figure B.5: Study on Pinterest images: Filesize (B.5a), Popularity (B.5b) and Categories (B.5c).

Popularity of	Distribution
Pins	MZipf ($\alpha = 0.26$, $\beta = -0.89$)
Repins	Mzipf ($\alpha = 0.73$, $\beta = 1534.70$)
Pins+Repins	MZipf ($\alpha = 0.72$, $\beta = 1432.27$)

Table B.4: Popularity distribution function.

a combination of both of them. The curve that represents the repins and all the combination of both curves show a similar function of probability. We can observe that the probability of publishing a pin is almost an uniform function: except for 10% of published pins, the rest creates a solid straight line which highlights that all the selected images have the same probability to be chosen. In the other hand, repins have a different behavior: There exist popular elements and then starting from 20% of the elements there is the heavy tail that begins. The popularity of the images can be represented with a MZipf function. In the Table B.4, we show the fitted probability distribution for the popularity of publication of pins, repins and the combination of both of them. The probability distribution found for the pins is really close to an uniform distribution due to the small α parameter. The repins and combination of pins and repins show probability distributions comparable to those found in China VoD and representations of popularity in CDN [59].

B.3.5 Classification of the Images

In Pinterest, users group their pins into boards. These boards are classified into predefined categories. Unfortunately, the categories were not collected during the crawling process. Even though, we aim at describing the content of the images in order to have a better picture of users' publications.

In this section, we classify the published images with a supervised machine learning technique: Support Vector Machines (SVM) [139].

As first step, we analyze the Pinterest pre-defined categories and we define our categories based on them. we defined our categories as follows: *People*, *Food & Drink*, *Women/Men/Design Fashion*, *Humor & Quotes*, *Products*, *Home Decoration*,*Weddings*, *Architecture & Travels*, *Gardening*, *Film/Music/Books*, *Art*, *Photography & Drawings*, *Babies & Kids*, *Animals*, *Cars & Motorcycles*. Many Pinterest categories were ignored because they involve non-visual concepts such as *Other*, *Education*, *Health & Fitness*, *Geek*, *Technology* and *History*. Many other categories were merged: *Women Fashion*, *Men Fashion* and *Fashion Design* into *Women/Men/Design Fashion*; *Architecture* and *Travel* into *Architecture & Travel*; *Humor* and *Quotes* into *Humor & Quotes*.

We continue by a manual classification of the images selecting the category that fit the best with the image. We classified 1,200 images manually. We asked our human-classifiers: what are the categories that fit the best with the image?.

We aimed at building a function that predicts whether an image belongs to a certain category. We then proceed to build a SVM for each one of the categories. Following the procedure proposed at [139], we use as training set the manually classified images and we complete it with images from Google Search Engine using as search tag the name of the category. The feature vector consists of SIFT features. To build the models we resort to Support Vector Machines (SVM) with a Linear Kernel.

The results of the classification of the 429,197 images is shown in Figure B.5c. We were surprised by the low number of images containing people on it: only 15%. This fact shows that most of the images are about things. Then, we remarked the importance of *Food & Drink* (10%), *Women/Men/Design Fashion* (9.9%), *Humor & Quotes* (9.8%), *Home Decoration* (8%), *Products* (9%) and *Weddings* (7.5%). Unlike other social networks like Twitter where most of the content consists on headlines and persistent news [110], classifications given to the images indicate that Pinterest is a social network mostly devoted to leisure and entertainment.

B.3.6 Lexical analysis on Pins' description

We have analyzed the images published in the Pinterest OSN. In this section, we analyze the text published with the images. Every pin is jointly published with a description. This description is manually written by the user.

To analyze this we have considered all the description messages published with images and we perform a lexical analysis. We have decomposed every description message as a sequence of words. We remove stop words such as *the*, *is*, *at*, *that*. This process is commonly done in processing of natural language [90].

There are 4,411,161 pins. 156,174 pin's descriptions (3%) are empty. It means 97% of the pins has a description attached. The longest description was 7,063 characters and the average description has 46 characters with an standard deviation of 62.

In the Figure B.6, we show the frequency of the most repeated words in the publications of Pinterest users. Clearly, we see *love* as the most repeated word. We can observe many words such

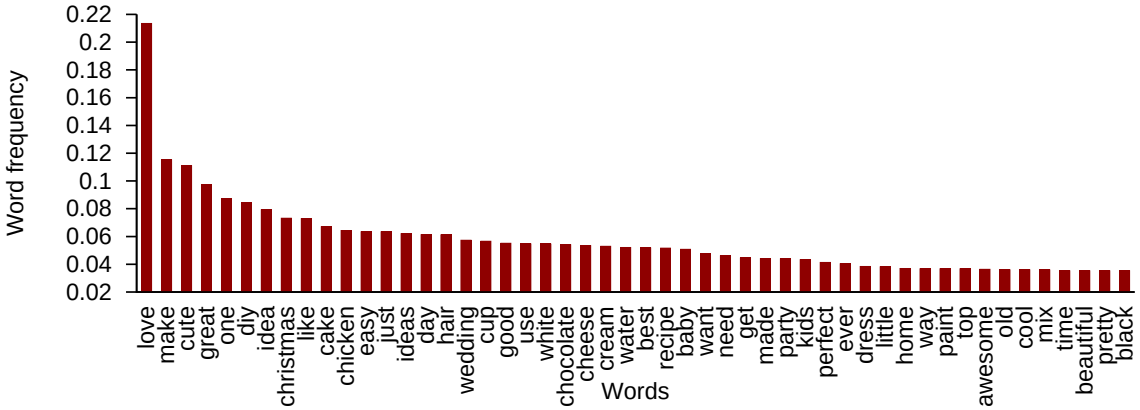


Figure B.6: Lexical analysis of pins description.

as *chicken*, *wedding*, *party*, *baby*, *chocolate*, *recipe* that describes the publications. Words such as *cake*, *chocolate*, *cream*, *cheese*, *recipe* points the importance of cooking in the social network. Words such as *make*, *idea*, *recipe* suggest tutorials and propositions to do. Other words such as *baby*, *wedding*, *party*, *kids*, *dress* signals the appearance of fashion-related subjects. The lexical analysis gives an insight that Pinterest a social network dedicated to leisure and entertainment while other social networks focuses on social relationships (Facebook) or news (Twitter).

B.4 Related Work

There is a vast research literature on online social networks such as Facebook, Twitter and LinkedIn [103, 72, 110]. The interest for users behavior in social networks has increased in the last years. [72] characterize the behavior of users in Orkut, MySpace, Hi5 and LikedIn and focus on distributional properties of OSN sessions. [103] takes a similar approach to characterize users sessions and users sessions within Facebook, Hi5, LinkedIn and StudiVZ. [104] focus on Spotify and they found daily patterns of usage and difference of behavior between desktop and mobiles users. [110] studies behavior of trends in Twitter with interesting insights into the impact in terms of audience and time. [105] suggests that users with similar topics of interests are more likely to be connected and they prove it in a Last.fm data set. Subsequently, the concern for users publications has grown. [106] gives an insight of users publications in an adult media distribution website and [107] study the publication of short videos in Weibo -the chinese social network.

In the case of Pinterest, recent publications highlight the participation of women in the OSN. As women make up 79% of Pinterest users, [102] study behavior of women in contrast to men with regards to repinning characteristics and the language used. [101] focus on genres as well but describing the differences of interests. [108] works on content curation: why Pinterest users tend to organize the content and the way they chosen to do it. In their work, they claim that websites with highly repinned images tend not to have a high PageRank which is in contradiction with our results of Section B.3.2. The description of their experiment is not completely exposed (in particular their rank of pins) and they served of Google Search API which is currently deprecated and it does not provide any longer the values of PageRank.

B.5 Conclusion

In this annex, we unveil the publications of the emerging Pinterest social network. We crawled the Pinterest website for several months and collected relevant information to characterize it. We showed that 84% of the activities are retransmissions and Pinterest is a *retransmission network*. Moreover, we showed that retransmissions lead to the increase of links toward a particular website. We also analyze the images published in Pinterest and study their filesize, popularity and perform a classification into categories. Finally, we performed a lexical analysis on the description of images. We derive from all our results that Pinterest is a social network devoted to leisure and entertainment in comparison with other social networks that target mostly social relationships (Facebook) and breaking news (Twitter).

Our findings describe users behavior in Pinterest and we expected to use them to feed SONETOR, the social network traffic generator. Although this study has bring many insights about a social network, we were only capable of describing publications. We were not able to extract information about the requests for the images and publications. In consequence, we were not able to use this social traffic as input for our experiments.

Glossary

This Glossary is intended to assist readers of this thesis. It includes a summary of the acronyms and technical definitions used across the document.

Anycast : Network addressing and routing methodology which routes datagrams towards the topologically nearest node, identified by the same destination address
AS : Autonomous System

Cache Hit : Metric used to asses cache efficiency in a network of caches.
Caching Replacement Policy : See Replacement Policy
Caching Strategy : Management policy that decide what and where information should be stored
CCN : Content Centric Network
CDF : Cumulative Distribution Function
CDN : Content Delivery Network
COMET : COntent Mediator architecture for content-aware nETworks
CRP : Cache replacement policy

DHT : Distributed Hash Table
Diversity : Measure to quantify the number of distinct elements stored across a network
DNS : Domain Name System
DONA : Data Oriented Network Architecture
DPI : Deep Packet Inspection

Eigenvector : Measure of the influence of a node in a network

FIX : Caching Strategy with a fix probability acceptance rate
FTP : File Transfer Protocol

Hop Reduction Ratio : Measure of the percentage of all the distances traveled from clients to servers in a network of caches
HRR : Hop Reduction Ratio
HTTP : Hypertext Transfer Protocol
HTTPS : Hypertext Transfer Protocol over Secure Socket Layer

ICN : Information Centric Network
IP : Internet Protocol
ISP : Internet Server Provider

LCD : Leave Copy Down
LCE : Leave Copy Everywhere
LFU : Last Frequently Used
LRU : Last Recently Used

MAGIC : Max-Gain In-Nework Caching Strategy
MFU : Most Frequently Used
MPC : Most Popular Caching Strategy
MRU : Most Recently Used
MZipf : Maldelbrot Zipf. Generalization of the Zipf function

NDN.JS : Javascript framework for CCN
NetInf : Network of Information, ICN architecture proposal
NRS : Name Resolution System

OSN : Online Social Networks
OSPF : Open Shortest Path First
OSPFN : Open Shortest Path First for Content Centric Networks

P2P : Peer to Peer
Pagerank : Measure of the influence of a node in a network. It is a modification of Eigenvector and largely used by Google in its search engine

Percentage of Cached Elements : Relative number of caching operations performed by a node using a certain caching strategy with regards to another caching strategy.

PlanetLab : Global research testbed for the deployment of planetary-scale experiments

pmf : Probability distribution function

Proxy Server : A proxy server process requests by forwarding them to remote servers, it intercepts responses and handles the response to the client

PURSUIT : ICN architecture proposal

RAND : Random replacement policy

REDIS : Open-source, networked, in-memory, key-value data store

Replacement Policy : Defines the structure of the cache and the method to evict elements when space is needed

RV : Rendez-vous node

SACS : Socially-Aware Caching Strategy

SAIL : Scalable & Adaptive Internet Solutions

Server Farm : Collection of computer servers to accomplish server needs far beyond the capabilities of a single computer

SONETOR : Social Network Traffic Generator

SQUID : Web proxy caching applications that supports HTTP, HTTPS and FTP protocols among others.

Stale : See Percentage of Cached Elements

STU : Simulation Time Unit

TM : Topology Manager

Triad : Translating Relaying Internet Architecture integrating Active Directories

TTL : Time To Live

URI : Uniform Resource Identifier

URL : Uniform Resource Locator

VoD : Video on Demand

VoIP : Voice over IP

WWW : World Wide Web

Zipf : Empirical law which states that given some corpus of content, the frequency of any piece of content is inversely proportional to its rank in the frequency table. The most often piece of content will occur approximately twice as often as the second most frequence piece.

Bibliography

1 Bibliography

[1] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 1, no. 40, pp. 35–41, 1977.

[2] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson, "Adaptive web caching: Towards a new global caching architecture," *Computer Networks and ISDN Systems*, vol. 30, no. 22-23, pp. 2169–2177, Nov. 1998. [Online]. Available: http://dx.doi.org/10.1016/S0169-7552(98)00246-3

[3] C. Aggarwal, J. L. Wolf, and P. S. Yu, "Caching on the world wide web," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 11, no. 1, pp. 94–107, Jan. 1999. [Online]. Available: http://dx.doi.org/10.1109/69.755618

[4] P. Cao and S. Irani, "Cost-aware www proxy caching algorithms," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, ser. USITS'97. Berkeley, CA, USA: USENIX Association, 1997, pp. 18–18.

[5] E. Cohen, B. Krishnamurthy, and J. Rexford, "Evaluating server-assisted cache replacement in the web," in *Algorithms—ESA '98*. Springer, 1998, vol. 1461, pp. 307–319.

[6] R. P. Wooster and M. Abrams, "Proxy caching that estimates page load delays," in *Selected Papers from the Sixth International Conference on World Wide Web*. Essex, UK: Elsevier Science Publishers Ltd., 1997, pp. 977–986. [Online]. Available: http://dl.acm.org/citation.cfm?id=283554.283259

[7] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2013-2018," *Cisco White Paper*, no. 7, feb 2014.

[8] Nielsen Media Research, "Nielsen cross-platform report q3," *Nielsen White Paper*, no. 7, feb 2011.

[9] McKinsey & Company, "Internet matters: The net's sweeping impact on growth, jobs, and prosperity," *McKinsey White Paper*, may 2011.

[10] I. Psaras, L. Saino, M. Arumaithurai, K. Ramakrishnan, and G. Pavlou, "Name-based replication priorities in disaster cases," in *IEEE Conference on Computer Communications INFOCOM 2014, NOM Workshop*, ser. NOM'14. IEEE, 2014.

[11] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2011-2016," *Cisco White Paper*, no. 5, feb 2012.

[12] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini, "Multipath congestion control in content-centric networks," in *IEEE Conference on Computer Communications INFOCOM 2013, NOMEN Workshop*.

[13] A. Tanenbaum, *Computer Networks*, 5th ed. Prentice Hall Professional Technical Reference, 2010.

[14] J. Wang, "A survey of web caching schemes for the internet," *SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 36–46, Oct. 1999. [Online]. Available: http://doi.acm.org/10.1145/505696.505701

[15] T. Krenc, O. Hohlfeld, and A. Feldmann, "An internet census taken by an illegal botnet: A qualitative assessment of published measurements," *SIGCOMM Computer Communications Review*, vol. 44, no. 3, pp. 103–111, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656893

[16] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[17] F. Despaux, Y.-Q. Song, and A. Lahmadi, "Combining Analytical and Simulation Approaches for Estimating End-to-End Delay in Multi-hop Wireless Networks," in *Proceedings of the 8th Internationl Conference on Distributed Computing in Sensor Systems (DCOSS)*, ser. DCOSS 2012. IEEE, 2012, pp. 317–322.

[18] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '07. New York, NY, USA: ACM, 2007, pp. 181–192. [Online]. Available: http://doi.acm.org/10.1145/1282380.1282402

[19] D. R. Cheriton and M. Gritter, "Triad: A new next-generation internet architecture," http://www-dsg.stanford.edu/triad/, Tech. Rep., July 2000.

[20] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1–26, 2013. [Online]. Available: http://dx.doi.org/10.1109/surv.2013.070813.00063

[21] C. Dannewitz, "NetInf: An Information-Centric design for the future internet," 2009.

[22] C. Dannewitz, E. Bauer, M. Becker, F. Beister, N. Dertmann, M. Kionka, M. Mohr, F. Steffen, S. Stey, and S. Weber, "OpenNetInf documentation design and implementation," University of Paderborn Technical Report TR-RI-10-318, Tech. Rep., Jun. 2010.

[23] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing Information Networking Further: From PSIRP to PURSUIT," Oct. 2010.

[24] Deliverable, COMET, http://www.comet-project.org/deliverable, "D3. 2: Final Specification of Mechanisms, Protocols and Algorithms for the Content Mediation System," *November 15th*, 2011.

[25] B. Ohlman, D. Corujo, G. Boggia, G. Tyson, E. Davies, A. Molinaro, and S. Eum, "Information-centric Networking: Baseline Scenarios. Internet

Draft," draft-irtf-icnrg-scenarios-03, Internet Engineering Task Force, Feb. 2015. [Online]. Available: https://tools.ietf.org/html/draft-irtf-icnrg-scenarios-03

[26] H. Song, N. Zong, Y. Yang, and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement," RFC 6646 (Informational), Internet Engineering Task Force, Jul. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6646.txt

[27] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396 (Draft Standard), Internet Engineering Task Force, Aug. 1998, obsoleted by RFC 3986, updated by RFC 2732. [Online]. Available: http://www.ietf.org/rfc/rfc2396.txt

[28] D. Wessels and K. Claffy, "Internet Cache Protocol (ICP), version 2," RFC 2186 (Informational), Internet Engineering Task Force, Sep. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2186.txt

[29] ——, "Internet Cache Protocol (ICP), version 2," RFC 2187 (Informational), Internet Engineering Task Force, Sep. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2187.txt

[30] L. Wang, A. M. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF based routing protocol for Named Data Networking," NDN Consortium, Tech. Rep. NDN-0003, 2012.

[31] J. Moy, "OSPF Version 2," RFC 2328 (Internet Standard), Internet Engineering Task Force, Apr. 1998, updated by RFCs 5709, 6549, 6845, 6860. [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt

[32] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "MDHT: A Hierarchical Name Resolution Service for Information-centric Networks," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 7–12. [Online]. Available: http://doi.acm.org/10.1145/2018584.2018587

[33] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN Consortium, Technical Report NDN-0005, October 2012. [Online]. Available: http://named-data.net/techreports.html

[34] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a caching simulator for information centric networking (icn)," in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS '14. ICST, Brussels, Belgium, Belgium: ICST, 2014.

[35] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in content-oriented architectures," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2018584.2018586

[36] V. Cate, "Alex – a global filesystem," in *IN PROCEEDINGS OF THE 1992 USENIX FILE SYSTEM WORKSHOP*, 1992, pp. 1–12.

[37] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "ACT: audio conference tool over named data networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 68–73. [Online]. Available: http://dx.doi.org/10.1145/2018584.2018601

[38] H. Wu, J. Li, T. Pan, and B. Liu, "A novel caching scheme for the backbone of named data networking," in *In proceedings of IEEE International Conference on Communications (ICC) 2013*, 2013, pp. 3634–3638.

[39] P. Faratin, D. Clark, P. Gilmore, S. Bauer, A. Berger, and W. Lehr, "Complexity of internet interconnections: Technology, incentives and implications for policy," in *in Proceedings of 35th Research Conference on Communication, Information and Internet Policy (TPRC)*, 2007.

[40] V. Pacifici and G. Dán, "Stable content-peering of autonomous systems in a content-centric network," in *In Proceedings of 9th Swedish National Computer Networking Workshop (SNCNW)*, 2013.

[41] W. Yang, D. Trossen, and J. Tapolcai, "Scalable forwarding for information-centric networks," in *IEEE International Conference on Communications (ICC) 2013*, June 2013, pp. 3639–3644.

[42] H. Xie, Y. Wang, and G. Wang, "Scale content centric networks via reactive routing," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013, pp. 3530–3535.

[43] W. Wong, L. Wang, and J. Kangasharju, "Neighborhood search and admission control in cooperative caching networks," in *In Proceedings of IEEE Global Communications Conference (GLOBECOM), 2012*, Dec 2012, pp. 2852–2858.

[44] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, Aug. 2010. [Online]. Available: http://doi.acm.org/10.1145/1842733.1842736

[45] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *Proceedings of the Re-Architecting the Internet Workshop*, ser. ReARCH '10. New York, NY, USA: ACM, 2010, pp. 5:1–5:6. [Online]. Available: http://doi.acm.org/10.1145/1921233.1921240

[46] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 44–49. [Online]. Available: http://doi.acm.org/10.1145/2018584.2018596

[47] A. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks„" Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia., Feb. 2007. [Online]. Available: http://www.gridbus.org/reports/CDN-Taxonomy.pdf

[48] M. J. Freedman, "Experiences with coralcdn: A five-year operational view," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 7–7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855711.1855718

[49] J. Wang, C. Peng, C. Li, E. Osterweil, R. Wakikawa, P.-c. Cheng, and L. Zhang, "Implementing instant messaging using named data," in *Proceedings of the Sixth Asian Internet Engineering Conference*, ser. AINTEC '10. New York, NY, USA: ACM, 2010, pp. 40–47. [Online]. Available: http://doi.acm.org/10.1145/1930286.1930292

[50] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, "Voccn: Voice-over content-centric networks," in *Proceedings of the 2009 Workshop on Re-architecting the Internet*, ser. ReArch '09. New York, NY, USA: ACM, 2009, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/1658978.1658980

[51] W. Shang, J. Thompson, M. Cherkaoui, J. Burke, and L. Zhang, "NDN.JS: A javascript client library for Named Data Networking," in *IEEE INFOCOMM 2013 NOMEN Workshop*, April 2013.

[52] C. Tschudin and M. Sifalakis, "Named Functions and Cached Computations," in *In Proceedings of IEEE Consumer Communications and Networking Conference 2013, special session on Research and Case Study for Designing and Deploying Information-centric Networks*, 2014.

[53] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 55–60. [Online]. Available: http://doi.acm.org/10.1145/2342488.2342501

[54] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks (extended version)," *Computer Communications*, vol. 36, no. 7, pp. 758–770, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2013.01.007

[55] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," in *21st International Conference on Computer Communications and Networks (ICCCN)*, July 2012, pp. 1–7.

[56] B. Mathieu, P. Truong, W. You, and J. Peltier, "Information-centric networking: a natural design for social network applications," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 44–51, July 2012.

[57] E. J. Rosensweig, D. S. Menasché, and J. Kurose, "On the steady-state of cache networks." in *IEEE Conference on Computer Communications INFOCOM 2013*. IEEE, 2013, pp. 863–871.

[58] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *IEEE Conference on Computer Communications INFOCOM 2012 NOMEN Workshop*. IEEE, Mar. 2012, pp. 280–285. [Online]. Available: http://dx.doi.org/10.1109/infcomw.2012.6193506

[59] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," *Computing Research Repository*, vol. abs/1202.0108, 2012. [Online]. Available: http://arxiv.org/abs/1202.0108

[60] S. K. 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," in *Proceedings of the ACM SIGCOMM 2013 Conference*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 147–158. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486023

[61] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *Proceedings of the 3rd ACM SIGCOMM Workshop on*

Information-centric Networking, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 27–32. [Online]. Available: http://doi.acm.org/10.1145/2491224.2491232

[62] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep. 1, 2011.

[63] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, Dec. 2003. [Online]. Available: http://doi.acm.org/10.1145/954339.954341

[64] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnsim: An highly scalable ccn simulator." in *In proceedings of IEEE International Conference on Communications (ICC) 2013*. IEEE, 2013, pp. 2309–2314.

[65] A. S. Tanenbaum, *Modern Operating Systems*, 3rd ed. NJ, USA: Prentice Hall Press, 2007.

[66] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, March 2012, pp. 268–273.

[67] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-as cooperative caching for content-centric networks," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 61–66. [Online]. Available: http://doi.acm.org/10.1145/2491224.2491234

[68] G. Rossini and D. Rossi, "A dive into the caching performance of content centric networking." in *The International IEEE Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks*. IEEE, 2012, pp. 105–109.

[69] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks." in *In Proceedings of IEEE Conference on Computer Communications INFOCOM 2012, NOMEN Workshop*. IEEE, 2012, pp. 316–321.

[70] J. Ren, W. Qi, C. Westphal, J. W. Kejie Lu, S. Liu, and S. Wang, "Magic: a distributed max-gain in-network caching strategy in information-centric networks," in *In Proceedings of IEEE Conference on Computer Communications INFOCOM 2014, NOM Workshop*, April 2014.

[71] L. A. Adamic and B. A. Huberman, "Zipf's law and the Internet," *Glottometrics*, vol. 3, no. 5, 2002.

[72] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in *ACM SIGCOMM*, ser. IMC '09. NYC, USA: ACM, 2009, pp. 49–62. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644900

[73] L. Gyarmati and T. Trinh, "Measuring user behavior in online social networks," *Network Magazine of Global Internetworking*, vol. 24, no. 5, pp. 26–31, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1109/MNET.2010.5578915

[74] Watts, Duncan J. and Dodds, Peter S., "Influentials, Networks, and Public Opinion Formation," *Journal of Consumer Research*, vol. 34, no. 4, pp. 441–458, 2007.

[75] Song, Kaisong and Wang, Daling and Feng, Shi and Yu, Ge, "Detecting opinion leader dynamically in chinese news comments," in *Proceedings of the 2011*

international conference on Web-Age Information Management, ser. WAIM'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 197–209.

[76] A. Langville and C. Meyer, "A Survey of Eigenvector Methods for Web Information Retrieval," *SIAM Rev.*, vol. 47, no. 1, pp. 135–161, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1137/S0036144503424786

[77] M. Franceschet, "Pagerank: standing on the shoulders of giants," *Communications of the ACM*, vol. 54, no. 6, pp. 92–101, Jun. 2011. [Online]. Available: http://doi.acm.org/10.1145/1953122.1953146

[78] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th International Conference on World Wide Web*, ser. WWW '03. New York, NY, USA: ACM, 2003, pp. 640–651. [Online]. Available: http://doi.acm.org/10.1145/775152.775242

[79] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte carlo methods in pagerank computation: When one iteration is sufficient," *SIAM Journal on Numerical Analysis*, vol. 57, no. 2, pp. 890–904, 2007.

[80] S. Acharyya and J. Ghosh, "Outlink estimation for pagerank computation under missing data," in *WWW 2004*, ser. WWW Alt. '04. New York, NY, USA: ACM, 2004.

[81] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation*. Springer, 2008.

[82] J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," in *In Proceedings of Neural Information Processing Systems 2012*, 2012, pp. 548–556.

[83] I. Cantador, P. Brusilovsky, and T. Kuflik, "Hetrec," in *In Proceedings of ACM Recommender Systems 2011*, ser. RecSys 2011. NYC, USA: ACM, 2011.

[84] J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," in *In Proceedings of Neural Information Processing Systems 2012*, 2012.

[85] C. Bernardini, T. Silverston, and O. Festor, "SONETOR: a Social Network Traffic Generator," in *In proceedings of IEEE International Conference on Communications (ICC) 2014*, pp. 3374–3380.

[86] ——, "A Pin is Worth a Thousand Words: Characterization of Publications in Pinterest," in *In proceedings of IEEE International Wireless Communications & Mobile Computing Conference (IWCMC) 2014 - TRaffic Analysis and Characterization (TRAC)*, pp. 3374–3380.

[87] K. Pentikousis, B. Ohlman, D. Corujo, G. Boggia, G. Tyson, E. Davies, A. Molinaro, and S. Eum, "Information-centric Networking: Baseline Scenarios. Internet Draft," draft-irtf-icnrg-scenarios-02, Internet Engineering Task Force, Jun. 2013. [Online]. Available: https://tools.ietf.org/html/draft-irtf-icnrg-scenarios-02

[88] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of ccn-caching trees," in *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I*, ser. Networking 2011. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 78–91. [Online]. Available: http://dl.acm.org/citation.cfm?id=2008780.2008789

[89] M. Granovetter, "Threshold Models of Collective Behavior," *American Journal of Sociology*, vol. 83, no. 6, pp. 1420–1443, 1978. [Online]. Available: http://dx.doi.org/10.2307/2778111

[90] M. Bates, "Models of natural language understanding," *Proceedings of the National Academy of Sciences*, vol. 92, no. 22, pp. 9977–9982, 1995.

[91] S. ULC, "Sandvine global internet phenomena report-1h2012," 2012.

[92] A. Baid and D. Raychaudhuri, "Wireless access considerations for the mobilityfirst future internet architecture," in *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, May 2012, pp. 1–5.

[93] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013, "Information Centric Networking". [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613002235

[94] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[95] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, and A. Mauthe, "A survey of mobility in information-centric networks: Challenges and research directions," in *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, ser. NoM '12. New York, NY, USA: ACM, 2012, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2248361.2248363

[96] P. Mockapetris, *RFC 1035 Domain Names - Implementation and Specification*, Internet Engineering Task Force, November 1987. [Online]. Available: http://tools.ietf.org/html/rfc1035

[97] K. Katsaros, G. Xylomenos, and G. C. Polyzos, "MultiCache: An Overlay Architecture for Information-centric Networking," *Computer Networks*, vol. 55, no. 4, pp. 936–947, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2010.12.012

[98] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," University of Michigan, Tech. Rep. CSE-TR-433-00, 2002.

[99] H. Tuomisto, "A consistent terminology for quantifying species diversity? yes, it does exist," *Oecologia*, vol. 164, no. 4, pp. 853–860, 2010.

[100] M. Tortelli, D. Rossi, G. Boggia, and L. Grieco, "Ccn simulators: Analysis and cross-comparison," in *In ACM Conference on Information-Centric Networking, ICN-2014, Demo Session.*, Paris, France, Sep. 2014.

[101] E. Gilbert, S. Bakhshi, S. Chang, and L. Terveen, "I need to try this?: a statistical overview of pinterest," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2013, pp. 2427–2436.

[102] R. Ottoni, J. P. Pesce, D. B. L. Casas, G. F. Jr., W. M. Jr., P. Kumaraguru, and V. Almeida, "Ladies first: Analyzing gender roles and behaviors in pinterest." in *In proceedings of International Wireless Communications & Computing Conference 2013*, 2013.

[103] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding online social network usage from a network perspective." in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, 2009.

[104] B. Zhang, G. Kreitz, M. Isaksson, J. Ubillos, G. Urdaneta, J. A. Pouwelse, and D. Epema, "Understanding user behavior in spotify," in *IEEE Conference on Computer Communications INFOCOM 2013*. IEEE, 2013, pp. 220–224. [Online]. Available: http://dblp.uni-trier.de/db/conf/infocom/infocom2013.html#ZhangKIUUPE13

[105] R. Schifanella, A. Barrat, C. Cattuto, B. Markines, and F. Menczer, "Folks in folksonomies: Social link prediction from shared metadata," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, ser. WSDM '10. New York, NY, USA: ACM, 2010, pp. 271–280. [Online]. Available: http://doi.acm.org/10.1145/1718487.1718521

[106] G. Tyson, Y. Elkhatib, N. Sastry, and S. Uhlig, "Demystifying porn 2.0: a look into a major adult video streaming website," in *Proceedings of the 13th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '13. New York, NY, USA: ACM, pp. 417–426. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504739

[107] Z. Guo, J. Huang, J. He, X. Hei, and D. Wu, in *Proceedings of the 14th International Conference on Passive and Active Measurement*, ser. PAM'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 166–175. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36516-4_17

[108] C. Zhong, S. Shah, K. Sundaravadivelan, and N. Sastry, "Sharing the loves: Understanding the how and why of online content curation." in *In proceedings of International Wireless Communications & Computing Conference 2013*, 2013.

[109] N. Litvak and K. Avrachenkov, "The Effect of New Links on Google PageRank," INRIA, Tech. Rep., 2004.

[110] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 591–600. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772751

2 Webography

[111] Blackadder FP-7 PURSUIT Prototype. `https://github.com/fp7-pursuit/blackadder`. Accessed: 2014-12-16.

[112] Blackhawk FP-7 PURSUIT Prototype. `http://users.piuha.net/blackhawk/0.3/`. Accessed: 2014-12-16.

[113] CCNx project. `http://www.ccnx.org`. Accessed: 2014-12-16.

[114] CCNx project documentation. `https://www.ccnx.org/releases/latest/doc/technical/ContentObject.html`. Accessed: 2014-12-16.

[115] Content-Centric Networking Packet Level Simulator. `http://www-sop.inria.fr/members/Frederic.Urbani/ns3dceccnx/`. Accessed: 2014-12-16.

[116] FP-7 CONVERGENCE Project. `http://www.ict-convergence.eu`. Accessed: 2014-12-16.

[117] ICN Simulator. `http://sourceforge.net/projects/icnsim/`. Accessed: 2014-12-16.

[118] Mobility First Prototype. `http://mobilityfirst.winlab.rutgers.edu/Prototype.html`. Accessed: 2014-12-16.

[119] NS3 DCE CCNx. `https://code.google.com/p/ccnpl-sim/`. Accessed: 2014-12-16.

[120] NSF Mobility First project. `http://mobilityfirst.winlab.rutgers.edu/`. Accessed: 2014-12-16.

[121] PlanetLab Platform. `http://www.planetlab.org`. Accessed: 2014-12-16.

[122] Qwilt, transparent caching and video delivery platform. `http://www.qwilt.com`. Accessed: 2014-04-18.

[123] Redis, key-value cache and store server. `http://www.redis.io`. Accessed: 2014-12-16.

[124] SAIL NetInf. `http://www.netinf.org`. Accessed: 2014-12-16.

[125] SQUID: Optimizing Web Delivery. `http://www.squid-cache.org`. Accessed: 2014-04-18.

[126] Bussiness week article. http://goo.gl/0qutg. Accessed: 2014-04-18.

[127] Health leaders media article. http://goo.gl/ljbwd. Accessed: 2014-04-18.

[128] http://mashable.com/2012/11/06/obama-wins-twitter. Accessed: 2014-04-18.

[129] https://github.com/jlhutch/pylru. last seen: 10-07-2014.

[130] SocialCCNSim website. `https://github.com/mesarpe/socialccnsim`.

[131] Sonetor website. `https://github.com/mesarpe/sonetor`.

[132] http://techcrunch.com/2012/02/07/pinterest-monthly-uniques/.

[133] Accessed: 2014-04-18.

[134] http://toolbar.netcraft.com/stats/topsites. Accessed: 2014-04-18.

[135] http://www.alexa.com/siteinfo/pinterest.com.

[136] http://www.businessinsider.com/pinterest-raises-225-million-2013-10.

[137] http://www.internetlivestats.com/internet-users/. Accessed: 2014-04-18.

[138] http://www.reuters.com/article/2013/02/21/net-us-funding-pinterest-idusbre91k01r20130221.

[139] http://www.robots.ox.ac.uk/ vgg/share/practical-image-classification.htm.

Résumé

Content Centric Networking (CCN) est une architecture pour l'Internet du futur. CCN inclut des fonctionnalités de cache dans tous les noeuds du réseau. Son efficacité dépend largement de la performance de ses stratégies de cache. C'est pour cela que plusieurs études proposent des nouvelles stratégies de cache pour améliorer la performance d'un réseau CCN. Cependant parmi toutes ces stratégies, ce n'est pas évident de décider laquelle fonctionne le mieux. Il manque un environnement commun pour comparer ces stratégies. De plus, il n'est pas certain que ces approches soient les meilleures alternatives pour améliorer la performance du réseau.

Dans cette thèse, on vise le problème de choisir les meilleures stratégies de caches pour CCN et les contributions sont les suivantes. On construit un environnement commun d' évaluation dans lequel on compare via simulation les stratégies de caches disponibles: Leave Copy Everywhere (LCE), Leave Copy Down (LCD), ProbCache, Cache "Less" For More et MAGIC. On analyse la performance de toutes ces stratégies et on décide la meilleure stratègie de cache pour chaque scénario. Ensuite, on propose deux stratégies de cache basées sur la popularité pour CCN. On commence avec un étude de la popularité de contenu et on présent la stratégie Most Popular Caching (MPC). MPC privilèges la distribution de contenu populaire dans les caches afin d' ameliorer les autres stratégies de cache. Dans une deuxième étape, on présent une stratègie de cache basé dans l'information des réseaux sociaux: Socially-Aware Caching Strategy (SACS). SACS privilèges la distribution de contenu publié par les utilisateurs les plus importantes.

Mots-clés: ccn, caching, strategy, icn, popularity, social networks.

Abstract

Content Centric Networking (CCN) is a new architecture for a future Internet. CCN includes in-network caching capabilities at every node. Its efficiency depends drastically on performances of caching strategies. A lot of studies proposing new caching strategies to improve the performances of CCN. However, among all these strategies, it is still unclear which one performs better as there is a lack of common environment to compare these strategies.

In this thesis, we address the challenge of selecting the best caching strategies for CCN. The contribution of this thesis are the following. We build a common evaluation scenario and we compare via simulation the state of the art caching strategies: Leave Copy Everywhere (LCE), Leave Copy Down (LCD), ProbCache, Cache "Less" For More and MAGIC. We analyze the performance of all the strategies in terms of Cache Hit, Stretch, Diversity and Complexity, and determine the cache strategy that fits the best with every scenario. Later on, we propose two novel caching strategies for CCN based on popularity. First, we study popularity of content and we present Most Popular Caching (MPC) strategy. MPC privileges distribution of popular caches into the caches and thus, it overcomes other caching strategies. Second, we present an alternative caching strategy based on social networks: Socially-Aware Caching Strategy (SACS). SACS privileges distribution of content published by influential users into the network. Both caching strategies overcome state of the art mechanisms and, to the best of our knowledge, we are the first to use social information to build caching strategies.

Keywords: ccn, caching, strategy, icn, popularity, social networks.