La gestion du trafic dans les réseaux orientés contenus

Nada Sbihi Benkirane

Thèse de Doctorat
Université Pierre et Marie Curie – Paris 6

Spécialité

Systèmes Informatiques

présentée par

Mme. Nada Sbihi

pour obtenir le grade de

Docteur de l'université Pierre et Marie Curie – Paris 6

Gestion du trafic dans les réseaux orientés contenus

Jury

James ROBERTS	**Directeur de thèse**	**Chercheur, INRIA et SystemX**
Serge FDIDA	**Directeur de thèse**	**Professeur, UPMC Sorbonne universités – Paris 6**
Leila SAIDANE	**Rapporteur**	**Professeur, ENSI–Tunisie**
Walid DABBOUS	**Rapporteur**	**Chercheur, INRIA Sophia Antipolis**
Sebastien TIXEUIL	**Examinateur**	**Professeur, UPMC Sorbonne universités – Paris 6**
Diego PERINO	**Examinateur**	**Chercheur, Bell Labs–Alcatel-Lucent**
Gwendal SIMON	**Examinateur**	**Maître de conférence, Telecom Bretagne**

Remerciements

Je tiens tout particulièrement à remercier mon encadrant de thèse M. Jim Roberts de m'avoir donné la chance de poursuivre mes études. Son suivi et ses précieux conseils m'ont apportés beaucoup d'aide durant ma thèse. Son experience est une source inépuisable dont j'ai pu apprendre grâce à son encadrement et sa générosité.

Je remercie également M.Serge Fdida d'avoir accepté de diriger ma thèse ainsi que Mme. Leila Saidane et Mr. Walid Dabbous mes rapporteurs, pour leurs remarques et suggestions sur la version préliminaire de ce document.

Je suis reconnaissante à M.Sébastien Tixeuil, M. Diego Perino et M. Gwendal Simon pour avoir accepté de m'honorer de leur participation au jury de cette thèse.

Je tiens à remercier M. Philippe Robert de m'avoir acceuilli au sein de l'équipe Réseaux, algorithmes et Probabilités(RAP) à l'INRIA.
Je suis très reconnaissante à Mme. Christine Fricker pour son aide, sa générosité et sa gentillesse. Ton savoir scientifique, ta pédagogie et ta modestie m'ont donné beaucoup d'espoirs.

Je remercie également les anciens et nouveaux membres de l'équipe Rap et particulièrement Virginie Collette.

Un grand merci à mes parents pour leur encouragement et Otmane, mon époux pour son soutien, petit clin d'oeil particulier à mon fils Amine.

Merci à ma famille, amies, et collégues.

Résumé

Les réseaux orientés contenus (CCN) ont été créés afin d'optimiser les ressources réseau et assurer une plus grande sécurité. Le design et l'implémentation de cette architecture est encore à ces débuts. Ce travail de thèse présente des propositions pour la gestion de trafic dans les réseaux du future. Il est nécessaire d'ajouter des mécanismes de contrôle concernant le partage de la bande passante entre flots. Le contrôle de trafic est nécessaire pour assurer un temps de latence faible pour les flux de streaming vidéo ou audio, et pour partager équitablement la bande passante entre flux élastiques. Nous proposons un mécanisme d'Interest Discard pour les réseaux CCN afin d ?optimiser l'utilisation de la bande passante. Les CCN favorisant l'utilisation de plusieurs sources pour télécharger un contenu, nous étudions les performances des Multipaths/ Multisources ; on remarque alors que leurs performances dépendent des performances de caches. Dans la deuxième partie de cette thèse, nous évaluons les performances de caches en utilisant une approximation simple et précise pour les caches LRU. Les performances des caches dépendent fortement de la popularité des objets et de la taille des catalogues. Ainsi, Nous avons évalué les performances des caches en utilisant des popularités et des catalogues représentant les données réelles échangées sur Internet. Aussi, nous avons observé que les tailles de caches doivent être très grandes pour assurer une réduction significative de la bande passante ; ce qui pourrait être contraignant pour l'implémentation des caches dans les routeurs. Nous pensons que la distribution des caches devrait répondre à un compromis bande passante/mémoire ; la distribution adoptée devrait réaliser un coût minimum. Pour ce faire, nous évaluons les différences de coût entre architectures.

Abstract

Content Centric Network (CCN) architecture has been designed to optimize network resources and ensure greater security. The design and the implementation of this architecture are only in its beginning. This work has made some proposals in traffic management related to the internet of the future. We argue that it is necessary to supplement CCN with mechanisms enabling controlled sharing of network bandwidth by competitive flows. Traffic control is necessary to ensure low latency for conversational and streaming flows, and to realize satisfactory bandwidth sharing between elastic flows. These objectives can be realized using "per-flow bandwidth sharing". As the bandwidth sharing algorithms in the IP architecture are not completely satisfactory, we proposed the Interest Discard as a new technique for CCN. We tested some of the mechanisms using CCNx prototype software and simulations. In evaluating the performance of multi-paths we noted the role of cache performance in the choice of selected paths. In the second part, we evaluate the performance of caches using a simple approximation for LRU cache performance that proves highly accurate. As caches performance heavily depends on populations and catalogs sizes, we evaluate their performance using popularity and catalogs representing the current Internet exchanges. Considering alpha values, we observe that the cache size should be very large ; which can be restrictive for caches implementation in routers. We believe that the distribution of caches on an architecture creates an excessive bandwidth consumption. Then, it is important to determine a tradeoff bandwidth/memory to determine how we should size caches and where we should place them ; this amounts to evaluate differences, in cost, between architectures.

Table des matières

Introduction générale **10**

I La gestion du trafic dans les réseaux orientés contenus **16**

1 Introduction **17**
1.1 Problématique 17
1.2 Etat de l'art 18
1.2.1 Contrôle du trafic au coeur du réseau 18
1.2.2 Protocole de transport 19
1.2.3 Multipath et multisource 19

2 Partage de bande passante **21**
2.1 Identification des flots 21
2.2 Caches et files d'attente 22
2.3 Le principe du flow aware networking 23
2.3.1 Partage des ressources dans les CCN 23
2.3.2 Contrôle de surcharge 25

3 Mécanismes pour CCN **26**
3.1 Mécanismes pour les utilisateurs 26
3.1.1 Détection des rejets 26
3.1.2 Protocole de transport 27
3.2 Mécanismes pour opérateurs 28
3.2.1 Motivation économique 28
3.2.2 Interest Discard 28

4 Stratégies d'acheminement **31**
4.1 Multicast 31
4.2 Multisources 32
4.2.1 Protocole de transport Multipath 32

4.2.2 Performance de MPTCP . . . 32
4.2.3 CCN et routage multipath . . . 38
4.2.4 Performances des multipaths . . . 38

5 Simulations et expérimentations 42
5.1 Simulations . . . 42
5.2 Expérimentations . . . 43
5.2.1 Fair sharing . . . 43
5.2.2 Interest discard . . . 44
5.2.3 Scénarios et résultats . . . 44

6 Conclusion 46

II Performances des caches 48

7 Introduction 49
7.1 Problématique . . . 49
7.2 Etat de l'art . . . 49
7.3 Contributions . . . 49

8 Mesure du trafic et performances des caches 51
8.1 Mesure du trafic . . . 51
8.1.1 Types de contenu . . . 51
8.1.2 La taille des contenus et des objets . . . 52
8.1.3 Distribution de la popularité . . . 53
8.2 Le taux de hit d'un cache LRU . . . 54
8.2.1 Independent Reference Model . . . 54
8.2.2 Les modèles analytiques . . . 55
8.2.3 La formule de Che . . . 55
8.3 Autres politiques de remplacement . . . 57
8.3.1 Le cache Random . . . 57
8.3.1.1 Relation entre taux de hit et temps moyen de séjour . . . 57
8.3.1.2 Approximation de Fricker . . . 59
8.3.1.3 Approximation de Gallo . . . 60
8.3.2 Le cache LFU . . . 61
8.3.3 Comparaison des politiques de remplacement . . . 63

9 Les performances des hiérarchies de caches 64
9.1 Caractéristiques d'une hiérarchie de caches . . . 64
9.1.1 Politique de remplacement . . . 64
9.1.2 Les politiques de meta-caching . . . 65
9.1.3 Les politiques de forwarding . . . 67
9.2 Performances des hiérarchies de caches . . . 69
9.2.1 Généralisation de la formule de Che . . . 69

9.2.2 Cas d'application : hiérarchie de caches avec mix de flux 72

10 Conclusion **76**

III Coûts d'une hiérarchie de caches **77**

11 Introduction **78**
11.1 Problématique 78
11.2 Etat de l'art 79

12 Coût d'une hiérarchie de caches à deux niveaux **82**
12.1 Différence des coûts entre structures 83
12.2 Estimation numérique des coûts 84
12.2.1 Coût normalisé 85
12.2.2 Solution optimale en fonction du taux de hit global 87
12.3 Exemple : coûts des torrents 88

13 Coopération de caches **90**
13.1 Load sharing 90
13.2 Caches coopératifs 91
13.3 Routage orienté contenu 95

14 Hiérarchies alternatives **99**
14.1 Coût d'une hiérarchie de caches à plusieurs niveaux 99
14.1.1 Evaluation des coûts 99
14.1.2 Coopération de caches 101
14.2 Coûts et politiques de remplacement 102
14.2.1 Politique LFU 102
14.2.2 Politique Random 105

15 Conclusion **107**

Conclusion générale **108**

Introduction générale

Les réseaux orientés contenus

L'idée de passer d'un réseau à connexion point à point à un réseau orienté contenu date de plus d'une décennie. En effet le projet TRIAD [1] proposait déjà une architecture ICN. Cependant, peu de travaux ont été construits sur la base de ce projet, sans doute parce que les contenus à cette époque n'avaient pas le même poids énorme qu'ils prennent actuellement dans le trafic Internet. Quelques années après, d'autres propositions commencent à éclairer la recherche dans le domaine des réseaux orientés contenu.

DONA, une architecture orientée données a été proposée en 2007 par Koponen et al. [1]. Elle se base sur l'utilisation de noms autocertifiés et incorpore des fonctionnalités de caching. Plus récemment l'architecture CCN a attiré l'attention de la communauté scientifique, alertée par la croissance énorme du trafic de distribution de contenus et le succès des CDNs proposant des services payants aux fournisseurs de contenus. Les CDNs utilisent d'une manière intelligente les chemins menant aux contenus et implantent des caches partout dans le monde afin d'accélérer les téléchargements. Akamai reste le leader mondiale dans ce domaine.

Les opérateurs ne peuvent pas rester passifs dans cet univers des contenus et sont obligés d'envisager une mise à niveau de l'Internet afin de réduire les coûts engendrés par l'augmentation rapide du trafic de contenus, notamment de vidéos. L'architecture CCN vient au bon moment et suscite beaucoup d'intérêt de la communauté scientifique, d'autant plus que celui qui propose cette architecture n'est autre que Van Jacobson très connu pour des contributions marquantes au développement d'IP [2]. Plusieurs autres projets d'architecture orientée contenu ont été proposées dont 4WARD/SAIL [3] et PSIRP/PURSUIT [4].

Ghodsi et al. [5] ont comparé les différentes propositions de réseaux ICN et ont dressé les points de divergences et les points communs entre ces architectures. Ils déplorent aussi le peu de remise en question de l'utilisation des ICN en comparaison avec une solution d'évolution des CDNs.

Les architectures ICN présentent plusieurs points communs de design. Les échanges sur ces architecture sont basés sur le modèle publish/subsribe, le nom de l'objet est publié, un objet est demandé par l'envoi de paquet Interest. Les caches ICN sont utilisés pour tout

[1] http ://gregorio.stanford.edu/triad/

type de données et protocole. Ils sont ouverts à tout utilisateur qui peut ainsi déposer ses propres contenus dans les caches. Tous les noeuds du réseau utilisent un cache localisé au niveau routeur et les contenus sont sécurisés indépendamment de leur emplacement. C'est le serveur d'origine qui signe les contenus, et les utilisateurs peuvent vérifier les contenus grâce à leur signature publique.

Les architectures diffèrent dans certains points. Certains designs proposent la sécurisation des contenus avec un mécanisme d'auto-certification. Dans ce cas, les contenus portent des noms qui n'ont pas de signification évidente pour l'homme (un code à plusieurs caractères par exemple). La sécurité des contenus est assurée par la signature des objets. L'utilisateur peut vérifier l'authenticité de l'objet en validant sa signature au moyen d'une clef publique récupérée grâce aux PKI.

Le routage orienté contenus est aussi un point de divergence entre architectures, certaines architectures préconisent le routage orienté contenus basé sur BGP alors que d'autres proposent leur propre modèle de routage. Dans tous les cas, ce domaine reste encore mal exploré et pose des problèmes sérieux de passage à l'échelle.

CCN

Van Jacobson propose la nouvelle architecture d'Internet orientée contenu nommée CCN (Content-centric networking) [2]. Cette architecture permettrait une recherche directe d'un contenu sans avoir à identifier son détenteur, comme dans le cas des réseaux IP.

En effet, dans les réseaux IP les données sont recherchées par leur localisation et non pas par leur nom. Un utilisateur cherchant une donnée doit avoir une information sur l'adresse IP de la machine contenant cette information pour pouvoir la récupérer. Ce fonctionnement pose plusieurs problèmes de sécurité, de disponibilité et de complexité des processus de récupération de données.

L'architecture CCN a pour objectif de simplifier les processus de recherche. Un utilisateur cherche une donnée à travers son nom. Dés lors que la requête est lancée, une demande sous forme d'un paquet dit "Interest" est envoyée à son routeur d'accès. Si la donnée n'est pas présente dans le content store de ce routeur, la requête se propage au fur et à mesure dans le réseau. Une fois la donnée trouvée, elle suit le chemin inverse de la requête de recherche jusqu'à l'utilisateur final et sera stockée dans un "Content Store" dans les routeurs CCN intermédiaires.

Cette architecture offre plusieurs possibilités de disponibilité indépendamment de l'adresse d'une machine. La sécurité est associée directement aux données et pas aux "conteneurs" (liens, routeurs, serveurs,...) ce qui permet d'ajuster de manière très flexible le niveau de sécurité à la nature du contenu en question. Plus intéressant encore, les contenus ne sont plus associés à des conteneurs précis mais peuvent être dupliqués à volonté et stockés notamment dans des mémoires caches au sein du réseau.

Les contenus sont divisés en "chunks", chaque chunk ayant typiquement la taille d'un paquet IP. CCN respecte le déroulement logique d'une requête : un utilisateur demande une donnée en émettant des paquets de type "Interest" et reçoit en retour des paquets de données de type "Data". A chaque paquet Interest correspond un seul paquet Data et

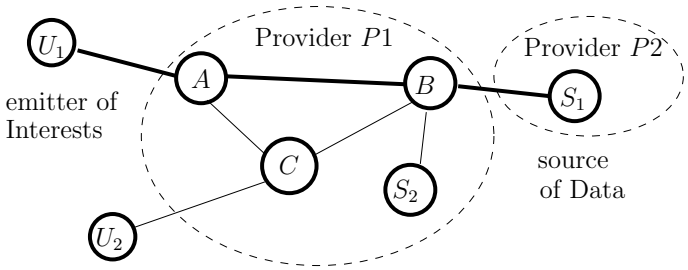


FIGURE 1 – Un segment du réseau CCN reliant un utilisateur U1 à une source S1

chaque paquet Data correspond à un Chunk.

La figure 1 représente un segment d'un réseau CCN. Pour récupérer des données du fournisseur P2, l'usager U1 envoie des paquets "Interest" pour le contenu demandé au travers des routeurs A et B. Supposant que les Content Stores de A et B ne contiennent pas le document demandé, les paquets Data suivent le chemin inverse de S1 vers U1 en passant par B et A.

Pour gérer l'envoi des données, trois types de mémoires sont utilisées au niveau de chaque noeud :

- Content Store : Dans cette base de données sont stockés des objets physiquement.
- Pending Interest Table (PIT) : Dans cette table sont stockés les noms des données et les interfaces demandeuses correspondantes.
- FIB : par analogie avec IP, le FIB dans CCN stocke les préfixes des noms des données ainsi que les prochaines interfaces à emprunter pour arriver à la donnée. Cette table est enrichie à travers les déclarations de détention de données par les noeuds du réseau.

Dans CCN il n'y a nul besoin d'acheminer les adresses source et destination à travers le réseau pour récupérer la donnée. Le format des paquets Interests et Data est explicité à la figure 2.

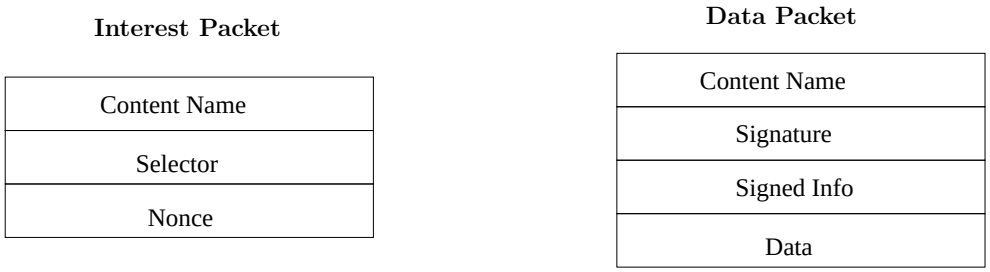


FIGURE 2 – Format des paquets CCN

A la réception d'un Interest par un noeud, ce dernier vérifie si le chunk demandé existe dans son Content Store. Si c'est le cas, le paquet Data sera envoyé à l'interface demandeuse. Sinon le chunk demandé sera recherché dans le PIT. S'il est trouvé, l'interface demandeuse sera rajoutée au PIT. Si les deux bases de données ne fournissent aucune information, on cherchera dans le FIB si une entrée matche avec le chunk recherché. Alors le paquet Interest sera acheminé vers les interfaces conduisantes à la donnée. La table PIT sera mise à jour avec une nouvelle entrée pour le chunk en question.

A la réception d'un paquet Data par un noeud, une recherche est effectuée dans le Content Store. Si une entrée matche, alors le paquet reçu est supprimé, car ceci implique que le chunk est déjà livré à toutes les interfaces demandeuses. Sinon la donnée sera recherchée dans le PIT. Si une entrée matche avec la donnée reçue, elle sera acheminée vers les interfaces demandeuses. Le chunk sera typiquement stocké en même temps dans le Content Store.

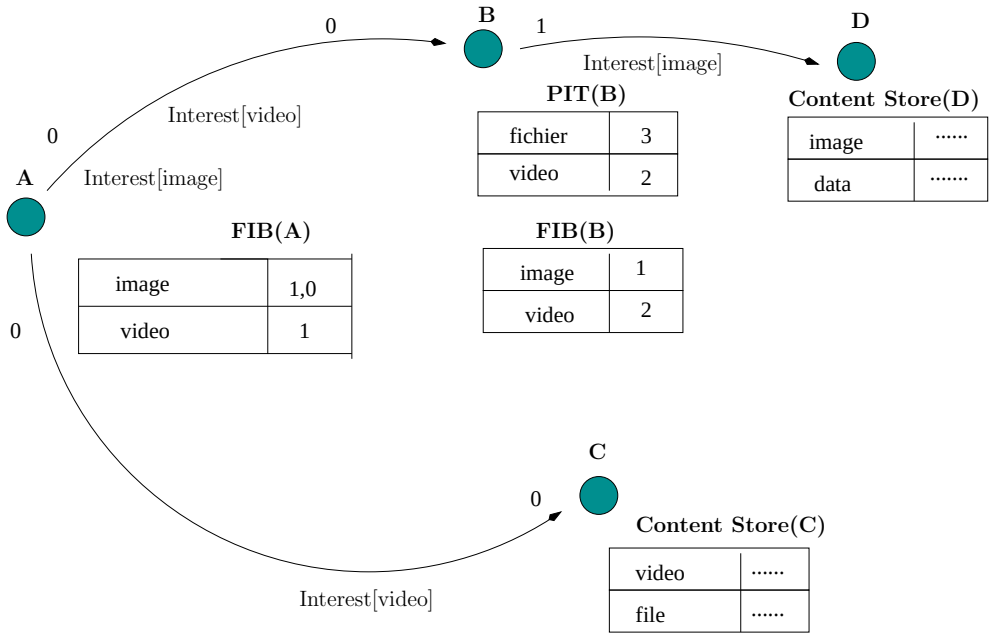


FIGURE 3 – La recherche des données dans CCN

Dans l'exemple de la figure 3, le noeud A cherche les chunks "video" et "image". Le FIB du noeud A indique que les paquets Interests doivent être acheminés vers l'interface 0 et 1 pour l'image, et vers l'interface 1 pour la vidéo. A la réception de l'Interest demandant la vidéo par le noeud B, ce dernier ignore l'Interest reçu car le PIT contient déjà une entrée. Cette entrée est mise à jour. Cependant, quand le noeud B reçoit l'Interest demandant l'image, il l'envoie à l'interface 1 indiquée par le FIB. Le noeud D, par la suite, achemine la donnée vers le noeud A. Cette donnée sera stockée dans tous les Content Stores des noeuds l'ayant reçu ou transité.

Le séquencement des paquets est établi grâce aux noms des chunks. Ces derniers sont organisés d'une façon hiérarchique. Ainsi, pour demander un segment il faut indiquer le

nom hiérarchique du chunk demandé dans le paquet Interest. Le FIB détermine l'interface de sortie adéquate grâce à un algorithme "longest prefixe match".

La gestion du trafic

La gestion du trafic consiste à contrôler le partage de la bande passante des liens afin d'assurer la qualité de service des diverses applications. Le partage équitable de la bande passante entre flots répond aux exigences des flux sensibles à débit faible et protège les flux adaptatifs des flux gourmands. La gestion du trafic dans les réseaux IP a fait l'objet de plusieurs travaux de recherches, mais jusqu'à présent aucune solution entièrement satisfaisante n'a été trouvée.

Jacobson et al. proposent un modèle CCN basé sur un échange Interest/Data : pour recevoir un paquet Data, l'utilisateur devrait envoyer un paquet Interest. Ils assurent que ce mécanisme réalise une bonne gestion de trafic. Ceci est insuffisant en réalité. Tous les utilisateurs n'utilisent pas un protocole de transport unique ; ce dernier peut être modifié par l'utilisateur final. Il est alors nécessaire de définir des mécanismes pour gérer le partage de la bande passante. La détection de rejets par numéros de séquences des paquets n'est plus applicable sur CCN ; l'utilisation des multipaths et multichemins change l'ordre des paquets, et le timeout ne suffit pas pour assurer de bonnes performances du protocole de transport. D'autre part, dans CCN, l'utilisateur ne peut plus utiliser un protocole de transport multipath car il ignore les destinations possibles de ces paquets.

En plus des différences entre CCN et IP en termes de modèle d'échange de données, les CCN implémentent des caches au niveau de chaque routeur. Ainsi, l'utilisation des caches réduit la consommation de bande passante, et les délais de transmission.

Contributions de la thèse

Le rapport est structuré comme suit :

- Gestion du trafic dans les réseaux orientés contenus : Nous proposons des mécanismes pour gérer les flots dans CCN, en identifiant les flots par le nom d'objet, et en adaptant le Fair queuing à CCN. Nous proposons un mécanisme Interest Discard pour protéger les opérateurs des rejets de Data, un nouveau modèle de tarification, une détection rapide des rejets pour contrôler les fenêtres des protocoles de transport, et nous évaluons les performances des Multipaths. Nous évaluons les mécanismes proposés par simulation et expérimentation.

- Performance des caches : Nous évaluons les types des contenus, leur taille, et leur loi de popularité. Nous utilisons la formule de Che comme modèle analytique fiable pour mesurer les taux de hit des caches LRU, nous adaptons cette formule au cas Random,

nous évaluons le taux de hit d'une hiérarchie à deux niveaux, et nous proposons une différentiation de stockage de données pour améliorer la probabilité de hit.

- Coûts des hiérarchies de caches : Nous cherchons un optimum pour un tradeoff bande passante/mémoire, nous effectuons une estimation des couts, nous appliquons les calculs à un cas réel de données de type torrents et nous comparons les performances d'une hiérarchie coopérative et non-coopérative.

Le travail de thèse a contribué à la rédaction des articles suivants :

- S. Oueslati, J. Roberts, and N. Sbihi, Flow-Aware traffic control for a content-centric network, in Proc of IEEE Infocom 2012.

- C. Fricker, P. Robert, J. Roberts, and N. Sbihi, Impact of traffic mix on caching performance in a content-centric network, in IEEE NOMEN 2012, Workshop on Emerging Design Choices in Name-Oriented Networking

- J. Roberts, N. Sbihi, Exploring the Memory-Bandwidth Tradeoff in an Information-Centric Network, International Teletraffic Congress, ITC 25, Shanghai, 2013.

Première partie

La gestion du trafic dans les réseaux orientés contenus

Chapitre 1

Introduction

1.1 Problématique

Jacobson et al [2] remettent en cause l'architecture TCP/IP en constatant qu'il est nécessaire de mettre en place une nouvelle architecture Internet répondant mieux à la croissance exponentielle de la demande pour des contenus de tous types. Conçu au début pour des échanges simples, l'Internet devient en effet le moyen incontournable pour consulter les sites du Web, échanger vidéos et audios, et partager des fichiers. Son utilisation ayant évolué, son architecture devrait suivre.

Plusieurs projets sont consacrés à la conception du réseau du futur. La proposition CCN de Jacobson et al. [2] est une des plus crédibles et a fait l'object de nombreuses études. Nous avons constaté cependant que la définition de cette nouvelle architecture reste incomplète, notamment en ce qui concerne la gestion du trafic. L'objet de la présente partie du rapport est de décrire nos études sur cette question.

Le contrôle de trafic est essentiel afin d'assurer aux flots audios et vidéos des délais faibles même quand ils partagent la bande passante avec des flots de données à haut débit. Il importe également d'empêcher d'éventuels "flots gourmands" d'altérer la qualité des autres flots en s'accaparant une part excessive de la bande passante des liens qu'ils partagent.

Nous pouvons diviser les axes de recherches concernant la gestion du trafic en 3 volets :

- *Le partage de bande passante.* Dans CCN comment faut il partager la bande passante entre flots et comment identifier un flot quand les adresses IP ne sont plus utilisées dans les paquets ? Comment exploiter la particularité des réseaux CCN, dite de "flow balance", où les paquets Data suivent exactement le chemin inverse des paquets Interest.

- *Le protocole de transport.* La conception d'un protocole de transport sous CCN est plus compliqué qu'en IP. En effet, nous ne pouvons plus prendre en compte la succession des numéros de séquence des paquets comme garantie de non congestion, car, sous CCN même sans congestion les paquets ne se suivent pas forcément. Un flot ne

se dirige pas vers une destination particulière et unique. Les paquets Data peuvent venir de n'importe quelle source de données, y compris des caches, et peuvent suivre des chemins différents.

- *Multipath et multisource.* Sous CCN le téléchargement des objets de sources multiples devient une opportunité intéressante pour augmenter les débits. Cette multiplicité a l'avantage de pouvoir améliorer les débits de téléchargment. Cependant, il y a également un inconvénient car, les paquets empruntant plusieurs chemins différents, l'opportunité d'exploiter les caches devient plus faible.

1.2 Etat de l'art

Lorsque nous avons fait les études présentées dans cette partie, à partir du Chapitre 2, il n'y avait pas de publications qui abordaient les questions mentionnées ci-dessus. Cet état de fait a changé depuis et dans la présente section nous évoquons quelques articles dont les propositions peuvent mettre en cause nos choix.

1.2.1 Contrôle du trafic au coeur du réseau

Nous proposons, dans le chapitre 2, la mise en oeuvre dans les files d'attente des routeurs d'un ordonnancement de type Deficit Round Robin (DRR) afin d'assurer l'équité du partage de bande passante. Nous associons à cet ordonnancement un mécanisme "Interest discard" de rejet sélectif de paquets Interest dans la direction inverse. D'autres ont proposé d'assurer l'équité du partage en ordonnançant plutôt les flots de paquets Interest dans un mécanisme dit "Interest shaping". Dans ce cas, les paquets Data sont acheminés dans une simple file FIFO.

Dans [6], Fdida et al. décrivent un mécanisme d'Interest shaping implémenté au niveau de chaque routeur CCN. La file de transmission des paquets Data est observée et un seuil de remplissage est fixé. Le débit des paquets Interest est régulé suivant le débit des paquets Data et en tenant compte de la différence entre la taille de la file d'attente et le seuil. Ainsi les paquets Interest peuvent être retardés si le nombre de paquets Data stockés en file d'attente dépassent le seuil.

Dans l'article [7], Gallo et al. proposent un mécanisme d'Interest shaping presque identique à la nôtre sauf que les paquets Interest subissent un retard d'envoi en cas de congestion afin de limiter leur débit. Des files virtuelles accueillent les paquets Interest selon l'objet recherché et un compteur est attribué à chaque file.

Les mécanismes d'Interest shaping peuvent provoquer des pertes de débits dans certains scénarios. Dans l'exemple de la figure 1.1, si on applique l'Interest shaping sur tout le trafic on va retarder les paquets Interest indépendamment de leur appartenance aux flots. On va alors transmettre plus de paquets Interest du flot2 que du flot1. Sachant que le flot 2 sur le lien suivant perd en bande passante puisque la capacité du lien est inférieur à la capacité du lien emprunté par le flot 1, cette situation particulière entraine une perte en capacité.

Il nous semble que notre choix d'agir directement sur le flux de données en imposant l'équité par ordonnancement DRR est plus robuste. l'Interest discard n'intervient alors

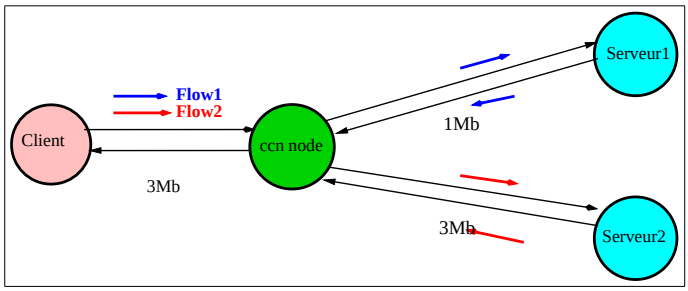


FIGURE 1.1 – Exemple illustrant la perte en débit en utilisant le shaping sans classification par flot

que comme mécanisme de contrôle secondaire permettant le routeur de ne pas inutilement demander des paquets Data qui ne peuvent pas être acheminés à cause de la congestion sur la voie descendante. Il est notable aussi que le mécanisme d'Interest discard s'intègre naturellement avec l'ordonnancement DRR dans la même "line card".

1.2.2 Protocole de transport

Dans notre proposition, le protocole de transport n'est pas critique car le contrôle d'équité du partage est réalisé par ordonnancement DRR. D'autres ont proposé plutôt de maintenir de simples files FIFO en se fiant au protocole de transport pour le contrôle d'équité.

Dans IP deux moyens permettent de détecter un rejet de paquet : les numéros de séquences des paquets s'ils ne se suivent pas ou le timeout si le délai d'acquittement dépasse un seuil. Dans CCN les paquets Data, même s'ils ne subissent aucun rejet, n'ont pas forcément le bon ordre puisque les paquets peuvent être reçus de plusieurs sources différentes. D'autre part le timeout calculé sur la base du RTT change tout le temps et ceci pourrait conduire à des timeouts inopinés.

Gallo et al. [8] définissent un protocole de transport dénommé ICP (Interest Control Protocol). Ils proposent la détection de rejet par timeout mais en mettant à jour régulièrement le seuil de temporisation selon un historique. CCN dans ses premières versions proposait un timeout fixe ce qui sanctionnait lourdement les flots en nécessitant une attente excessive avant de détecter les rejets. Vu que les RTTs peuvent varier dans CCN non seulement à cause de la congestion, mais, du fait qu'on peut récupérer les objets de plusieurs sources, cette détection par timeout semble difficile à mettre en oeuvre.

Dans notre approche, détaillée plus loin, nous proposons un mécanisme de notification explicite de congestion qui permettrait une détection rapide de congestion sans besoin de timeout.

1.2.3 Multipath et multisource

Dans CCN, un utilisateur ne peut pas connaitre d'avance les chemins empruntés par les paquets, car les paquets Interest ne contiennent que le nom de l'objet et le numéro du

paquet demandé, sans aucune précision sur la destination. La difficulté alors de détecter ou de séparer les chemins rend l'utilisation des protocoles de type MPTCP (multipath TCP) quasiment impossible. De ce fait, on ne peut pas utiliser une fenêtre par chemin car l'utilisateur n'a aucun moyen d'acheminer ses paquets en suivant un chemin précis. Ce sont les routeurs qui choisissent le chemin d'un paquet Interest.

Gallo et al. [7] proposent un couplage entre protocole de transport multipath et un mécanisme de forwarding au niveau des routeurs. Ils adaptent le protocole ICP au cas multipath en collectant les RTT venant de différents chemins. En effet, un paquet Interest ne peut pas "savoir" à l'avance vers quel chemin il serait acheminé, mais, une fois acheminé par le réseau, un paquet Data "connait" surement le chemin qu'il a emprunté. Le récepteur peut donc identifier et enregistrer les chemins empruntés par les paquets. Ceci permet notamment d'estimer le RTT de chaque chemin.

Cette approche nous paraît lourde à mettre en oeuvre. Dans notre proposition, on envisage l'utilisation de multipaths mais l'imposition d'un ordonnancement DRR par flot rend inefficace les mécanismes de coopération à la MPTCP. Nous envisageons donc d'autres mécanismes pour éviter les écueils d'un contrôle de congestion indépendant par chemin. Cependant, nous notons également que le gain de débit dû à l'utilisation simultanée de plusieurs chemins n'est réalisable que par des flots ne subissant pas de limitation au niveau de l'accès. Cette observation nous mène à croire qu'un seul chemin, judicieusement choisi, peut largement suffire dans la grande majorité des cas.

Par ailleurs, Rossini et al. [9] identifie un phénomène de "pollution" des caches causé par l'utilisation des multipaths. Leurs résultats montrent des taux de hit nettement inférieurs en cas d'utilisation de chemins multiples, dû à l'éparpillement des copies de paquets dans les différents caches.

Chapitre 2

Partage de bande passante

L'utilisation des caches au sein des CCN contribue à l'amélioration des délais de téléchargement et réduit les problèmes de congestion. Mais la demande en trafic est en augmentation permanente, et il reste nécessaire d'utiliser des mécanismes de gestion du trafic.

En effet, le contrôle de congestion est nécessaire pour assurer des délais négligeables pour les flots voix et vidéos, et pour éviter une consommation abusive de la bande passante par des flots agressifs. Des travaux antérieurs sur le réseau IP suggèrent que le partage de bande passante par flot imposé par un ordonnancement dans les routeurs est une solution efficace. Il réalise en effet une différentiation de service implicite et assure de bonnes performances. Les flots dans les CCN peuvent être identifiés selon l'objet recherché et il est possible ainsi d'appliquer une politique de congestion orientée flot.

Pour gérer le trafic et éviter la congestion, nous adoptons donc une politique "flow-aware" où les actions de gestion tiennent compte de l'appartenance des paquets à des flots. Nous pensons que ceci est préférable à un contrôle de congestion où la performance dépend d'un protocole mis en oeuvre dans l'équipement des utilisateurs, comme dans le réseau IP actuel. En effet, rien ne garantit l'utilisation fidèle de ce protocole. De plus, plusieurs variantes de TCP qui voient le jour le rendant de plus en plus agressif vis à vis des versions antérieures.

2.1 Identification des flots

Dans IP, un flot est défini par les adresses IP et les numéros de port ; cela correspond typiquement à une requête précise et spécifique. Dans un CCN, au niveau de l'interface réseau, on ne peut acheminer qu'une seule demande pour le même objet. Un CCN utilise le multicast en envoyant l'objet reçu à toutes les interfaces indiquées dans la PIT. Nous identifions un flot grâce aux noms de l'objet recherché et le fait que les paquets sont observés au même point du réseau et sont rapprochés dans le temps.

La figure 2.1 montre le format de l'entête des paquets en CCN. Les paquets Data et Interest portent le même nom d'objet mais, actuellement, il n'y a pas moyen de parser

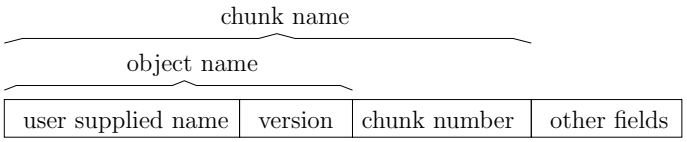


FIGURE 2.1 – Format des paquets CCN

ce nom car c'est le 'chunk name' qui identifie un paquet data. Le champ 'object name' pourrait être analysé afin d'identifier d'une façon unique les paquets correspondants à la demande d'objet mais ceci nécessiterait une modification du code CCNx.

2.2 Caches et files d'attente

La déclaration faite dans l'article de Van Jacobson que "LRU remplace FIFO" semble être inexacte. Une file d'attente ne peut jouer le rôle d'un cache, et vice versa, puisque la file d'attente devrait être de taille faible pour permettre un accès rapide à la mémoire, alors qu'un cache devrait être de taille grande même si le temps d'accès est plus lent. On démontre ceci par des exemples.

On considère une politique d'ordonnancement idéal (LFU) où les objets les plus populaires sont stockés dans un cache de taille N. L'implémentation d'une telle politique est possible selon [10]. On considère des applications générant la majorité du trafic Internet comme YouTube et BitTorrent. La popularité des objets de ces applications suit la loi Zipf de paramètre α, c'est à dire que la popularité du $i^{ème}$ objet le plus populaire est proportionnelle à $i^{-\alpha}$. Des observations rapportées dans les articles [11] et [12] placent ce paramètre à 0.75 environ. On considère que les objets ont tous la même taille et que le catalogue est de M objets.

La probabilité de hit global h pour une politique LFU peut être exprimée par :

$$h = \frac{\sum_{i=1}^{N} i^{-\alpha}}{\sum_{i=1}^{M} i^{-\alpha}}.$$

Si on considère un catalogue et une taille de cache assez grands, on a $h \approx \frac{N}{M}^{1-\alpha}$.

Pour 100 millions vidéos Youtube de taille 4MB [13] il faut 640 GB de mémoire pour un taux de hit de 20 % ou de 25 TB de mémoire pour un taux de hit de 50 %. Pour 400 000 torrents avec une taille moyenne de 7 GB, il faut 4 TB pour un taux de hit de 20 %, ou de 175 TB de mémoire pour un taux de hit de 50 %.

Par ailleurs, la taille d'une file d'attente est au maximum égale au produit de la bande passante par le délai RTT [14] ; pour un lien à 10 Gb/s et un délai de 200 ms, la taille d'une file d'attente est donc de l'ordre de 300 MB.

Il est donc clairement nécessaire de distinguer deux mémoires différentes : des mémoires cache de grande taille pour le stockage des données, et une file d'attente de taille beaucoup moins importante et avec un accès rapide.

Le contrôle du trafic s'appuie sur la gestion des files d'attente. Il est donc important également pour cette raison de bien reconnaître la distinction entre ces files et le Content Store.

2.3 Le principe du flow aware networking

Afin de protéger les flots sensibles audio et vidéo, des techniques pour la gestion de la QoS comme Diffserv et Intserv ont été proposées. Le marquage de paquets dans Diffserv est une technique qui a été implémentée pour prioriser les flots voix ou vidéo. Cependant, c'est une technique qui reste complexe à mettre en oeuvre et sujette à une possibilité de modification illicite du marquage. Il est bien connu par ailleurs que les approches orientées flots d'Intserv sont trop complexes et ne passent pas à l'échelle.

On pourrait éventuellement protéger les flots sensibles et assurer une certaine équité dans le partage des ressources en utilisant un protocole de transport comme TCP. Cependant, l'utilisateur garde dans tous les cas une possibilité de modifier les implémentations ou bien d'utiliser des versions agressives de TCP.

Nous pensons que le flow aware networking, déja proposé pour les réseaux IP peut être adapté aux réseaux CCN. Ceci consiste à mettre en place deux fonctionnalités :

1. le partage équitable de bande passante par flot,
2. le contrôle de surcharge.

Nous développons ces deux points ci-dessous.

2.3.1 Partage des ressources dans les CCN

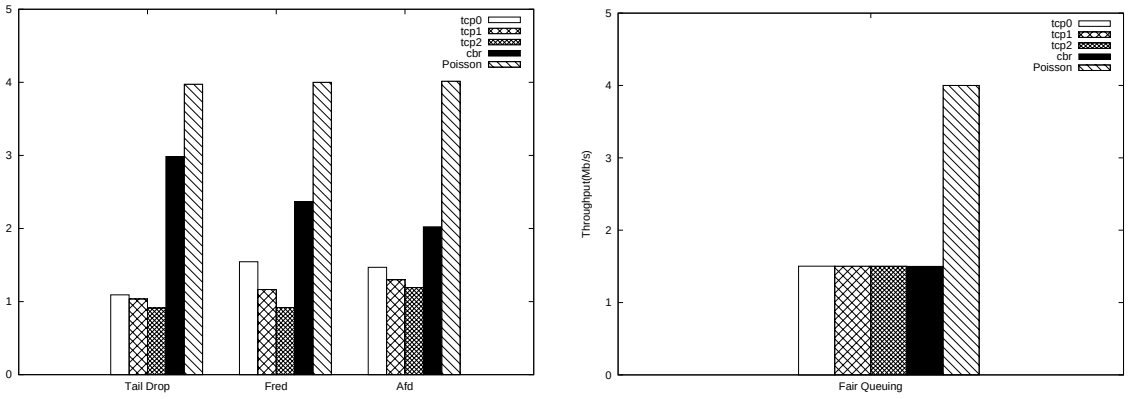
Le partage équitable de la bande passante offre plusieurs avantages très connus (voir [15], [16], [17] et [18]). Nous citons à titre d'exemple :

- les flots émettant des paquets à un débit inférieur au débit équitable ne subissent pas de rejets de paquets.
- les flot sensibles (conversationnels et streaming) ayant généralement des débits faibles sont protégés et bénéficient d'un service de différentiation implicite,
- les flots agressifs ne gagnent rien en émettant des paquets trop rapidement car leur débit ne peut dépasser le débit équitable.

Pour une utilisation équitable de la bande passante, et pour protéger les flot sensibles, nous proposons que le réseau CCN assure lui-même le partage équitable des ressources réseau. Les avantages de cette technique ont été largement développés dans les articles précités ; elle peut s'adapter parfaitement aux réseaux CCN.

Pour le partage équitable des ressources, nous avons revu plusieurs propositions. Nous comparons par simulations certains protocoles fair drop, et un protocole fair queuing. On considère un lien à 10 Mb/s partagé par les flux suivants :

- TCP1 : facteur AIMD=1/2, RTT=10ms
- TCP2 : facteur AIMD=1/2, RTT=30ms
- TCP3 : facteur AIMD=1/2, RTT=50ms
- CBR : flux à débit constant de 3Mb/s
- Poisson : Un flux poissonien de paquets représentant la superposition d'un grand nombre de flux de faible débit. La figure 2.2 présente les résultats obtenus.



(a) Algorithmes Fair drop (b) Fair queuing

FIGURE 2.2 – Comparaison des protocoles fair drop et du protocole Fair queuing

Il est clair que le fair queuing est l'algorithme le plus efficace. Il a également l'avantage d'être sans paramètre. Son passage à l'échelle a été démontré à condition d'assurer également un contrôle de surcharge.

En effet, il a été démontré dans [19] que le nombre de flots ne dépasse pas quelques centaines si la charge du réseau reste inférieur à 90 %, une valeur de charge au-dessus des exigences des opérateurs. Le débit équitable pour un lien de capacité C et de charge ρ est estimé à $C(1-\rho)$ [20]. Dans un réseau à charge normale (ne dépassant pas 90 %) ce débit est largement suffisant pour les flots streaming et conversationnels.

Le partage équitable n'est pas un objectif en soi, mais un moyen d'assurer un débit acceptable et de protéger les flots adaptatifs des flots gourmands. Ainsi, tout flot ne dépassant pas le débit équitable ne subit aucun rejet de paquet dont le délai est très faible. C'est aussi un moyen automatique pour assurer un fonctionnement normal sur Internet sans se soucier des comportements de l'utilisateur final.

Notons que la possibilité de contourner l'imposition d'un partage équitable par la génération de multiples flots au lieu d'un seul est très limitée [21]. Très peu d'usagers peuvent en fait émettre du trafic à un débit plus fort que le débit équitable. Le débit pour la plupart des usagers est limité par leur débit d'accès. De plus, dans un réseau

CCN où les flots sont définis par le nom de l'objet, la possibilité de démultiplier les flots est beaucoup plus limitée qu'en IP.

Nous proposons l'utilisation d'un algorithme d'ordonnancement comme DRR [22] où les files par flot sont modélisées par des listes chainées utilisant une structure nommée ActiveList. Il a été démontré que le nombre de flots dans ActiveList est limité à quelques centaines pour une charge ne pas dépassant 90%, indépendamment de la capacité du lien C [19]. Ces résultats démontrent le "scalabilité" de l'ordonnancement DRR.

2.3.2 Contrôle de surcharge

La demande en trafic est le produit du débit d'arrivée des flots par la taille moyenne d'un flot. On dit qu'un réseau est en surcharge si la demande dépasse la capacité du lien. Dans ce cas, le réseau est instable : le nombre de flots en cours croît et leur débit devient très faible.

Le partage équitable de la bande passante par flot n'est donc scalable que si la charge du lien (demande divisée par la capacité du lien) est normale. On considère la valeur maximale d'une charge normale égale à 90%. Si la charge du réseau dépasse 90%, le nombre de flots devient trop grand et donc lourd à gérer.

Pour contrôler la charge, on pourrait mettre en place un mécanisme de contrôle d'admission. Ceci consiste à éliminer tout nouveau flot dès que la charge du réseau atteint une valeur maximale. Pour ceci, il faut sauvegarder une liste de flots en cours et écarter tout nouveau flot arrivant. Cependant, pour un réseau bien dimensionné, le problème de surcharge ne se pose qu'en certains cas rares, comme par exemple une panne sur un lien réseau.

Pour CCN, plutôt qu'un contrôle d'admission complexe et peu utilisé, nous proposons la simple suppression des paquets d'une certaine liste de flots afin de réduire le niveau de charge. Cette liste pourrait être définie de manière arbitraire (par une fonction hash sur l'identité du flot) ou, si possible, de manière plus sélective en n'incluant que les flots les moins prioritaires.

Chapitre 3

Mécanismes pour CCN

3.1 Mécanismes pour les utilisateurs

L'usager dans un CCN doit participer à la gestion du trafic en modulant la vitesse à laquelle il envoie les Interests pour les chunks d'un même objet. L'utilisation du protocole de transport adaptatif dans un réseau CCN équipé de l'ordonnancement DRR n'est pas nécessaire pour assurer le bon fonctionnement du réseau. Cependant, nous recommandons l'utilisation d'un protocole adaptatif pour éviter les réémissions multiples, dues aux rejets Interest.

3.1.1 Détection des rejets

Nous proposons pour l'utilisateur un mécanisme de détection rapide de rejet dont voici la description. Si un paquet est rejeté au niveau de la file d'attente en raison de sa saturation ou en raison d'une politique de gestion de la file d'attente, on envoie à l'utilisateur l'entête du paquet data sans le payload. L'utilisateur, en recevant un paquet data de payload nul, sait que le paquet a été rejeté, et donc réajuste la vitesse d'émission des Interests pour éviter l'accumulation de réémissions inutiles. Mieux encore, une modification peut être apportée à l'entête du paquet data afin de préciser qu'il correspond à un discard.

Cette technique de détection de rejet est particulièrement intéressante dans les CCN, puisqu'on ne peut pas se baser sur le séquencement des paquets. En effet, contrairement à TCP/IP, deux paquets qui se suivent à l'origine ne se suivent pas forcément à la réception, car la source d'une donnée n'est pas forcément unique, et le chemin n'est pas forcément le même pour tous les paquets.

Actuellement, dans l'implémentation CCNx, la détection est faite uniquement en se basant sur le timeout. Ceci reste très imprécis, et peut poser des problèmes. A titre

d'exemple, avec un timeout fixe, comme c'est le cas de CCNx dans ses premières versions, une détection de rejet ne correspond pas forcément en fait à un rejet, mais à un temps de propagation et de transmission dépassant le timeout. Réduisant la fenêtre dans un tel cas ne fait que détériorer les débits des flots malgré la disponibilité de la capacité dans les liens.

Un protocole de transport efficace arrive à détecter rapidement les rejets. Ceci devient plus difficile lorsque le protocole de transport connecte l'utilisateur à deux sources ou plus. Dans CCN, un utilisateur ne peut savoir à l'avance s'il est servi par deux sources, car la seule donnée qu'il détient est le nom d'objet. De plus, toute l'architecture du réseau et la localisation des serveurs de données sont invisibles pour lui, ce qui est le but de CCN. L'article de Carofiglio et al. [8] apporte une réponse à ce problème. En utilisant un historique un peu large, on collecte des informations statistiques sur le nombre de chemins utilisés ainsi que les RTT recensés pendant les échanges, et on construit, sur la base de ces informations, un protocole de transport multipath efficace.

En tous les cas, nous sommes peu favorables à l'utilisation des multipaths et multi-sources. Les liens réseaux ont des capacités tellement grandes que le débit d'un seul flot ne peut guère atteindre la capacité du lien. Le débit des flots est limité en fait par le débit d'accès au réseau qui est faible par rapport au débit des liens au coeur de réseau. Donc, un seul lien non surchargé est largement suffisant pour q'un flot réalise son débit maximal. Par contre, nous sommes favorables à des mécanismes de load balancing en cas de surcharge de certains liens, ce qui devrait être assez exceptionnel dans un réseau bien dimensionné.

Avec la détection rapide des rejets, le paquet Interest qui a subi le discard est transformé en paquet data sans payload, avec éventuellement une modification légère de l'entête afin de signaler le discard. Le paquet traverse le chemin inverse en supprimant au fur et à mesure les entrées correspondantes à la PIT. A la réception du paquet data, l'utilisateur ou les utilisateurs corrigeront le problème détecté en adaptant au mieux le débit d'émission selon le protocole de transport.

3.1.2 Protocole de transport

Dans le cas d'un ordonnancement fair queuing, l'utilisateur ne gagne pas en bande passante en étant très agressif. Les réémissions multiples d'Interest sont une conséquence directe d'un protocole de transport agressif ne prenant pas en compte le feedback réseau. Si le fair sharing est imposé, comme nous l'avons suggéré, le protocole de transport n'est plus vu comme un outil pour réaliser le partage de bande passante.

Le plus simple serait d'envoyer des paquets Interest à débit constant, mais dans ce cas, le récepteur devrait gérer une large liste de paquets Interest en attente.

Nous proposons plutôt un protocole AIMD (additive increase/multiplicative-decrease) comme TCP avec détection rapide de perte et en utilisant une fenêtre adaptative CWND (Congestion Window). Le nombre de paquets Interest d'un flot qui transite

dans le réseau ne doit pas dépasser la taille de la fenêtre CWND. A la détection d'un rejet par détection rapide ou timeout, la fenêtre est réduite suivant un facteur ; plus ce facteur est proche de 1, plus le protocole est aggressif. En absence de perte, la fenêtre CWND croît linéairement selon un certain taux. Encore, plus ce taux est grand, plus le protocole est aggressif.

3.2 Mécanismes pour opérateurs

En plus de l'ordonnancement "fair queuing", nous envisageons un mécanisme préventif de rejet d'Interest. L'opérateur devrait également exploiter les sources multiples de certaines données en appliquant une stratégie d'acheminement adaptée.

3.2.1 Motivation économique

Il est important de mettre en place une motivation économique pour encourager l'opérateur à déployer cette nouvelle architecture. Il est également important que le fournisseur de réseau soit rémunéré pour le trafic qu'il écoule.

On considère que le fournisseur devrait être payé pour les data envoyés ; par exemple, dans la figure 3.1, l'utilisateur U1 paye le fournisseur P1 pour les data qu'il fournit, et P1 paye P2 pour les data reçus. Cette approche fournit bien la motivation nécessaire pour déployer un réseau CCN muni de caches, car le fournisseur, en utilisant des caches, ne serait pas amené à acheter le même contenu plusieurs fois.

Les frais devraient couvrir les coûts d'infrastructure (bande passante et caches), et leur nature exacte pourrait prendre de nombreuses formes, y compris les tarifs forfaitaires et des accords de peering.

3.2.2 Interest Discard

L'utilisateur ne paye le fournisseur que s'il reçoit effectivement la donnée. Le fournisseur doit donc assurer un contrôle de congestion afin d'éviter la perte des données transmises à ses clients. Il a intérêt à rejeter les Interests en excès pour éviter de racheter des données qui ne peuvent pas être revendues à cause d'une congestion éventuelle. Afin d'éviter un tel problème, nous proposons un mécanisme complémentaire pour protéger le fournisseur.

Supposons le lien AB dans la figure 3.1 congestionné ; B va limiter le débit des Interests envoyés vers le fournisseur P2 pour éviter d'acheter des données qui ne seront pas revendues à l'utilisateur final U1 puisqu'elles seront perdues à cause de la congestion. Nous appelons ce mécanisme "Interest Discard".

On considère le lien AB de la figure 3.1. A reçoit les paquets Interest de U1 et renvoie ces paquets vers B. B reçoit dans l'autre sens les paquets de données récupérés du fournisseur P2 et applique le Deficit Round Robin sur les paquets Data envoyés vers A.

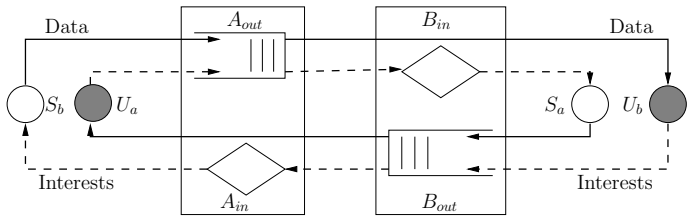


FIGURE 3.1 – Les cartes réseaux des routeurs A et B traversées par des paquets Interest et Data

B calcule en effet un débit d'équité correspondant à l'inverse du temp de cycle moyen de DRR. Le débit des Interest est limité par des rejets forcés en utilisant un sceau à jetons. Le débit des Interests est limité au débit correspondant au débit équitable réalisé actuellement par le DRR (tenant compte de la taille différente des paquets Interest et Data). Le sceau à jetons est donc alimenté au rythme de l'ordonnancement. Notons que le DRR et le sceau à jetons seront réalisés dans la même carte réseau facilitant ainsi leur couplage. Nous présentons le pseudocode interest Discard, cet algorithme doit être utilisé au niveau de chaque interface réseau. Il est exécuté à l'arrivée d'un interest à l'interface, et à chaque cycle Round Robin incrémenter tous les compteurs de l'interface. Nous résumons ci-dessous l'algorithme exécuté au niveau de l'interface réseau.

Algorithm 1 A l'arrivée d'un paquet interest à l'interface réseau

```
Récupérer le nom d'objet du paquet name
Calculer le hash du nom d'objet hash
if file[hash]∄ then
   Créer la file ayant comme ID hash
   Attribuer un compteur count[hash] au flux
   Initialiser le compteur count[hash] = b
end if
if count(hash)==0 then
   rejeter l'interest i
else
   count(hash)- - ;
end if
```

Nous adoptons le DRR comme algorithme FAIR QUEUING, il utilise un nombre fixe de files. On note M le nombre maximal de files Round Robin

Algorithm 2 A la sortie de l'interface réseau

$i = 0$
while $i < M$ **do**
  if $file[i]\exists$ **then**
    Servir le premier paquet de $file[i]$
  end if
  if $file[i] = \varnothing$ et $count[i] = b$ **then**
    Supprimer la file physique $file[i]$
  end if
  for $i = 0$ **to** $M - 1$ **do**
    $count[i] = count[i] + 1$
  end for
end while

Chapitre 4

Stratégies d'acheminement

L'architecture CCN offre de nouvelles possibilités d'acheminement. Il est en particulier possible d'utiliser plusieurs sources pour un même objet. Nous avons fait quelques études sur l'acheminement multi-sources. On démontre que les mécanismes de gestion du trafic fonctionnent bien dans ce contexte. Cependant, avant de poursuivre la recherche dans ce domaine, il est essentiel de comprendre les possibilités réelles qu'offrirait un réseau CCN en fonction de sa politique de stockage. Nos études dans cette direction sont décrites dans les deux parties suivantes de ce rapport.

4.1 Multicast

Sur la figure 3.1 les utilisateurs U_a et U_b demandent le même objet stocké au niveau du fournisseur S1. Si les demandes se passent en même temps, une seule demande sera enregistrée au niveau du routeur B et donc le flot correspondant à l'objet demandé aura le même débit des flots unicast dans le lien BS_1. Il n'y a donc pas lieu de distinguer ces flots dans l'ordonnanceur DRR.

Si les demandes sont décalées dans le temps et que l'utilisateur U_a commence le téléchargement des paquets avant l'utilisateur U_b, il est possible que le débit accordé au flot soit divisé par deux au niveau du lien BS_1. Mais puisque le routeur B est doté d'une mémoire cache, il est très probable que l'utilisateur U_b trouve les paquets précédemment téléchargés par U_a au niveau du cache B, et donc seuls les paquets non téléchargés par U_1 seront demandés et traverseront le lien S_1B. Le débit des flots multicast est donc égal au débit des flots unicast sur toutes les branches du réseau. Il suffit que le temps de téléchargement d'un objet soit inférieur au temps de stockage de l'objet dans un cache. Encore, il n'y a pas lieu de distinguer ces flots dans l'ordonnanceur DRR.

Il est important alors de maintenir une mémoire sauvegardant les paquets des flots en cours, évitant ainsi la division du débit des flot par deux ou plus au cas où plusieurs flots cherchant le même objet arrivent en même temps sur une interface réseau, et que ces flot soient non synchronisés.

4.2 Multisources

4.2.1 Protocole de transport Multipath

L'utilisation des multipaths n'a pas de sens que si les performances du réseau s'améliorent, et que cette utilisation ne réduit pas le débit des flots et leurs performances par rapport à une utilisation complètement unipath. Il est important de vérifier que l'utilisation du routage multipath ne diminue de manière sensible la région de stabilité. Un réseau stable, est un réseau où les flots sont servis en un temps fini, c'est à dire qu'aucun flot souffre de congestion à cause de la charge sur un lien de son chemin.

Dans un modèle entièrement unipath, un réseau est stable à condition que chaque chemin r constitué d'un ensemble de liens $L(r)$ et traversé par un ensemble de flots S vérifie :

$$\sum_{k \in S} \rho_k < C_l \qquad \forall l \in L(r).$$

Le fair queuing, que nous préconisons d'utiliser au niveau des routeurs, n'est pas seulement bénéfique pour protéger les flot gourmands, et pour réaliser une certaine équité entre les flots, mais c'est aussi un moyen de maximiser la région de stabilité. En effet, les auteurs de [23] ont démontré que le fair queuing offrait une région de stabilité maximale, et donc permettrait une utilisation optimale du réseau.

Dans un régime multipath, les auteurs de [24] ont démontré qu'un réseau est stable sous la condition :

$$\sum_{r \in S} \rho_r < \sum_{l \in L(s)} C_l.$$

4.2.2 Performance de MPTCP

Afin de mieux comprendre l'utilisation du routage multipath dans CCN, nous avons étudié d'abord l'impact sur la performance des différentes versions de MPTCP envisagées actuellement pour le réseau IP. Les protocoles Multipaths sont classés en deux catégories : les protocoles non coordonnés et les protocoles coordonnés. Un protocole de transport non coordonné permet de récupérer un maximum de débit mais cette augmentation de débit pénalise le débit des autres flots car le flot multipath consomme plus de capacité que les autres. L'exemple illustré dans la figure 4.1 montre que l'utilisation du TCP non coordonné peut réduire le débit des flots unicast en offrant plus de débit aux flots multichemins, ce qui est loin de nos objectifs. En

effet, le flot $f1$ partage le lien à 2 Mb/s avec le flot $f2$ malgré la disponibilité d'un autre lien entièrement dédié pour lui et pouvant servir tout le débit requis.

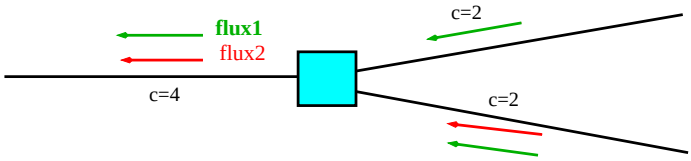


FIGURE 4.1 – Le TCP Multipath non coordonné n'assure pas l'équité

Le MPTCP coordonné répond à la condition de stabilité. Sa description est présenté par l'IETF[1] ; la RFC 6356 décrit les fonctionnalités d'un protocole de transport MPTCP. Il offre beaucoup d'avantages tels que la fiabilité en maintenant la connexion en cas de panne, le load balancing en cas de congestion ou l'utilisation du réseau wifi et ADSL en même temps, par exemple, pour récupérer le même contenu. Chaque flot utilisant MPTCP est subdivisé en plusieurs sous-flots, chacun éventuellement suivant un chemin différent. Chaque sous-flot r détient sa propre fenêtre w_r [25].

On trouve deux types de protocole MPTCP coordonné : le MPTCP semi couplé et le MPTCP couplé. Dans le cas d'un protocole MPTCP couplé, à la détéction d'un rejet, la fenêtre $cwnd_r$ du sous-flot r décroit à $max(cwnd_r - cwnd_{total}/2, 1)$. Dans le cas d'un contrôle de congestion MPTCP semi couplé la fenêtre d'un sous-flot décroit selon sa propre fenêtre de congestion uniquement. En utilisant un MPTCP entièrement couplé, les fenêtres s'annulent à tour de rôle rendant le fonctionnement de ce protocole instable [26].

Dans notre proposition du module de gestion du trafic pour CCN, nous avons proposé l'utilisation du Round Robin, il s'avère que le Round Robin invalide le fonctionnement du MPTCP coordonné. Les ajustements faits par l'utilisateur ne corrigent pas ce problème. Nous avons observé ce phénomène en simulant avec ns2 un tronçon de réseau présenté dans la figure 4.2.

Un flot MPTCP émis par le noeud A est constitué de deux sous-flot : TCP0, qui suit le plus long chemin vers le récepteur MPTCP (noeud D) A-B-C-D, et $TCP2$, qui suit le plus court chemin vers le récepteur MPTCP A-D. Un flot TCP généré par le noeud E suit le plus court chemin E-B-C-F vers le récepteur TCP.

Nous comparons dans un premier temps les débits des flots $TCP0$ et $TCP2$ partageant le lien B-C avec une politique Tail drop dans tous les noeuds du réseau.

La figure 14.3 confirme l'efficacité du MPTCP coordonné qui réalise l'équité en prenant en compte toute la bande passante offerte au flot, et non pas la bande passante offerte par un lien uniquement. Avec le MPTCP non coordonné le flot TCP0 partage

[1] http ://datatracker.ietf.org/wg/mptcp/charter/

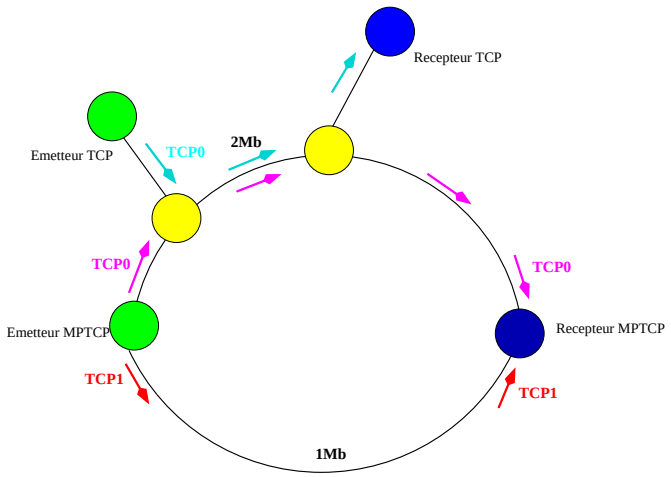
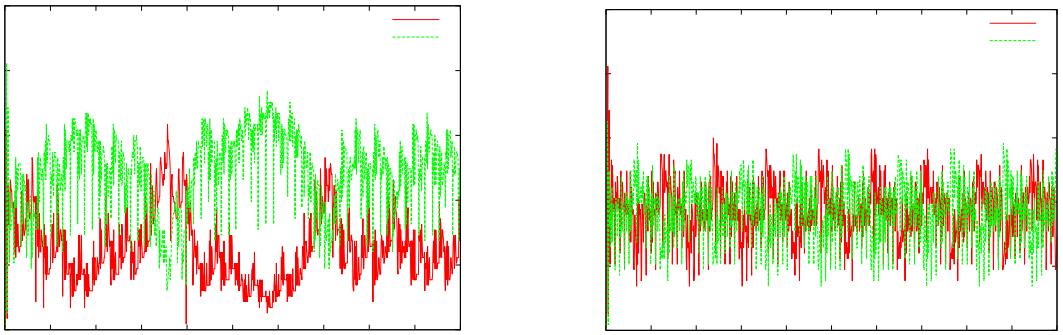


FIGURE 4.2 – un réseau pour illustrer les Multipaths



(a) MPTCP coordonné

(b) MPTCP non coordonné

FIGURE 4.3 – MPTCP coordonné priorise le flot unipath

équitablement la bande passante avec TCP2, ce qui est équitable localement sur le lien, mais ne l'est pas globalement dans le réseau, puisque l'émetteur TCP reçoit un débit supplémentaire du sous-flot TCP1. Le protocole MPTCP coordonné corrige ce problème et offre au flot unipath TCP1 plus de débit que TCP0.

Nous recommençons le même scénario mais en appliquant une politique Round Robin au noeud B. Nous observons le débit des flots TCP0 et TCP2. On obtient les résultats présentés dans la figure 4.4 :

Le MPTCP coordonné avec Round Robin réalise une équité par lien. Les flots TCP0 et TCP2 partage équitablement la bande passante au niveau du lien B-C, contrairement aux résultats observés avec Tail Drop, ou le flot TCP2 gagnait plus de débit. L'effet

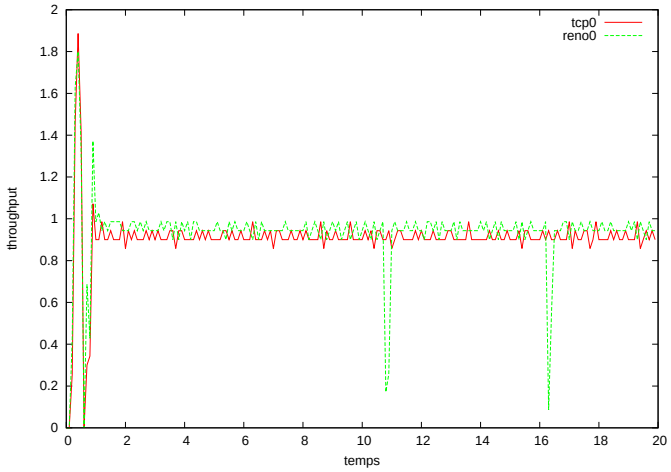


FIGURE 4.4 – Le Round Robin annule l'efficacité du MPTCP

des changements des fenêtres d'une manière coordonnée sont invalidés par le Round Robin. L'effet du partage équitable local par lien du Round Robin est dominant, et le MPTCP coordonné perd son efficacité.

Pour pallier à ce problème ,on propose un nouveau protocole MPRR(Multipath Round Robin) permettant de réaliser l'équité niveau flot et de préserver les performances du protocole MPTCP, le protocole MPTCP est décrit par les algorithme ci dessous :

Algorithm 3 A l'arrivée d'un paquet interest à l'interface réseau I_0 et sortant de l'interface I_1

Récupérer le nom d'objet du paquet *name*
Calculer le hash du nom d'objet *hash*
if $file[hash] \nexists$ **then**
  Créer la file ayant comme ID *hash*
  Attribuer un compteur $mark[hash]$ au flux
  Initialiser le compteur $mark[hash] = 0$
end if
if interest marqué **then**
  mark[hash]++ ;
else
  envoyer le paquet interest à l'interface I_1
  if Une autre interface I_2 réseau est utilisé par le flux name **then**
    envoyé un paquet interest marqué à I_2
  end if
end if

Algorithm 4 A la sortie de l'interface réseau

```
i = 0
while i < M do
  if file[i]∃ then
    if mark[i] == 0 then
      Servir le premier paquet de file[i]
    end if
  else
    mark[i]- - ;
  end if
  if file[i] = ∅ then
    Supprimer la file physique file[i]
  end if
end while
```

Au niveau d'un routeur, si deux interfaces I1 et I2 sont utilisées pour envoyer les paquets d'un flot f, à chaque fois qu'un paquet Interest du flot f est envoyé vers une interface, un paquet Interest dupliqué et marqué serait envoyé à la deuxième interface. Les interfaces servant un flot multipath sont sauvegardées au niveau de la FIB. Au niveau du cache, à chaque fois qu'on reçoit un paquet Interest marqué, il prendrait place dans la file d'attente du flot f et le compteur Interest discard du flot f serait décrémenté. Mais ce paquet ne serait pas acheminé vers les autres interfaces du réseau, il est utilisé uniquement pour mettre en place l'équité entre flots.

Chaque flot a une file unique au niveau de chaque interface. La file correspondante au flot f est traité une seule fois à chaque cycle Round Robin comme tous les flots arrivants à l'interface. Afin de vérifier notre protocole, on utilise le simulateur Simpy pour simuler le réseau de la figure 4.5.

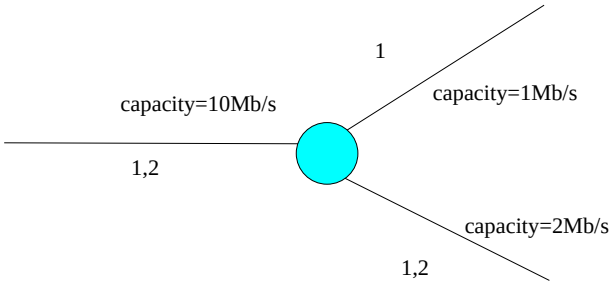
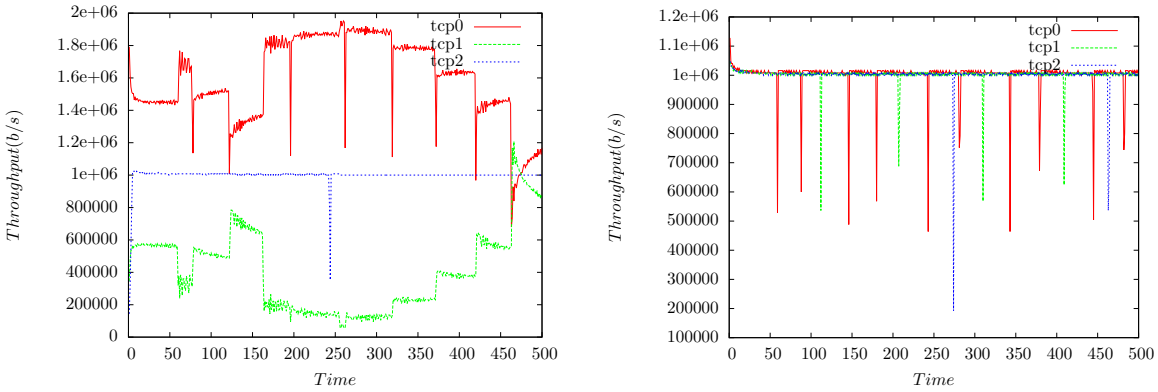


FIGURE 4.5 – réseau simulé avec Simpy

En utilisant le MPRR au niveau des interfaces on obtient les résultats présentés dans la figure 4.6 Le débit du flot TCP0 partageant le lien avec le sous-flot MPTCP TCP1, et le débit du sous flot MPTCP circulant sur le deuxième lien TCP2 en utilisant les politiques Tail drop, Round Robin et MPRR sont représentés dans la figure 4.6.



(a) politique Tail drop

(b) politique Round Robin

FIGURE 4.6 – Débits des flots avec politique Tail drop (gauche) et Round Robin (droite)

Le MPTCP offre des débits approximativement équitables tenant compte de la capacité global offerte au flot. Le Round Robin assure une équité au niveau de chaque lien, ce qui a pour effet de réaliser une inéquité au sens global en prenant compte de la capacité globale offerte.

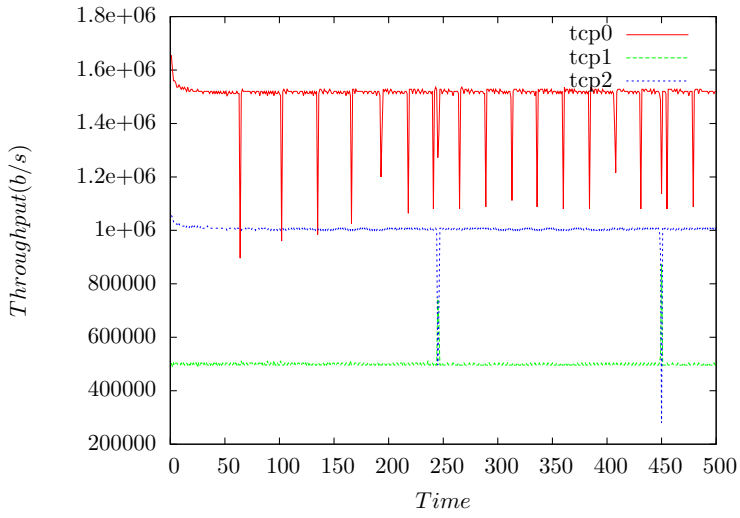


FIGURE 4.7 – Débits des flot avec politique Multipath Round Robin

Le MPRR corrige le problème provoqué par l'utilisation du Round Robin en réalisant des débits presque équitables en prenant compte toute la capacité offerte aux flots, les graphes montrent une amélioration par rapport au Tail drop traditionnel.

4.2.3 CCN et routage multipath

Même si le MPTCP s'avère performant en utilisant Tail drop, ou en utilisant le MPRR, on ne peut pas faire entièrement confiance à tous les utilisateurs de l'Internet. L'utilisateur détient toujours la possibilité de modifier son protocole de transport, ou d'utiliser le MPTCP non coordonné, par exemple. Rien n'oblige l'utilisateur à participer à la gestion de congestion.

On note aussi la difficulté d'implémenter un protocole MPTCP dans CCN, du fait que l'utilisateur ne peut choisir la destination de chaque paquet. On pourrait envisager une méthode statistique se basant sur l'historique des chemins traversés et leur RTTs. Si on ne peut choisir le chemin qui serait traversé à l'émission, il reste possible de savoir le chemin qui a été suivi par un paquet à la réception en stockant les noeud traversés dans chaque paquet, par exemple [27]. Malheureusement cette méthode reste compliquée et demande beaucoup de modifications au niveau de chaque paquet. De plus, la coordination des routeurs est nécessaire pour réaliser l'équité globale dans le réseau.

Nous pensons que les multipaths dans CCN ne devrait pas être gérés par les utilisateurs, ou du moins les utilisateurs ne sont pas responsables de la gestion du trafic dans le réseau. Nous ne pouvons qu'émettre des conseils permettant à l'utilisateur d'utiliser au mieux la bande passante et d'éviter les rejets d'Interests successifs rendant leur protocole de transport compliqué à gérer. C'est le réseau qui devrait distribuer les paquets équitablement sur les liens disponibles.

4.2.4 Performances des multipaths

Nous pensons que les flots multipaths devraient être géré par le réseau lui-même. Pour évaluer les performances des multipaths, nous proposons d'étudier deux segments de réseaux présentés dans la figure 4.8.

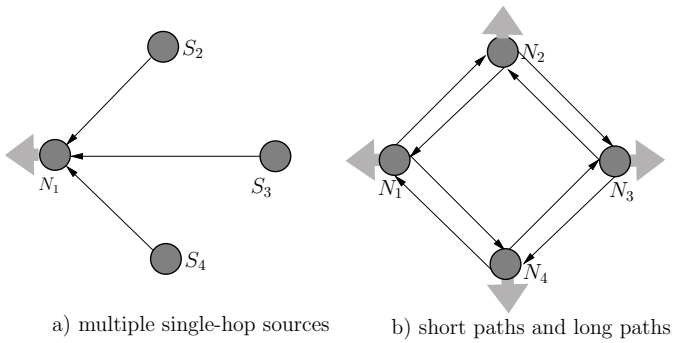


FIGURE 4.8 – Deux réseaux pour illustrer les Multipaths

Dans la figure 4.8a, les flot arrivants au noeud N_1 peuvent récupérer les données du

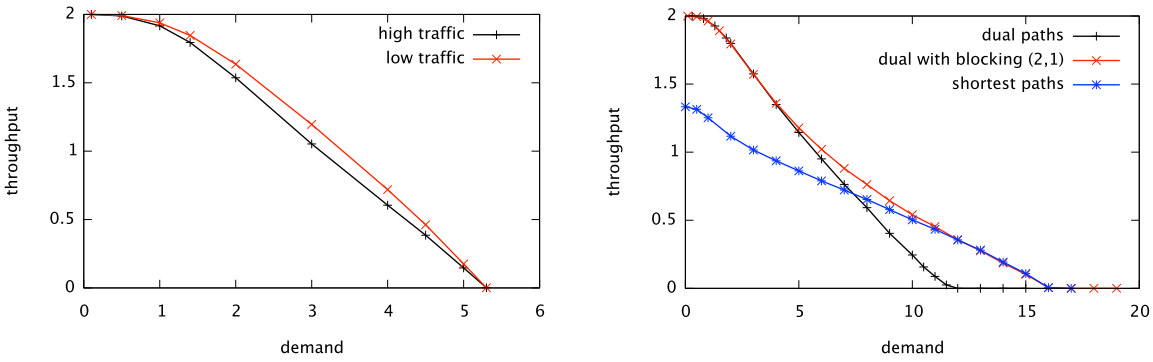
routeur S_1 localisée dans le noeud N_1, sinon dans le cas où la donnée est introuvable, la récupérer d'une des 3 sources S_4,S_2 ou S_3.

Dans la figure 4.8b, deux routeurs sont choisi aux hasard pour récupérer les données. Nous comparons les 3 cas suivants :

- un seul chemin qui correspond au chemin le plus court en nombre de sauts est utilisé par les flots,
- deux chemins sont utilisés conjointement,
- on utilise un contrôle de charge sélectif qui consiste à refuser l'accès au lien à tout flot multipath si le lien est un chemin secondaire, et si la charge du lien dépasse un certain seuil.

On utilise des simulations Monte-Carlo pour évaluer le débit moyen des flots en fonction de la charge, et comparer ainsi les performances des stratégies d'acheminement.

La figure représente la bande passante en fonction de la charge. Pour le premier



(a) a (b) b

FIGURE 4.9 – MPTCP coordonné priorise les flot unipath

réseau 4.9(a) la charge maximale à partir de laquelle le réseau est instable est de 5,29 exprimée en unités du débit d'un lien. Ceci peut être prédit par calcul. Dans ce cas, les chemins ont le même nombre de sauts. Dés qu'un chemin est surchargé on ne peut plus l'utiliser. L'utilisation du réseau est maximal avec le DRR et le contrôle de charge.

Les débits du deuxième réseau montrent que l'utilisation des chemins multiples sans aucune stratégie mène à une perte en capacité (la région de stabilité est réduite). Dans ce cas le débit offert est maximale au début. L'utilisation des chemins unicast offre une meilleur capacité en trafic, mais les débits offerts au début sont plus faibles. Afin d'offrir des débits plus importants à faible charge, tout en offrant une meilleure capacité en trafic, nous proposons d'appliquer le contrôle de charge sélectif.

Dès qu'un seuil de débit est atteint dans un lien, il faut refuser l'accès aux flots multipaths si le lien appartient à un chemin secondaire. Pour distinguer les chemins secondaires et principaux pour un flot on peut marquer les paquets qui traversent un chemin secondaire. On observe effectivement que les performances obtenues avec un contrôle de charge sélectif sont mieux que celles obtenus avec une utilisation exclusive des chemins les plus courts, et évidement mieux que l'utilisation aléatoire des chemins multipath.

En réalité les liens au coeur du réseau ont des débits très élevés dépassant le débit d'accès des utilisateurs. L'utilisation des chemins multiples n'est pas forcément bénéfique, les performances des caches localisés au niveau des routeurs peuvent sérieusement décroitre. Pour illustrer ce phénomène, on considère un réseau simple représenté dans la figure 4.10.

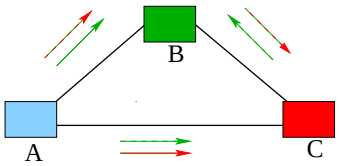


FIGURE 4.10 – tronçon de réseau pour démontrer l'impact des multipath sur le hit global

Des flots arrivent au noeud A selon un processus de Poisson. Si l'objet se trouve dans le noeud A alors le cache A répond à la requête. Si l'objet demandé se trouve dans les deux caches B et C alors un des deux caches est choisi au hasard et l'objet est récupéré du cache choisi. Si un des caches contient l'objet et que l'autre ne le contient pas, le chemin le plus long menant au cache détenteur d'objet est sélectionne avec une probabilité P. Si aucun des caches B ou C ne contient l'objet alors il est dirigé vers le serveur d'origine à travers le routeur B ou C (choisi au hasard). La popularité des requêtes suit une loi Zipf(0.8). On trace la probabilité de hit global de cette micro architecture en fonction de la probabilité de choix du chemin le plus long pour différentes valeurs de la taille des caches. On choisit une taille de catalogue de 10^4 objets. Conformément à la proposition CCN les caches ont la même taille.

Cette exemple illustre un contre exemple des bénéfices tirés par les multipaths. Même si les multipaths paraissent comme une solution intéressante pour augmenter le trafic véhiculé sur Internet, son utilisation dans les réseaux CCN tel que présenté par Van Jacobson ne parait pas forcément bénéfique. Ce constat a aussi été démontré par Rossini et al. [9]. Il est clair que le mieux pour les réseaux CCN est de n'utiliser les chemins les plus longs que pour les cas extrêmes ou un flot ne peut être servi par son chemin le plus court à cause d'une charge maximale dans un lien du chemin, et que le chemin le plus long ne contient aucun lien proche de la surcharge.

Nous proposons de maintenir le choix des chemins les plus courts comme choix principal. Si un flot est rejeté de son chemin le plus court à cause de la surcharge d'un lien appartenant à son chemin on peut dans ce cas seulement envisager d'emprunter un chemin secondaire, à condition que ce chemin n'atteint pas un certain seuil de

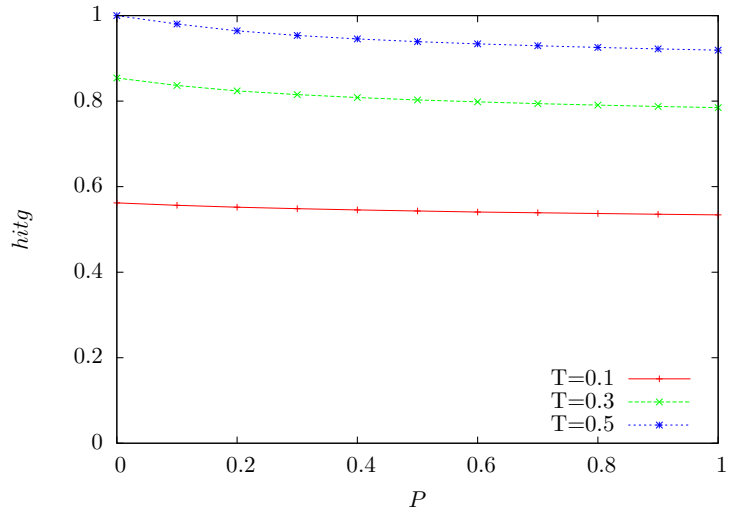


FIGURE 4.11 – Taux de hit en fonction de la probabilité du choix du plus long chemin

charge.

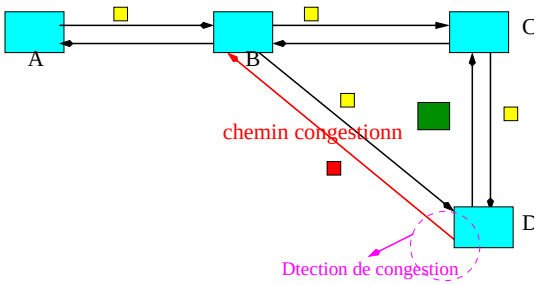


FIGURE 4.12 – Chemin plus longs choisis en cas de congestion uniquement

La figure 4.12 montre un exemple de l'utilisation des multipaths dans CCN. Quand un objet se trouve dans le cache D, les Interests suivent le plus court chemin A-B-D. Mais quand le cache D reçoit l'Interest il ne peut le servir à travers le chemin inverse car une congestion est détecté. Le flot est alors redirigé vers le noeud d'avant B qui propose un autre chemin vers le cache D. Le chemin étant non congestionné le flot emprunte le chemin A-B-C-D. La FIB du noeud B est mise à jour afin de véhiculer tous les paquets Interest du flot vers l'interface menant au noeud C et non pas au noeud D. D'autre part, un routage orienté contenu est lourd à mettre en place, une mise à jour des FIB à chaque fois qu'un objet est supprimé d'un cache peut avoir un impact sur plusieurs tables, surtout si les tables enregistrent les paquets et non les objets.

Chapitre 5

Simulations et expérimentations

Nous avons testé, par simulation et expérimentation certains des mécanismes que nous avons proposés.

5.1 Simulations

On considère un lien à 10 Mb/s partagé entre un flot Poisson à 5 Mb/s et un ensemble de flots permanents. Les flots sont ordonnancés en utilisant le DRR. Le flot de Poisson représente de manière simplifiée un ensemble de flots de débit beaucoup plus faible que le débit équitable. Les paquets de ce flot sont supposés appartenir à des flots différents. Les flots permanents simulés sont :

- AIMD (7/8) : l'utilisateur final implémente un contrôle de congestion AIMD avec facteur de réduction de la fenêtre CWND $\beta = 7/8$ et, pour ce flot, RTT = 10 ms.
- AIMD (1/2) : l'utilisateur final implémente un contrôle de congestion AIMD avec facteur de réduction de la fenêtre CWND $\beta = 1/2$ et RTT = 200 ms.
- CWIN (5) : L'utilisateur final maintient une fenêtre fixe de 5 packets et RTT = 200 ms.
- CWIN (100) : L'utilisateur final maintient une fenêtre fixe de 100 packets et RTT = 200 ms.
- CBR : L'utilisateur final envoie des paquets Interest à taux fixe correspondant à un débit constant de paquets data de 4 Mb/s.

Les résultats sont résumés dans les tableaux 5.1 et 5.2. On distingue les cas d'une détection rapide par l'utilisateur nommée "Rapid" et une détection par timeout fixé à 1s. On distingue aussi les deux cas avec "Interest discard" ou sans "Interest discard".

TABLE 5.1 – Débits en Mb/s

Flow	sans discard		Interest discard	
	Rapid	TO (1s)	Rapid	TO (1s)
AIMD (7/8)	1.20	1.24	1.23	1.31
AIMD (1/2)	1.19	1.10	1.12	0.84
CWIN (5)	0.19	0.19	0.19	0.19
CWIN (100)	1.20	1.24	1.23	1.32
CBR	1.20	1.24	1.23	1.32

TABLE 5.2 – Taux de rejets et de discard (perte/discard)

Flow	sans discard		Interest discard	
	Rapid	TO (1s)	Rapid	TO (1s)
AIMD (7/8)	.006/0	.01/0	0/.01	0/.01
AIMD (1/2)	.002/0	.003/0	0/.001	0/.003
CWIN (5)	.006/0	0/0	0/.0	0/0
CWIN (100)	.30/0	.18/0	0/.65	0/.22
CBR	.76/0	.75/0	0/.75	0/.74

Les flot agressifs CWIN(100) et CBR ont des débits à peu près égaux au flot TCP(7/8), mais le taux de rejets des data des flot agressifs est très important (30% pour le CWIN(100) et 76% pour CBR) ; les rejets data sont convertis en rejets Interest en utilisant le mécanisme Interest Discard. A partir de ces résultats, nous recommandons que les utilisateurs choisissent un protocole de transport AIMD agressif avec donc un facteur de réduction proche de 1.

5.2 Expérimentations

5.2.1 Fair sharing

Nous avons utilisé comme algorithme d'ordonnancement le DDR [22]. Cet algorithme utilise des files virtuelles, chacune correspondant à un identifiant de flot. A la réception d'un paquet, un hash est calculé à partir du nom d'objet, et le paquet est placé dans la file correspondante à cet identifiant. Les files sont implémentées comme une simple liste chainée appelé *ActiveList*. Lorsque la file globale atteint une taille maximale, le dernier paquet de la file de flot la plus longue est supprimé.

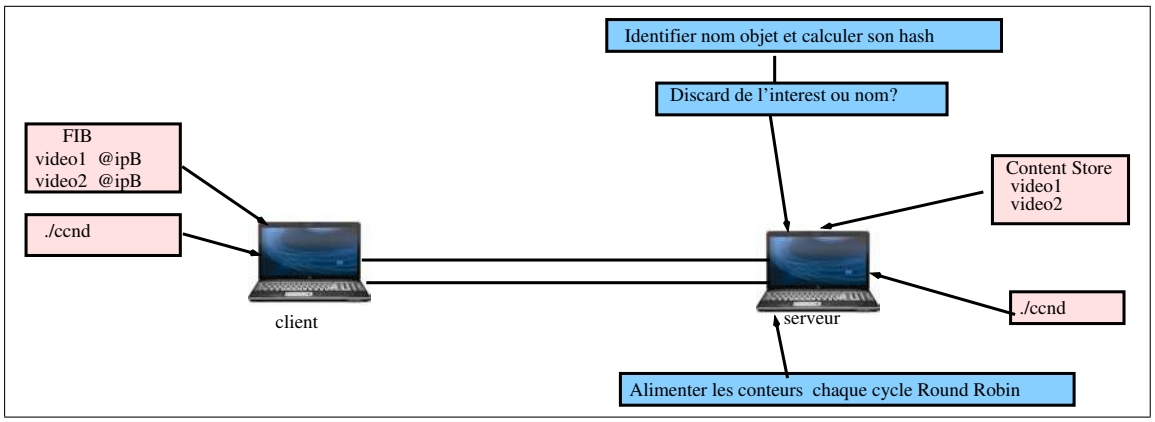


FIGURE 5.1 – Testbed et interest Discard

5.2.2 Interest discard

On implémente un compteur pour chaque flot dans l'*ActiveList* du DRR. Tous les compteurs seront incrémentés d'un quantum à chaque fois que l'ordonnanceur DRR complète un cycle (parcours toutes les files) jusqu'à une valeur maximale b.

A chaque fois qu'un Interest correspondant à un flot arrive sur la carte réseau, le compteur du flot est décrémenté d'un quantum. Si un Interest arrive et que le compteur du flot correspondant à l'Interest est à zéro, il faut supprimer l'Interest.

5.2.3 Scénarios et résultats

Nous avons implémenté un ordonnancement DRR [22] ainsi que l'Interest discard dans un réseau basique de deux noeuds. Un lien full duplex interconnecte deux machines Linux. Une machine joue le rôle du serveur et stocke des fichiers de données, l'autre machine est client, cherchant ces données. L'ordonnancement est implémenté dans l'espace noyau en utilisant une version modifiée du module sch_sfq développé par L.Muscariello et P.Viotti [28]. Cette nouvelle implémentation permet l'identification des flots par les noms d'objets. L'Interest discard est implémenté dans le noyau, les modifications suivantes ont été apportées :

- Création d'un compteur par flot.
- Incrémentation de tous les compteurs à chaque cycle DRR.
- Décrémentation d'un compteur à chaque fois qu'un Interest est envoyé.
- Rejet d'un Interest si le compteur est nulle.

Au niveau du serveur, nous appliquons un shaping à 10 Mb/s, lançons le démon ccnd, et chargeons des fichiers dans le référentiel. Au niveau de la machine cliente, nous lançons le démon ccnd, et récupérons les objets stockés dans le serveur en utilisant

deux applications : l'application ccncatchunks2 implémentée par PARC, et l'application cbr que nous avons développée pour envoyer les paquets Interest à un débit constant.

La figure 5.2 représente les débits instantanés dans le cas d'un ordonnancement FIFO, et dans le cas d'un ordonnancement DRR.

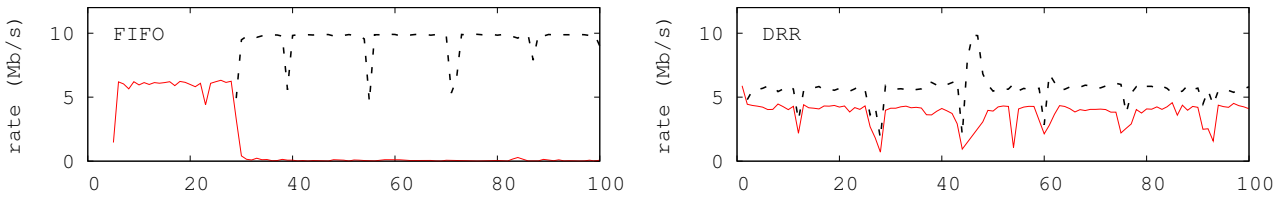


FIGURE 5.2 – Débits des flot : *cbr* et *ccncatchunks2*

Le flot ccncatchunks2 arrive à avoir son débit maximal en utilisant le Deficit Round Robin, contrairement à l'ordonnancement par défaut tail drop. Les résultats des expérimentations confirment les simulations. L'Interest Discard permet de convertir les rejets data en rejets Interest, ce qui aide à conserver la bande passante et à protéger l'opérateur.

	sans filtre	$b = 10$	$b = 100$
perte	.42	.002	.005
discard	0	.45	.46

Le tableau ci-dessus montre que l'Interest discard est un mécanisme efficace pour éviter les rejets data et donc éviter un gaspillage de la bande passante.

Chapitre 6

Conclusion

Dans cette partie, nous avons proposé un ensemble de mécanismes de gestion du trafic pour la proposition CCN. Cet ensemble comprend quatre volets essentiels :

- *La gestion du partage de bande passante.* Grâce à l'identification des flots par les noms d'objets, il est désormais possible de définir un flot sous CCN. Nous soulignions la nécessité de séparer files d'attente et caches parce qu'ils n'ont pas les mêmes exigences en termes de taille et de temps de réponse. La file d'attente devrait être de taille petite avec un temps de réponse rapide. Par contre les caches sont plus grands mais exigent un temps de réponse moins rapide et utilisent typiquement une politique LRU (remplacement de l'objet le moins récemment demandé). Le partage de bande passante est assuré au moyen de l'ordonnancement fair queuing (DRR de préférence) au niveau flot.
- *Des mécanismes pour utilisateurs.* Nous conseillons l'utilisation d'un protocole AIMD adaptatif, pas pour assurer l'équité, qui est réalisée directement par DRR, mais afin de limiter les pertes et le déclenchement de réémissions multiples lourdes à gérer. L'utilisateur ne gagne rien en étant agressif car le réseau partage équitablement la bande passante. La détection rapide des rejets assure l'efficacité du protocole de transport. Nous proposons donc une détection rapide de rejets au niveau des routeurs. Si un paquet Interest ne peut être servi ou si un paquet Data devrait être rejeté, un paquet data sans payload est envoyé vers l'usager. Nous utilisons cette méthode car, dans CCN, l'existence de chemins multiples entraine des problèmes de séquencement des paquets Data rendant impossible la détection rapide de perte en contrôlant les numéros de séquence. L'utilisation de sources multiples engendre en plus des variations importantes du RTT rendant difficile le réglage du seuil de Timeout.
- *Des mécanismes pour opérateurs.* Nous proposons un nouveau modèle de facturation où un usager ou un opérateur "achète" des paquets de Data en émettant

les paquets Interest correspondants. Ce mécanisme incite les opérateurs à investir dans des ressources réseaux afin de pouvoir "vendre" davantage de trafic. Les opérateurs sont également motivés à utiliser les caches afin d'éviter le rachat multiple fois d'un objet populaire. Nous proposons aussi un mécanisme d'Interest discard qui limite les rejets Data et permet à l'opérateur d'éviter de demander des paquets Data qui ne peuvent pas être revendus en aval.

– *Des stratégies d'acheminement.* Le multicast sous CCN est compatible avec le fair queuing que nous suggérons d'utiliser au niveau des routeurs. CCN utilise le multicast comme une partie de la proposition de sorte que deux flux synchronisés demandant le même objet ne peuvent le télécharger parallèlement sur un lien. Une seule demande est envoyée pour les deux flux, ce qui évite la division du débit des flux due au fair queuing. Si les demandes ne sont pas synchronisées l'utilisation des caches permet de maintenir le débit de téléchargement grâce au stockage temporaire des paquets en cours de téléchargement. Par contre l'utilisation du fair queuing peut poser un sérieux problème en ce qui concerne les multipaths. Le fair queuing annule le comportement du flux multipath coordonné et le transforme en un flux multipath non coordonné. Une équité locale par lien est réalisée mais l'équité globale ne l'est pas car le flux multipath reçoit plus de débit qu'un flux unipath. Nous corrigeons ce problème par la conception d'un protocole MPRR (multipath Round Robin). Un protocole de type MPTCP est difficile à réaliser sous CCN puisque l'utilisateur n'a aucune visibilité sur les chemins utilisés. Nous proposons donc une gestion par flot plutôt qu'une gestion par paquet. Il suffit d'observer la charge des liens et de n'accepter aucun nouveau flot sur un chemin long que si la charge des liens est assez faible. Nous avons également observé que l'utilisation de multipaths nuit à l'efficacité des caches dans certains cas.

Suite à nos observations liés à la dégradation du taux de hit global due à l'utilisation des multipaths, une étude des performances des caches est nécessaire, car la gestion du trafic en dépend. Cette étude est l'objet de la prochaine partie.

Deuxième partie

Performances des caches

Chapitre 7

Introduction

7.1 Problématique

Dans ce chapitre, nous traitons le problème de la performance des caches dans les réseaux orientés contenus. Compte tenu des modifications majeures à apporter aux réseaux dans le cas où une mise en oeuvre de CCN est envisagée (mise à jour des caches, protocoles, mémoires distribuées), il est important de mesurer le gain apporté par cette architecture.

Il est primordial de mesurer la quantité et la manière dont arrivent les objets. Les conclusions que nous pouvons tirer d'une étude de performances dépend de la popularité des objets arrivant aux caches, et de la taille des catalogues.

Nous souhaitons apporter des conclusions pratiques en utilisant des données réelles. On note que la diffusion de contenus représente 96% du trafic Internet. Ce contenu est constitué d'un mix de données. Nous avons alors mis en place une évaluation d'une hiérarchie de caches à deux niveaux en utilisant un mix de flux reflétant un échange réel de données sur Internet.

7.2 Etat de l'art

7.3 Contributions

Dans cette partie, on évalue le taux de hit, pour une hiérarchie de caches à deux niveaux, avec un mix de flux réel. Ceci en utilisant un modèle simple permettant d'effectuer des calculs rapides pour des tailles importantes de cache. Ce modèle, précédemment proposé dans la littérature, a été testé, vérifié, et démontré mathématiquement. Des simulations ont été effectuées pour confirmer son exactitude. Nous avons effectué

les calculs en utilisant un mix de flux reflétant le partage actuel du trafic sur Internet. Nous proposons un stockage des contenus VoD au niveau des routeurs d'accès, vu leur volume faible par rapport aux autres types de données. Les autres types devraient être stockés dans un cache très volumineux, probablement constituant un deuxième niveau de caches.

Chapitre 8

Mesure du trafic et performances des caches

Pour estimer les taux de hit d'une architecture à deux niveaux, il est primordial de mesurer les caractéristiques du trafic, car les taux de hit dépendent fortement de la nature du trafic et de son volume.

8.1 Mesure du trafic

Nous présentons les caractéristiques du trafic Internet, et nous discutons des paramètres les plus importants pour nos évaluations.

8.1.1 Types de contenu

Le "Cisco Visual Networking Index" publié en 2011 [29] classifie le trafic Internet et la demande globale prévue pour la période 2010-2015. 96% du trafic représente le transfert de contenus susceptibles d'être stockés dans les mémoires cache. On peut les classifier en quatre catégories :

- **Données web :** Ce sont les pages web visitées par les internautes.
- **Fichiers partagés :** Généralement gérés par des protocoles pair à pair, créant une communauté d'entraide : Un utilisateur (leecher) peut télécharger un fichier stocké dans une des machines des autres utilisateurs (seeders). Dès que son téléchargement est terminé, le leecher devient à son tour seeder. Les réseaux pair à pair rencontrent de plus en plus de problèmes à cause de la violation des droits

d'auteur par leurs utilisateurs. Ces derniers peuvent mettre en téléchargement du contenu illégal. Récemment, à titre d'exemple, le site Demonoid n'est plus disponible, probablement à cause de la violation des droits d'auteur.

- **Contenu généré par les utilisateurs (UGC) :** C'est un ensemble de contenus générés par les utilisateurs, ou directement mis à disposition par ces derniers. La communauté utilisant ce partage utilise des logiciels libres, des contenus avec des licences de droit d'auteur flexibles, permettant des échanges simples entre des utilisateurs, même éloignés géographiquement. A la différence des réseaux pair à pair, les données sont sauvegardées sur les serveurs privées du fournisseur de contenu. Il détient alors la possibilité de vérifier les contenus chargés par les utilisateurs avant leur publication.
- **Vidéo à la demande (VoD) :** C'est une technique de diffusion de données permettant à des utilisateurs de commander des films ou émissions. La télévision sur IP est le support le plus utilisé. Le service VoD est proposé généralement par des fournisseurs d'accès Internet, et il est dans la plupart des cas payant. Le contenu proposé est loué pour une période donnée, assurant ainsi le respect des droits numériques.

Les proportions du trafic sont indiqués dans le tableau 8.1.

	Fraction du trafic (p_i)		taille de la	taille moyenne
	2011	2015	population(N_i)	des objets (θ_i)
Web	.18	.16	10^{11}	10 KB
File sharing	.36	.24	10^5	10 GB
UGC	.23	.23	10^8	10 MB
VoD	.23	.37	10^4	100 MB

TABLE 8.1 – Les caractéristiques des contenus du trafic Internet

8.1.2 La taille des contenus et des objets

- **Web :** La société Netcraft [1] publie chaque mois le nombre de sites, estimé grâce à un sondage fait auprès de sociétés d'hébergement et d'enregistrement des noms de domaine. Elle estime le nombre de sites actifs à 861 379 152, en considérant la moyenne de nombre de pages par site à 273 [2] nous comptons plus de $2 * 10^{11}$ pages web. Pour notre étude, on suppose que le nombre de pages web est de 10^{11} et leur taille moyenne est de 10KB [30].
- **Fichiers partagés :** On estime le nombre de fichiers partagés grâce aux statistiques relevées sur le site Demonoid[3] à 400 000 fichiers de taille moyenne de 7.4 GB. Nous arrondissons ces chiffres dans le tableau 8.1.

[1] http ://news.netcraft.com/archives/category/web-server-survey/
[2] http ://www.boutell.com/newfaq/misc/sizeofweb.html
[3] www.demonoid.me/

- **UGC :** Les contenus UGC sont dominés par Youtube. Une étude récente, faite par Zhou et al. [31], estime le nombre de vidéos Youtube à 5×10^8 de taille moyenne de 10 MB. Actuellement avec une simple recherche du mot clef "a" sur Youtube nous comptons plus de 10^9 vidéos.
- **VoD :** Les vidéos à la demande sont estimées à quelques milliers et sont de taille moyenne de 100 MB. Ce sont sans doute des sous-estimations avec l'essore récente de certaines applications VoD mais elles sont suffisamment précises pour les évaluations présentées dans la suite.

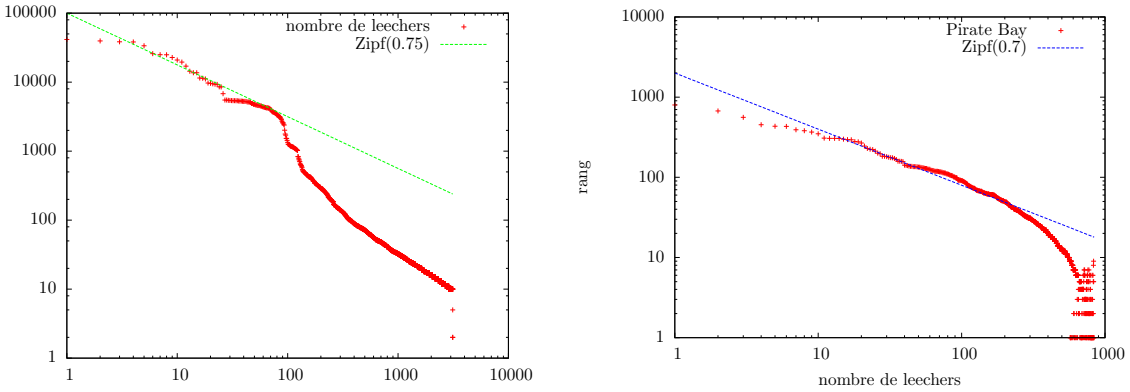
8.1.3 Distribution de la popularité

La distribution de la popularité est un des éléments essentiels du calcul des performances d'un cache.

- **Web :** La popularité des pages web suit généralement la loi de Zipf : le taux de demandes $q(n)$ pour le $n$iéme objet le plus populaire est proportionnel à $1/n^{\alpha}$. Selon [32] et [30] le paramètre α varie entre 0.64 and 0.83.
- **Fichiers partagés :** Il est possible de calculer la popularité des torrents en utilisant les statistiques extraites du site Demonoid. En entrant un mot clef, on peut classer les torrents d'une manière décroissante suivant le nombre de téléchargements en cours (mais le site ne permet l'affichage que des 10 000 premiers et les 10 000 derniers torrents). La loi de popularité correspond à peu près à une loi de Zipf de paramètre α égal 0.82. On estime que la popularité du site PirateBay suit une loi de Zipf de paramètre 0.75.
  On trace la popularité des vidéos partagés pour deux sites "PirateBay" et "torrentreactor" [4]. Après une recherche par mot clef, les sites affichent les vidéos et le nombre de leechers correspondants. En choisissant comme mot clef la seule lettre "a", et après un tri décroissant du nombre de leechers, nous traçons les popularités présentées dans 8.1(a) et 8.1(b). Pour le site torrentreactor, la popularité suit la loi Zipf(0.75) pour les premiers rangs, puis la courbe s'incline et suit une loi Zipf(1.2) pour la queue de la loi. La même observation concerne le site PirateBay.

- **UGC :** Les flux UGC suivent une loi de Zipf avec α estimé à 0.56 [11] ou à 0.8 [13]. Des travaux récents de Carlinet et al. [33] suggèrent plutôt une loi Zipf(0.88).
- **VoD :** L'étude de Carlinet et al. évalue également les VoD. La loi de popularité n'est pas de Zipf, mais une combinaison de deux lois de Zipf. La première est de paramètre 0.5 pour les 100 objets les plus populaires, la deuxième est de paramètre 1.2 pour les objets suivants. Des statistiques étudiées par Yu *et al.* [34] pour un service VoD en Chine suggèrent une loi de Zipf avec α variant ente 0.65 et 1.

[4] http ://www.torrentreactor.net



(a) torrentreactor

(b) Pirate Bay

FIGURE 8.1 – La popularité des vidéos partagées sur torrentreactor et Pirate Bay

8.2 Le taux de hit d'un cache LRU

8.2.1 Independent Reference Model

Afin d'utiliser les modèles mathématiques, on considère généralement un ensemble d'objets ayant des popularités fixes, ainsi qu'un catalogue d'objets fixe. C'est le modèle dit "independance reference model" ou IRM. En réalité, les objets changent de popularité et les catalogues ne cessent d'augmenter. Une prise en compte d'une telle complexité ne peut être résolue par modèle mathématique, et est très complexe à simuler. Cependant, la variance des popularités est négligeable par rapport au temps de remplissage d'un cache. On peut considérer que les modèles sont applicables sur un intervalle de temps où les popularités et les catalogues seront approximativement fixes.

Afin d'appliquer ce modèle mathématique, il faut aussi que les requêtes soient indépendantes. Ceci est vrai si des demandes arrivent d'un grand nombre d'utilisateurs agissant de façon indépendante. Ces conditions s'appliquent pour un premier niveau de cache. Mais pour les niveaux supérieurs, la corrélation des demandes invalide le modèle IRM. Cependant, selon Jenekovic et Kang [35], la corrélation des demandes qui débordent du premier niveau d'un simple réseau de caches à deux niveaux, a un faible effet sur la probabilité de hit observée. Nous avons d'ailleurs vérifié par simulation l'effet de la corrélation des demandes pour un réseau de caches en arbre.

Pour conclure, les modèles mathématiques basés sur l'IRM peuvent être appliqués pour les réseaux CCN, car la popularité des objets varient d'une manière faible par rapport à la vitesse de remplissage des caches et l'indépendance est respectée.

8.2.2 Les modèles analytiques

Une politique de remplacement LFU (least frequently used) reste la politique idéale ; mais, il est impossible de mettre en place cet idéal car la popularité des objets est en général inconnue. Van Jacobson propose un ordonnancement LRU (least recently used) dans tous les caches même si plusieurs travaux remettent en question les performances réseau avec une utilisation exclusive de LRU.

Les études de performance des réseaux CCN nécessitent l'utilisation de modèles mathématiques afin de confirmer et de généraliser les observations tirées des simulations et expérimentations. Notre objectif, qui est d'évaluer la performance pour les très grandes populations (jusqu'à 10^{11} pages web, par exemple) et leur mélange, n'est pas envisageable par simulation. Les modèles exacts sont très complexes, même dans le cas basique d'un seul cache. La complexité de ces modèles croit d'une façon exponentielle avec la taille des caches et le nombre d'objets. Il est donc plus intéressant de créer des modèles simplifiés basés sur des approximations.

La majorité des modèles ont été conçus pour une politique de remplacement LRU avec le modèle dit IRM (Independent Reference Model). Quelques travaux récents ont traité cette problématique. En effet, G. Carofiglio *et al.* [36] proposent un modèle généralisé dans le cas d'un réseau de caches (architecture en arbre) ; ce modèle se limite aux cas d'arrivées suivant un processus de Poisson et une loi de popularité de type Zipf avec $\alpha > 1$. Ce modèle s'applique à la mise en cache par paquet (comme CCN) et prend en compte la dépendance entre les paquets data d'un même objet. Un autre modèle pour les réseaux de caches a été proposé par E.Rosensweig *et al.* [37] ; c'est un modèle adapté à toute architecture. Cependant, la complexité du calcul du taux de hit, dû à Dan et Towsley [38], limite cette approche à des réseaux et des populations de taille relativement faible.

8.2.3 La formule de Che

Nous pensons qu'une mise en cache par objet est plus simple à déployer et à utiliser qu'une mise en cache par paquet. La proposition de Che *et al.* [39] est particulièrement intéressante. Hormis sa facilité d'utilisation par rapport aux autres modèles, sa grande précision a été démontrée dans plusieurs cas. On considère un cache de taille C, des objets appartenant à un catalogue de taille M arrivent au cache suivant une loi de popularité $pop(n)$ proportionnelle à $q(n)$. Sous un système conforme au modèle IRM, la probabilité de hit $h(n)$ d'un objet n selon l'approximation de Che est estimée à :

$$h(n) = 1 - e^{-q(n)t_c}, \tag{8.1}$$

où t_c est la solution de l'équation :

$$C = \sum_n (1 - e^{-q(n)t_c}). \tag{8.2}$$

Cette approximation est centrale pour le reste du travail. Voici quelques éléments expliquant sa précision et sa validité comme modèle mathématique. On note $T_c(n)$ le temps où exactement C objets différents de n ont été demandés. On suppose une première demande de l'objet n faite à l'instant 0, la prochaine requête pour l'objet n a lieu à τ_n, cette demande est un hit si $\tau_n < T_c(n)$. La probabilité de hit de l'objet n peut être exprimée par :

$$h(n) = P(\tau_n < T_c(n)). \tag{8.3}$$

Che et al. ont mesuré par simulation $T_c(n)$ et ont observé qu'il est presque déterministe et montre une très faible variation en fonction du rang même pour des catalogues petits (catalogue de 10 000 objets). Cette variable est presque indépendante de l'objet n et est caractéristique au catalogue On pose alors $E(T_c(n)) = t_c$ que Che et al considèrent comme le "temps caractéristique" du cache. Puisque les arrivées de requêtes suivent un processus de Poisson, le temps inter-arrivée τ_n suit une loi exponentielle de paramètre $q(n)$. On a donc la probabilité de hit $h(n)$ de l'objet n, en réécrivant (8.3 :

$$h(n) = P(\tau_n < t_c) = 1 - exp(-q(n)t_c)$$

.

Dans l'intervalle [0,t_c], nous avons exactement C arrivées sans compter l'objet n. Donc, à cet instant précis, parmi les M objets du catalogue, C objets exactement sont arrivés au cache à l'instant t_c, sans compter l'objet n. Ceci peut être exprimé par :

$$\sum_{i=1, i\neq n}^{M} P(\tau_i < t_c) = C$$

où τ_i est le temps séparant le début de l'observation $t = 0$ du temps d'arrivée de la demande de l'objet i au cache donc exponentielle de paramêtre $q(i)$. Cette équation permet de calculer le temps caractéristique des caches t_c. Mais pour plus de facilité, l'équation devient : $\sum_{i=1}^{M} P(\tau_i < t_c) = C$. Ceci est valable si la popularité individuelle de l'objet est relativement petite par rapport à la somme des popularités. En utilisant le fait que τ_i est de loi exponentielle de paramètre $q(i)$, l'équation devient l'équation (8.2), $C = \sum_{i=1}^{M}(1 - e^{-q(i)t_c})$.

L'approximation n'est pas seulement précise dans le cas, envisagé par Che et al, d'un grand cache et d'une population importante, mais également pour des systèmes très petits. Nous avons vérifié la validité de l'approximation par simulation, pour un seul cache de taille 10^4 et une loi Zipf(0.8) ou un cache de taille 16 et une loi géométrique Geo(0.5). La figure 8.2 montre la précision de l'approximation et sa validité même dans le cas d'un petit cache.

Pour calculer le $h(n)$, il faut d'abord trouver le t_c qui est le zéro de l'équation (8.2). Pour trouver le zéro de cette équation on utilise la méthode de Newton : On peut trouver une valeur proche du zéro d'une fonction $f(x)$ en calculant successivement

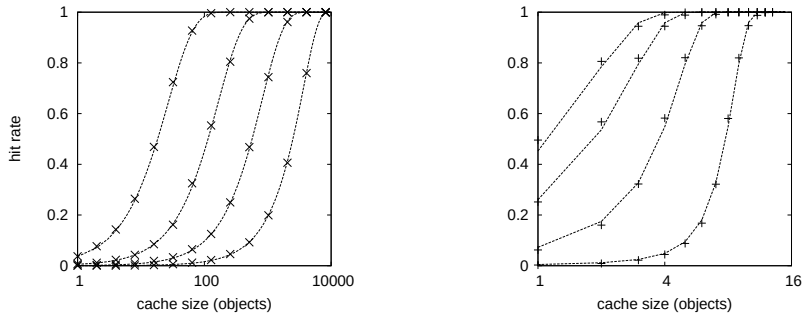


FIGURE 8.2 – Taux de Hit en fonction de la taille du cache en utilisant l'approximation de Che : à gauche, $N = 10^4$, popularité de Zipf(.8) , rangs 1, 10, 100, 1000 ; à droite, $N = 16$, popularité geo(.5), rangs 1, 2, 4, 8.

une suite de valeurs x_i jusqu'à l'obtention d'une approximation satisfaisante. x_{i+1} est calculé à partir de la valeur de x_i :

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{8.4}$$

Le x_0 est choisi arbitrairement et on calcule x_1 en utilisant la formule 8.4. On recalcule f(x_1) et si f(x_1) est suffisamment proche de zéro, x_1 est alors le zéro de f(x). Sinon on calcule x_2,... Nous constatons que la convergence est extrêmement rapide.

8.3 Autres politiques de remplacement

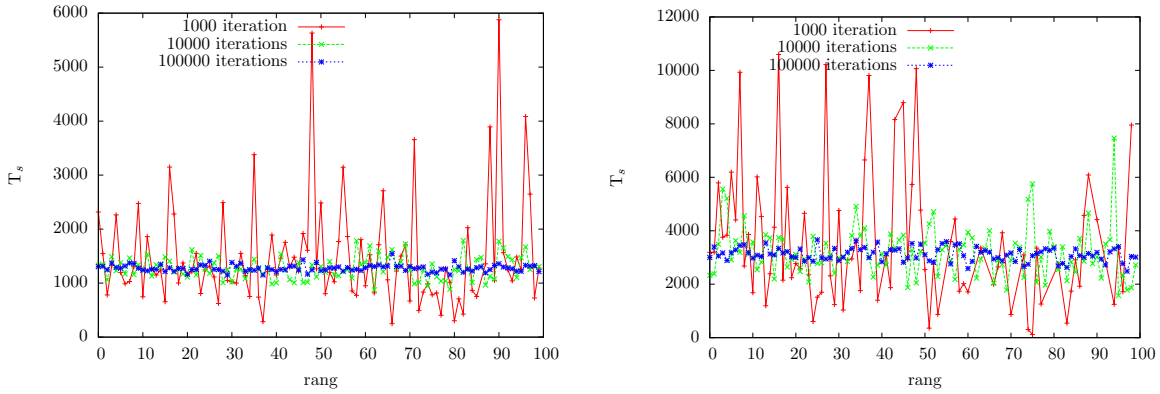
8.3.1 Le cache Random

8.3.1.1 Relation entre taux de hit et temps moyen de séjour

On considère un cache de taille C utilisant une politique de remplacement Random. Dans cette politique, lorsqu'il faut libérer de la place pour cacher un nouveau contenu, le contenu à éliminer est choisi au hasard. La taille du catalogue est M. On note $T_s(n)$ le temps de séjour de l'objet n. On commence nos observations sur des simulations pour un catalogue de taille petite (100 objets), et un cache de 50 objets. Nous étudions le cas des objets arrivant suivant une loi Zipf(0.6) et Zipf(1.2). Les requêtes arrivent avec un taux de 100 requêtes/s.

On lance la simulation pour 1 000, 10 000 et 100 000 itérations. On trace dans ces trois cas la moyenne du temps de séjour $T_s(n)$ en fonction de n (voir figure 8.3).

On remarque que, plus le nombre d'itérations augmente, plus la moyenne du temps de séjour tend vers une valeur précise. Cette valeur est la même quelque soit le rang. En se basant sur cette observation, on considère que la valeur du temps de séjour est indépendante de n (adoptant la même approximation que Che). Le T_s est fixe quand



(a) Zipf(0.6) (b) Zipf(1.2)

FIGURE 8.3 – Temps de séjour en fonction du rang pour un cache Random

le temps de simulation devient grand, car tout objet du cache a la même probabilité que les autres objets d'être éliminé. Son élimination dépend surtout du taux de miss du cache qui devient fixe et stable après un certain temps de simulation.

Nous appliquons la formule de Little. La probabilité de hit d'un objet i est exprimée par : $h(n) = \lambda(n)T_s(n)$ où $T_s(n)$ est la moyenne du temps de séjour de l'objet n. $T_s(n)$ étant fixe et indépendant de n on pose $T_s(j) = T_s \ \forall j$. Le taux d'entrée dans le cache est $\lambda(n) = (1 - h(n))pop(n)$ où $pop(n)$ est la popularité normalisée de l'objet n. On obtient alors : $h(n) = (1 - h(n))pop(n)T_s$
Finalement $h(n)$ peut être exprimé par :

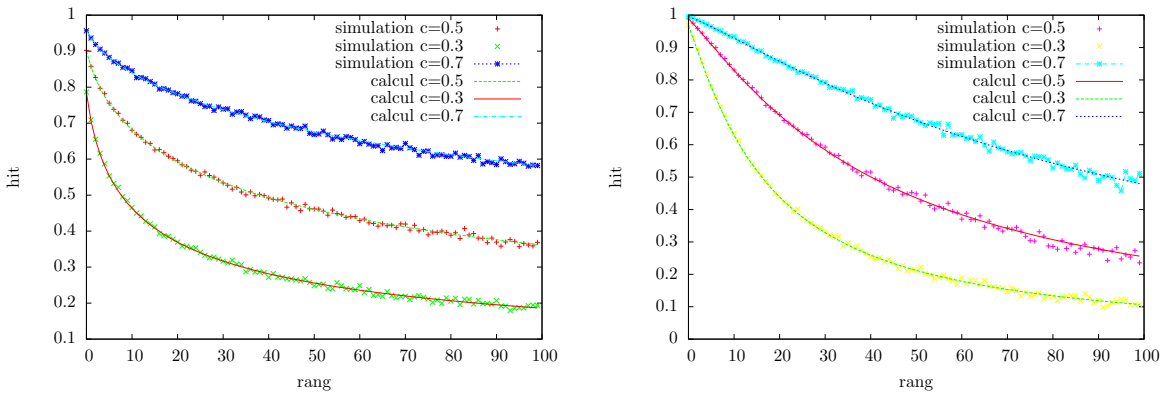
$$h(n) = \frac{pop(n)T_s}{1 + pop(n)T_s}. \tag{8.5}$$

La moyenne du temps de séjour peut être calculée en utilisant l'équation suivante (comme dans le cas d'un cache LRU) :

$$\sum_{i=1}^{M} h(i) = \sum_{i=1}^{M} \frac{pop(i)T_s}{1 + pop(i)T_s} = C. \tag{8.6}$$

L'équation (8.6) peut être résolue avec la méthode de Newton. Nous utilisons la valeur T_s retrouvée dans l'équation (8.5) pour déterminer les $h(n)$.

Les valeurs $h(n)$ trouvé par calcul et $h(n)$ trouvé par simulation sont comparées dans la Figure 8.4 pour différentes valeurs de taille de cache et pour les deux lois, Zipf(0.6) et Zipf(1.2) et pour différentes tailles de cache : $c = C/M = 0.3,\ 0.5,\ 0.7$.



(a) Zipf(0.6) (b) Zipf(1.2)

FIGURE 8.4 – Taux de hit en fonction du rang pour un cache Random

8.3.1.2 Approximation de Fricker

Dans Fricker, Robert and Roberts [40], l'approximation suivante est donnée :

$$T_s \simeq \frac{\tau_C}{\sum_{j \neq n} q(j)}, \tag{8.7}$$

où τ_C est une constante. On va discuter de la validité de cette approximation. Le temps de séjour d'un objet n peut être exprimé, comme représenté dans la figure 8.5, par une somme de temps t_j où t_j est la durée entre deux requêtes successives.

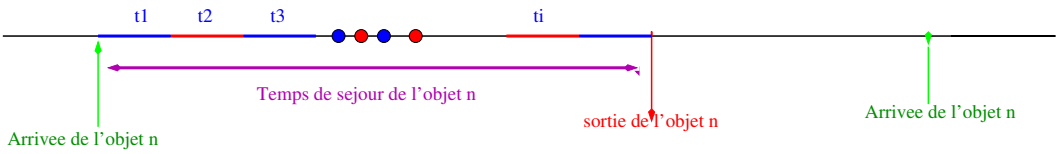


FIGURE 8.5 – Représentation du temps de séjour

A chaque arrivée au cache, l'objet n peut être retiré du cache avec une probabilité de $1/C$ si l'objet arrivé n'appartient pas déjà au cache. On suppose que toute nouvelle arrivée au cache implique une mise à jour même si cette arrivée est un hit. Soit n fixé. Calculons le temps de séjour $T_s(n)$. Tout objet i arrive au cache suivant un processus de Poisson de taux $q(i)$. Les temps inter-arrivées Z_i sont des variables aléatoires indépendantes de loi exponentielle de paramètre $q(i)$. La prochaine requête susceptible de retirer l'objet n du cache se passe à un temps X_{n1}

$$X_{n1} = \inf_{i \neq n}(Z_i). \tag{8.8}$$

Donc X_{n1} suit une loi exponentielle de paramètre $\sum_{i\neq n} q(i)$. Comme la politique de remplacement est Random, on en déduit facilement que

$$T_s(n) = \sum_{j=1}^{Y} X_{nj} \tag{8.9}$$

où X_{nj} est de loi exponentielle de paramètre $\sum_{i\neq n} q(i)$, et Y est de loi géométrique de paramètre $1 - 1/C$ sur N^*, indépendant de $(X_{nj})_{j\geq 1}$.

D'où, en passant à l'espérance dans l'équation (8.9),

$$T_s(n) = \sum_{i=1}^{+\infty} P(Y = i) \sum_{k=1}^{i} E(X_{nk}). \tag{8.10}$$

Or,

$$E(X_{nk}) = 1/\sum_{j\neq n} q(j),$$

et comme Y suit une loi géométrique de paramètre $1 - 1/C$ sur N^*, il vient que $E(Y) = C$. En effet, une v.a. de loi géométrique de paramètre a sur N^* est de moyenne $1/(1-a)$. En reportant dans l'équation (8.10 le temps de séjour moyen peut donc être exprimé par :

$$T_s(n) = \frac{E(Y)}{\sum_{j\neq n} q(j)} = \frac{C}{\sum_{j\neq n} q(j)}.$$

Revenons à l'approximation (8.7). L'idée sous-jacente dans [40] est qu'on peut approximer le temps de séjour de n en supposant que toute arrivée même un hit implique une mise à jour. Cela revient à supposer que tous les objets autres que n sont hors du cache. Intuitivement, cela est justifié si

1) le cache est petit devant la taille du catalogue,
2) les objets les plus populaires sont hors du cache car ce sont eux qui contribuent le plus à $\sum_{j\neq n} q(j)$.

Cette deuxième condition n'est pas du tout naturelle. On va voir, en traçant les différentes approximations du taux de hit, que cela est vrai pour une loi de popularité de Zipf de paramètre $\alpha < 1$ où les objets ont des popularités plus voisines que pour $\alpha > 1$ où les objets les plus populaires sont dans le cache avec forte probabilité.

8.3.1.3 Approximation de Gallo

Gallo et al [41] ont proposé une approximation pour le taux de hit pour une valeur de $\alpha > 1$. La probabilité de miss d'un objet i est approximée quand C est grand par :

$$Miss(i) = \frac{\rho_\alpha i^\alpha}{C^\alpha + \rho_\alpha i^\alpha}$$

où

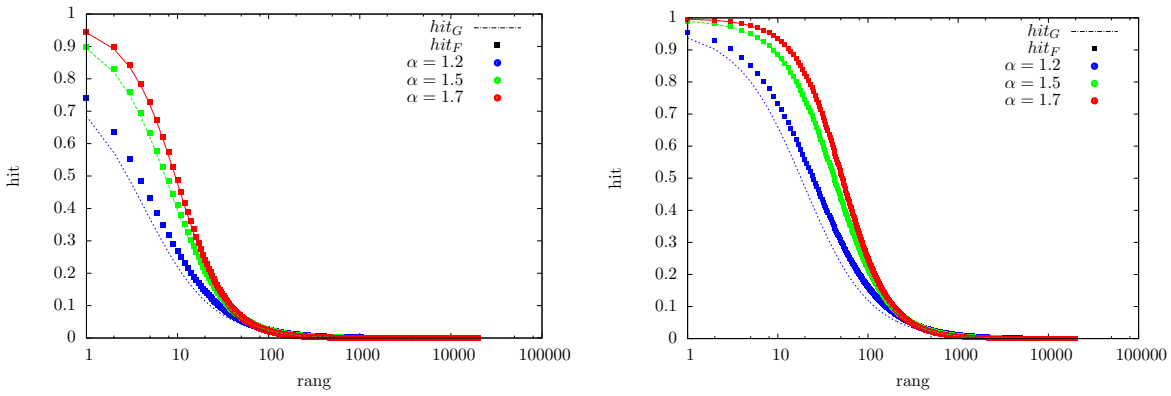
$$\rho_\alpha = \left(\frac{\frac{\pi}{\alpha}}{sin(\frac{\pi}{\alpha})} \right)^\alpha .$$

Cela revient à

$$hit_g(i) = 1 - Miss(i) \approx \frac{1}{\rho_\alpha (i/C)^\alpha + 1}.$$

Donc tout calcul fait, le temps de séjour devrait être proportionnel à $\left(C * \frac{sin(\frac{\pi}{\alpha})}{\frac{\pi}{\alpha}} \right)^\alpha$ pour $\alpha > 1$.

On compare l'approximation du taux de hit avec l'approximation de Gallo et al [41] hit_g pour des valeurs de $\alpha > 1$ pour un catalogue de $M = 20\ 000$. Voir Figure 8.6. On remarque que les deux approximations sont proches pour des valeurs de caches mêmes petites ($C \geq 20$).



(a) C=20 (b) C=100

FIGURE 8.6 – Comparaison des taux de hit avec l'approximation de Fricker et Gallo

8.3.2 Le cache LFU

Le cache LFU ne stocke que les objets les plus populaires. Donc, la probabilité de hit LFU peut être calculée, pour un catalogue de taille M et pour un cache LFU de taille C objets :

$$\begin{cases} hit(i) = 1; 0 \leq i \leq C, \\ hit(i) = 0; i > C. \end{cases}$$

La probabilité de hit globale d'un cache LFU peut donc être exprimée par :

$$hit_g = \sum_{i=1}^{C} pop(i)$$

où $pop(i)$ est la popularité normalisée de l'objet i.

Pour une loi de Zipf, la popularité normalisée de l'objet i peut être exprimée par :

$$pop(i) = \frac{1/i^\alpha}{\sum_{k=1}^{M} 1/k^\alpha}$$

Donc la probabilité globale de hit pour un cache LFU peut être exprimée par :

$$hit_g = \frac{\sum_{i=1}^{C} 1/i^\alpha}{\sum_{i=1}^{M} 1/i^\alpha}.$$

Soit i un entier et t un réel tel que $i \leq t \leq i+1$. Pour $\alpha > 0$, on a :

$$\frac{1}{(i+1)^\alpha} < \frac{1}{t^\alpha} < \frac{1}{i^\alpha}.$$

Donc,

$$\frac{1}{(i+1)^\alpha} < \int_i^{i+1} \frac{1}{t^\alpha}\,\mathrm{d}t < \frac{1}{i^\alpha},$$

d'où

$$\frac{(M+1)^{1-\alpha}-1}{1-\alpha} < \sum_{i=1}^{M} \frac{1}{i^\alpha} < \frac{M^{1-\alpha}}{1-\alpha}$$

et

$$\frac{(C+1)^{1-\alpha}-1}{1-\alpha} < \sum_{i=1}^{C} \frac{1}{i^\alpha} < \frac{C^{1-\alpha}}{1-\alpha}.$$

Puisque le nombre d'objets est grand, nous considérons $M+1 \approx M$. Nous utilisons des caches d'au moins quelques centaines d'objets donc, $C+1 \approx C$ nous concluons que :

$$\sum_{i=1}^{M} \frac{1}{i^\alpha} \approx \frac{M^{1-\alpha}}{1-\alpha} \text{ et } \sum_{i=1}^{C} \frac{1}{i^\alpha} \approx \frac{C^{1-\alpha}}{1-\alpha}. \tag{8.11}$$

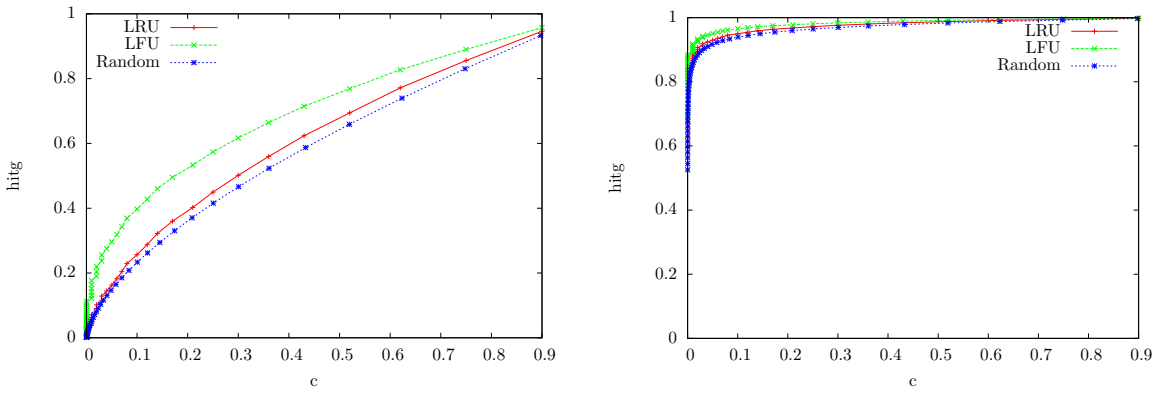
La probabilité de hit global pour un cache LFU peut être exprimée par :

$$hit_g = \left(\frac{C}{M}\right)^{1-\alpha},$$

8.3.3 Comparaison des politiques de remplacement

On compare les deux politiques de remplacement LRU et Random en fonction du rang, pour des valeurs de $\alpha = 0.6$ et $\alpha = 1.2$, et pour un cache de 50% la taille du catalogue. On fixe le catalogue $M = 10^6$ objets.

Il est clair que la politique LRU est plus performante que Random. On remarque aussi que l'écart entre LRU et Random est réduit pour un $\alpha = 1.2$, ce qui est confirmé par l'étude de Gallo et al. [41]. Cet écart se réduit de plus en plus quand α grandit. Mais pour une comparaison effective, il est impératif de comparer le taux de hit global des caches Random, LRU et LFU.



(a) Zipf(0.6)

(b) Zipf(1.2)

FIGURE 8.7 – Taux de hit global en fonctions des tailles de cache normalisé

On remarque que la différence des performances des caches est plus grande et visible dans le cas d'un Zipf(0.6). Cette différence diminue pour un $\alpha = 1.2$, non seulement entre LRU et Random, mais aussi LFU. Les caches deviennent aussi plus efficaces. Or, selon notre étude bibliographique et les statistiques tirées de certains sites fournisseurs de données, on sait que le α est généralement < 1. La politique LFU dans ce cas s'éloigne largement de LRU et Random. Mais un petit écart est à noter entre les deux politiques de remplacement LRU et Random.

Chapitre 9

Les performances des hiérarchies de caches

9.1 Caractéristiques d'une hiérarchie de caches

Pour évaluer les performances des hiérarchies de caches, il est important de préciser ses caractéristiques. Une hiérarchie de caches est différente d'une autre si un ou plusieurs de ces caractéristiques sont différentes.

9.1.1 Politique de remplacement

La fonction d'un algorithme de remplacement est de choisir un contenu à remplacer dans un cache quand le cache est plein, et qu'un nouveau contenu doit être enregistré. Un algorithme optimal élimine le contenu qui serait le moins utilisé. Pour ceci, l'évolution des popularités des objets devrait être connue. Puisque la variation des popularités des objets se produit sur un temps beaucoup plus grand que le temps nécessaire pour remplir un cache, les prédictions futures peuvent se baser sur le comportement passé ; ce qu'on appelle **principe de localité**. On trouve différentes politiques de remplacement

- **LRU (Least Recently Used)** : cet algorithme remplace le contenu utilisé le moins récemment. Il se base sur le principe de localité temporelle. Un objet très populaire sera demandé plus rapidement qu'un objet moins populaire. L'implémentation de cette politique est simple. Il suffit d'attribuer à chaque objet du catalogue un hash, le hash correspond à une case permettant de renseigner l'adresse de l'objet recherché (si l'objet n'existe pas, l'adresse correspond à NULL). Des pointeurs permettent de relier les objets et de sauvegarder l'ordre des objets.

- **LFU (Least Frequently Used)** : cet algorithme remplace le contenu le moins fréquemment utilisé. Il est optimal pour des popularités fixes, mais les variations des popularités des objets le rend moins efficace. Si les variations des popularités sont lentes, LFU est un bon algorithme de remplacement. De plus, il est facile à implémenter, il suffit d'attribuer à chaque contenu le nombre de fois où il a été demandé. Par contre, son implémentation matérielle serait coûteuse, car un compteur devrait être attribué à chaque contenu.
- **Random** : Cet algorithme remplace un contenu au hasard, il ne demande aucun enregistrement d'information, mais il est moins performant que LRU.
- **MRU** (Most recently used) : Cette politique élimine le contenu correspondant à la donnée la plus récemment demandée. Cette politique s'avère efficace comme politique de deuxième niveau dans le cas d'une hiérarchie de caches.

Gallo et al [41] ont démontré, par simulation et par calcul, que la différence entre les performances observées entre un cache LRU et Random ne sont pas importantes, et surtout pour des popularités d'objets suivant une loi de Zipf avec $\alpha > 1$ ($\alpha = 1.7$). Nous avons simulé des caches LRU et Random avec une loi Zipf (0.6), le constat reste le même. La différence constatée entre probabilité de hit obtenue avec LRU et celle obtenue avec Random est faible. Mais cette différence atteint 16% dans certains cas. Cette différence, même négligeable, peut réduire l'utilisation de la bande passante d'une manière importante.

9.1.2 Les politiques de meta-caching

- **LCE(Leave Copy Everywhere)** : Les objets sont copiés à chaque cache traversé.
- **Fix** [42] : Cette politique consiste à mettre dans le cache un objet selon une probabilité fixe.
- **LCD(Leave Copy Down)** [42] : Copier uniquement dans le cache suivant. Selon Laoutaris, cet algorithme offre les meilleurs résultats dans tous les cas étudiés, et donc parait le plus prometteur de tous.
- **ProbCache** [43] : Un objet est copié dans un cache suivant une probabilité calculée en se basant sur le nombre de sauts traversés. Psaras et al. présentent des résultats remettant en cause LCD comme meilleur algorithme de meta-caching, mais la différence des performances reste petite entre les deux algorithmes. D'autre part, Rossini el al. [44] constatent que, inversement, LCD offre de meilleurs résultats.
- **WAVE** [45] : c'est un algorithme de meta-caching orienté chunk. Il est similaire à LCD, sauf que des variables sont utilisées pour contrôler le stockage des données selon leur popularité. Les objets peu populaires ont peu de chance de traverser tous les caches menant au demandeur. Cet algorithme semble plus complexe que LCD, par opposition à LCD qui stocke naturellement les objets les plus populaires tout près des utilisateurs.

– **Btw** [46] : Cet algorithme se base sur le stockage des données uniquement dans les noeuds pertinents du réseau, c'est-à-dire ayant la probabilité la plus élevée d'aboutir à un hit pour les objets demandés.

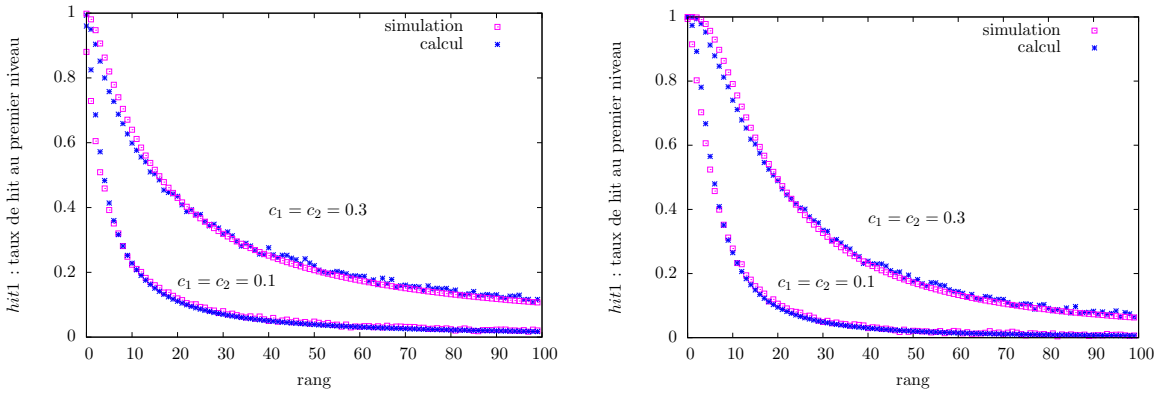
Plusieurs études mettent en valeur la politique LCD à cause de son efficacité constatée par les études comparatives, mais aussi par sa simplicité par rapport aux autres politiques. L'étude récente de Rossini et al. [44] confirme l'efficacité de LCD par rapport aux autres politiques proposées. Dans cette perspective, Laoutaris et al. ont présenté une étude portant sur une hiérarchie LCD [47] ; cette étude commence par une comparaison entre plusieurs politiques de meta-caching. La politique LCD semble être la meilleure avec MCD (Move copy down). Cette étude présente aussi un modèle analytique pour calculer numériquement le taux de hit pour une hiérarchie LCD à deux niveaux. La probabilité de hit au premier niveau d'un objet pour une hiérarchie de caches LCD est estimée à :

$$h_1(i) = exp(\lambda_i * \tau_1 - 1)/exp(\lambda_i \tau_1) + \frac{exp(\lambda_i * \tau_2)}{exp(\lambda_i * \tau_2) - 1}$$

où τ_1 et τ_2 représentent les temps caractéristiques des caches au premier et au deuxième niveau ; la probabilité de hit au premier niveau dépend de la taille du cache au deuxième niveau.

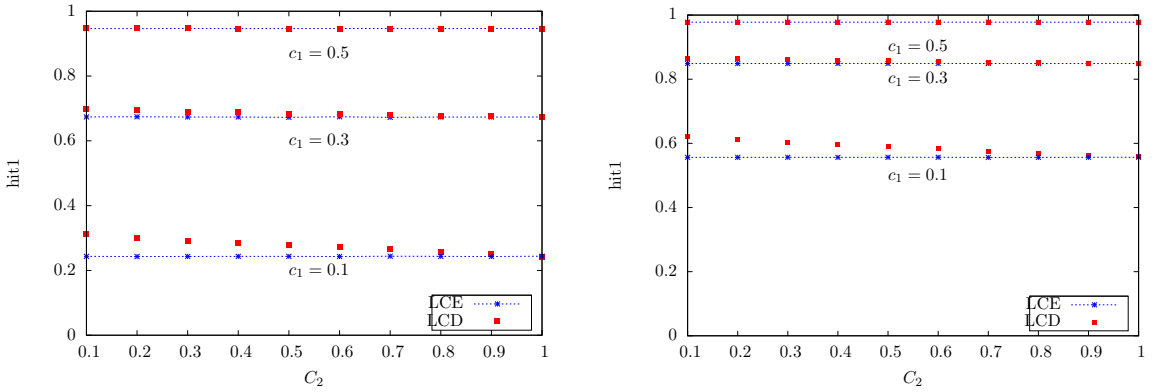
La probabilité de hit au deuxième niveau vient directement de la formule de Che pour LCE, en supposant les arrivées au deuxième niveau indépendantes. $h_2(i) = 1 - exp(-\lambda_i * Miss_1(i) * \tau_2)$ où $Miss_1(i)$ est le taux de miss de l'objet i au premier niveau de caches. Les temps caractéristiques sont calculés en utilisant la formule : $\sum_i h(i) = C$ où C est la taille du cache. Cette équation peut être utilisée au premier niveau comme au deuxième niveau de cache.

Nous obtenons alors deux équations à deux inconnues, τ_1 et τ_2. On résout ces équations en utilisant la méthode de Newton appliquée aux équations à deux inconnues. Pour ce faire, on est amené à calculer l'inverse de la matrice jacobienne pour trouver la solution des équations. Nous comparons le modèle mathématique avec les simulations ; nous observons les résultats dans le graphique 9.1 : Nous avons effectué des simulations comparant la politique LCE et LCD jugée la meilleure de toutes les politiques de métaching. Nous présentons dans les graphes 9.2 les résultats des simulations pour une hiérarchie de caches à deux niveaux, avec 10 serveurs au premier niveau, attachés à un serveur au deuxième niveau. Les résultats montrent un avantage de LCD par rapport à LCE. Cet avantage diminue au fur et à mesure que la taille des caches augmente, réduisant ainsi l'utilité de LCD. LCD permet de réduire le nombre de copies inutiles dans une hiérarchie. La politique MCD, en plus de copier les objets dans le cache inférieur, efface les objets dans les caches supérieurs, mais l'efficacité de cet algorithme reste presque identique à LCD, surtout dans le cas d'une hiérarchie à deux niveaux. Ainsi, LCD paraît la politique la plus simple et la plus rentable. Nous comparons aussi la probabilité de hit globale des hiérarchies afin d'évaluer l'efficacité globale des algorithmes :



(a) Zipf(0.6) (b) Zipf(0.9)

FIGURE 9.1 – Taux de hit calculé par simulation et modèle analytique de Che

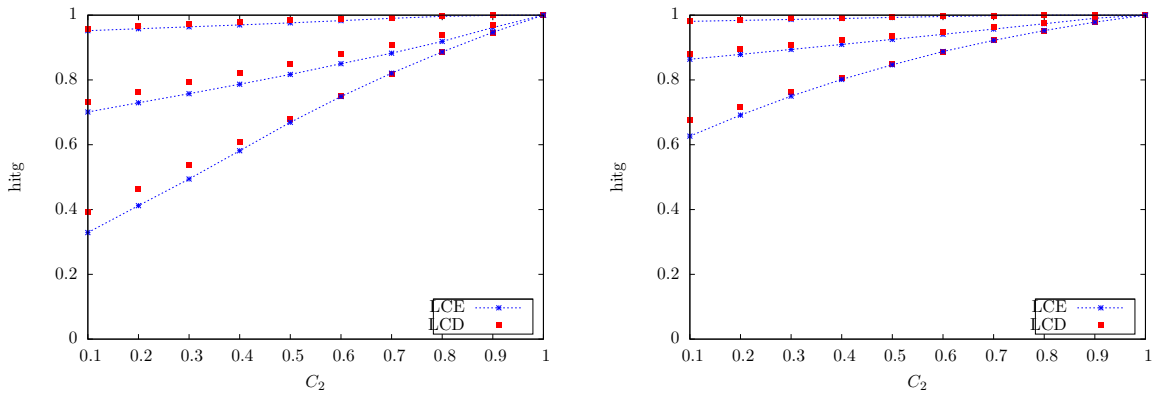


(a) Zipf(0.6) (b) Zipf(0.9)

FIGURE 9.2 – Taux de hit au premier niveau pour une hiérarchie LCE et LCD

9.1.3 Les politiques de forwarding

- **SPR (Shortest Path Routing)** : Cette politique consiste à chercher l'objet en suivant le chemin le plus court vers le serveur d'origine.
- **Flooding** [48] : Envoyer la demande à tous les noeuds et suivre le chemin trouvé. Cette technique est lourde à mettre en place et est très coûteuse.
- **INFORM** [49] : Chaque noeud sauvegarde les valeurs correspondant au temps de latence nécessaire pour arriver à une destination en passant par chaque noeud voisin. Le noeud voisin, menant à destination et offrant le moins de temps pour y arriver, est sélectionné pour récupérer ou envoyer les prochains paquets.
- **CATT** [50] : Le choix du prochain noeud à suivre pour arriver à la donnée est effectué suivant le calcul du paramètre nommé potential value. Ce paramètre



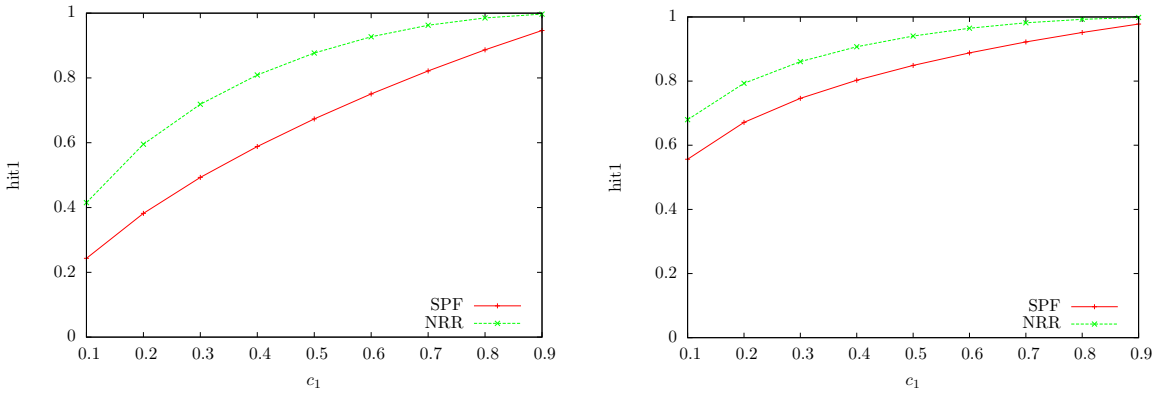
(a) Zipf(0.6) (b) Zipf(0.9)

FIGURE 9.3 – Taux de hit global pour une hiérarchie LCE et LCD à deux niveaux pour les cas de bas en haut $c_1 = 0.1$, $c_1 = 0.3$ et $c1 = 0.5$

peut être évalué selon le nombre de sauts, la situation géographique, ou la qualité de la bande passante séparant le noeud voisin des noeuds contenant les données.

- **NDN** [51] : Cette stratégie est proposée actuellement pour les réseaux orientés contenus. Elle utilise des tables FIB, PIT et CS ; mais les FIB doivent être remplies suivant une autre méthode. Cet algorithme suppose des tables FIB complètes.
- **NRR(Nearest Routing Replica)** : C'est une stratégie idéaliste qui consiste à trouver la copie la plus proche du cache. Cette politique est lourde à mettre en place car il faut maintenir des tables de routage orientées contenu très dynamiques.

La majorité des algorithmes proposés récemment se basent sur le calcul de paramètres de performances menant au cache contenant la donnée. Tout ceci nécessite des mécanismes de signalisation et d'échange de messages de contrôle afin d'identifier périodiquement les chemins menant à tous les objets. Cette opération est non seulement coûteuse, mais aussi pose des problèmes de passage à l'échelle. SPR reste jusqu'à présent l'algorithme le plus utilisé, et le plus simple ne présentant aucun problème de passage à l'échelle. Si la donnée est pertinente et populaire, elle devrait se trouver dans l'un des caches du plus court chemin menant au serveur d'origine. Ce dernier, est statique et invariable, sauf en cas de panne. L'utilisation des chemins secondaires est conditionnée par des problèmes de congestion dans le réseau.
La politique NRR semble être la politique la plus efficace offrant les meilleurs taux de hit. Nous comparons la politique SPF avec NRR afin d'évaluer la différence entre la politique la plus performante, et la politique la plus utilisée pour le routage. Les résultats sont présentés dans le graphique 9.4 :



(a) Zipf(0.6) (b) Zipf(0.9)

FIGURE 9.4 – Taux de hit au premier niveau pour les politiques de forwarding NRR et SPF

9.2 Performances des hiérarchies de caches

Dans cette partie, nous étudions le cas d'une hiérarchie LRU, avec une politique de forwarding se basant sur la recherche dans le serveur d'origine le plus proche. Nous nous limitons au cas de deux niveaux de cache ; Sem Borst [52] affirme qu'il n'y a aucune utilité à utiliser un cache coopératif de plus de deux niveaux.

9.2.1 Généralisation de la formule de Che

Il a été mentionné dans la section précédente que la corrélation des demandes à un deuxième niveau de cache n'a qu'une petite influence sur les probabilités de hit. Nous démontrons, par simulation, que la formule de Che reste valide pour un deuxième niveau de cache à condition d'avoir suffisamment de caches au premier niveau atténuant ainsi la corrélation entre les caches.

On considère un catalogue de M objets ; les demandes arrivent au premier niveau suivant les lois de popularité Zipf(α). On mesure la probabilité de hit global de l'architecture à deux niveaux, constituée de n caches au premier niveau et d'un seul cache au deuxième niveau. Les caches au premier niveau ont la même taille C_1. On pose $c_1 = C_1/M$, la taille normalisée des caches au premier niveau. T_2 est la taille du cache au deuxième niveau et $c_2 = C_2/M$ sa taille normalisée.

On utilise la formule de Che au deuxième niveau de caches, en considérant la popularité $pop_2(i)$ de l'objet i à l'entrée du cache au deuxième niveau :

$$pop_2(i) = p_{miss1}(i) * pop_1(i)$$

où $p_{miss1}(i)$ est la probabilité de miss de l'objet i au premier niveau.

La formule de Che au premier niveau s'applique normalement :

$$p_{miss1}(i) = exp(-pop_1(i) * t_{c1})$$

où t_{c1} est la solution de l'équation

$$C_1 = \sum_n (1 - e^{-pop_1(i)t_{c_1}}).$$

La forumule de Che appliquée au deuxième niveau donne :

$$pmiss_2(i) = exp(-pop_2(i) * t_{c2})$$

où t_{c2} est la solution de l'équation

$$C_2 = \sum_n (1 - e^{-pop_2(i)t_{c_2}}).$$

La probabilité de hit globale $hit_g(i)$ de l'objet i est calculée de :

$$hit_g(i) = hit_1(i) + hit_2(i) - hit_1(i) * hit_2(i)$$

et la probabilité de hit globale de toute l'architecture est :

$$hit_g = \sum pop_1(i) * hit_g(i).$$

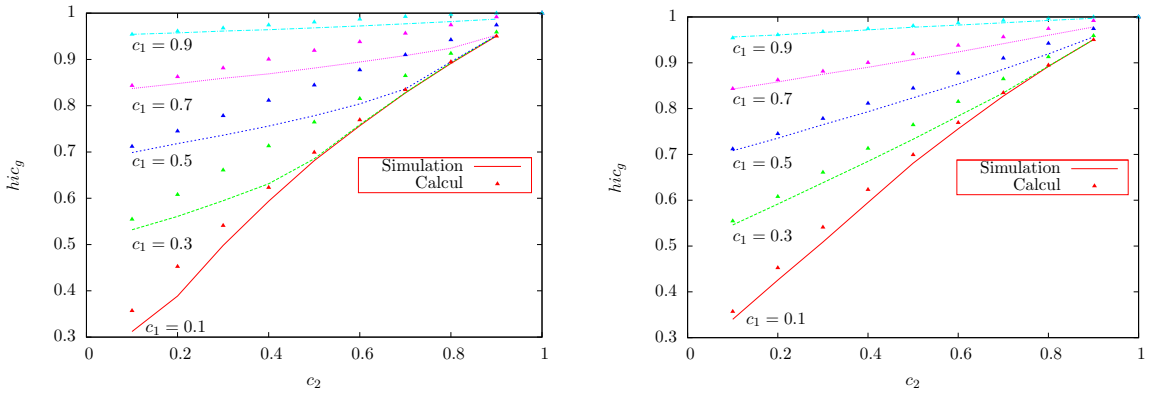
Comme on peut le remarquer, le n n'intervient pas dans le calcul du hit_g, Che propose une formule plus complexe que l'équation initiale incluant le n. La formule étant difficile à exploiter et nos calculs ne semblant pas donner des résultats plus satisfaisants, nous souhaitons savoir si la forumule de Che pour un cache est valable aussi pour plusieurs niveaux de cache. On compare les résultats des calculs avec les résultats des simulations dans les cas suivants :

On remarque que les résultats des simulations sont proches des résultats de calcul pour $n = 5$, l'approximation de Che devient de plus en plus exacte que n augmente. Nous avons vérifié que l'approximation est très précise pour $n \geq 10$.

On compare les taux de hit globaux à chaque niveau de cache pour une hiérarchie de caches à 5 niveaux obtenus par simulation hit_{g_s} et par calcul hit_{g_c}, les tableaux ci dessous représente le taux d'erreur T_e pour des valeurs de n de 2 et 5 (chaque noeud à n fils). Le taux d'erreur est calculé ainsi :

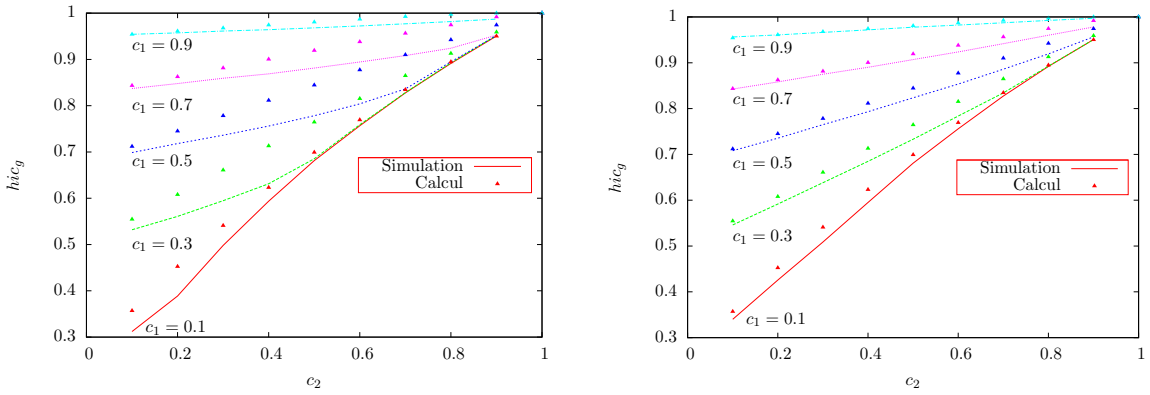
$$T_e = \frac{hit_{gs} - hit_{gc}}{hit_{gs}}$$

On effectue le calcul et la simulation dans le cas de caches de même taille normalisée c(cas CCN),on pose T_g la taille globale $T_g = 5 * c$; les conclusions tirées sont identiques, même si les tailles de cache sont différentes d'un niveau à un autre. Il est clair que



(a) n=2 (b) n=5

FIGURE 9.5 – Probabilités de hit global pour $\alpha = 0.6$ pour un nombre de caches au premier niveau égal à 2 (gauche) et 5 (droite)



(a) n=2 (b) n=5

FIGURE 9.6 – Probabilités de hit global pour $\alpha = 0.8$

la formule de Che reste une excellente approximation, même pour une hiérarchie de caches, et à tous les niveaux de caches. Chaque objet dans CCN est constitué d'un certain nombre de chunks, On peut appliquer l'approximation de Che au niveau des chunks. On considère que la dépendance entre chunks est négligeable.

On considère un catalogue de taille M objets, un cache de taille C ; chaque objet k est de taille T_k et constitué de T_k/s chunks, où s est la taille d'un chunk. La probabilité de hit du chunk o_{ik} de l'objet k est :

$$h(o_{ik}) = 1 - exp(-pop(o_{ik}) * t_c)$$

	level	$M = 10^4$	$M = 10^5$
$T_g = 20\%$	level2	13%	17%
	level3	23%	25%
	level4	28%	29%
	level5	30%	32%
$T_g = 80\%$	level2	14%	17%
	level3	23%	20%
	level4	25%	30%
	level5	28%	21%

TABLE 9.1 – Taux d'erreur pour la formule de Che pour $n = 2$

	level	$M = 10^3$	$M = 10^4$
$T_g = 20\%$	level2	10%	6%
	level3	11%	13%
	level4	11%	9%
	level5	12%	10%
$T_g = 80\%$	level2	1.3%	4%
	level3	3.5%	6.8%
	level4	5.8%	7.3%
	level5	6%	6.9%

TABLE 9.2 – Taux d'erreur pour la formule de Che pour $n = 5$

où t_c est la solution de l'équation :

$$C = \sum_{k=1}^{M} \sum_{i=1}^{T_k/s} (1 - exp(-pop(o_{ik}) * t_c).$$

Les chunks appartenant au même objet ont la même popularité que l'objet :

$$pop(o_{ik}) = pop(k).$$

Donc t_c est calculé avec l'équation :

$$C = \sum_{k=1}^{M} (T_k/s) * (1 - exp(-pop(k) * t_c).$$

9.2.2 Cas d'application : hiérarchie de caches avec mix de flux

Comme souligné précédemment, les données Internet sont non homogènes. Plusieurs catégories de données avec des lois de popularité et des tailles différentes partagent les ressources réseaux. Nous nous limitons à une hiérarchie à deux niveaux. On considère un réseau constitué d'un premier niveau de caches (situés dans les routeurs d'accès)

reliés à un grand cache de deuxième niveau situé au coeur du réseau (voir la figure 9.7). Notons que le deuxième niveau serait réalisé en pratique par un réseau de caches dont les contenus seraient typiquement coordonnés. L'étude d'un tel réseau est l'objectif de nos travaux suivants.

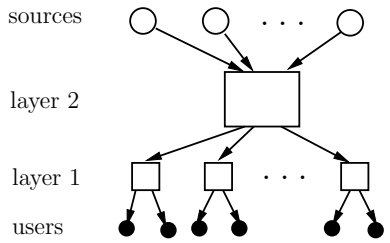


FIGURE 9.7 – Hiérarchie de caches à deux niveaux

On considère que le nombre de routeurs d'accès au premier niveau de caches est large ; donc, les requêtes arrivant au deuxième niveau de cache peuvent être considérées comme indépendantes (i.e., l'IRM s'applique pour le deuxième niveau). La popularité au deuxième niveau de cache est égale à $q'(n) = q(n)(1 - h(n))$; il suffit d'appliquer la formule de Che, en utilisant la nouvelle valeur de popularité, pour trouver la popularité de hit au deuxième niveau $h'(n)$. La figure 9.8 représente la probabilité globale de hit, $\sum_n q(n)(h(n) + (1 - h(n)h'(n))/\sum_n q(n)$, en fonction des tailles de cache au premier et au deuxième niveau. La loi de popularité est de Zipf et la figure montre l'impact de son paramétre α.

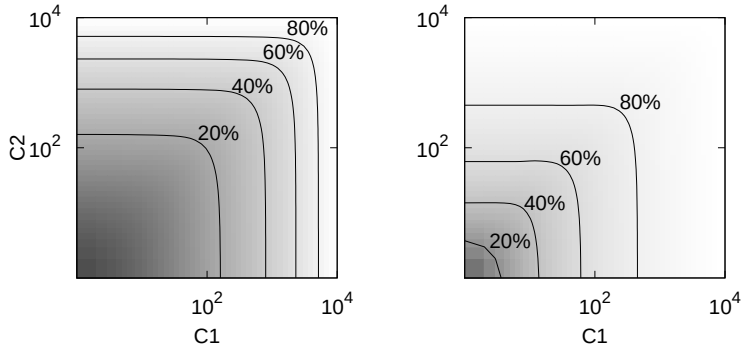


FIGURE 9.8 – Taux de hit(%) en fonction de la taille des caches dans les niveaux 1 et 2 : à gauche, $\alpha = .8, N = 10^4$; à droite, $\alpha = 1.2$.

La même approche a été appliquée dans le cas d'un mix de trafic, permettant ainsi de quantifier les tailles de cache aux deux niveaux pour une probabilité de hit cible.

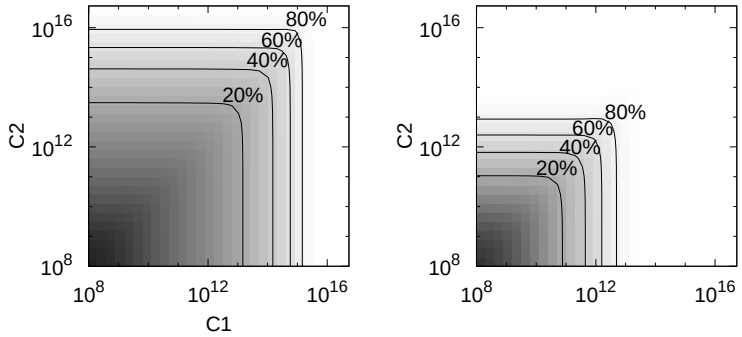


FIGURE 9.9 – Taux de hit(%) en fonction de la taille des caches dans les niveaux 1 et 2 : à gauche trafic UGC, $\alpha = .8$; à droite, VoD $\alpha = 1.2, N = 10^4$.

Les caches au premier niveau sont efficaces pour le trafic VoD voir figure 9.9. En effet, avec un cache à 10^{12}, la probabilité de hit du trafic VoD est de 80%. Avec un cache de même taille, on ne peut dépasser 20% de probabilité de hit pour les autres types de trafic : UGC, fichiers partagés et web. Il serait donc nécessaire de choisir des tailles de caches assez grandes. Les tailles de cache au premier niveau sont petites permettant ainsi leur insertion à un routeur d'accès. Les VoD peuvent ainsi être stockées au premier niveau, contrairement aux autres types de données.

Dans la table ci-dessous, nous évaluons la bande passante conservée par une succession de deux caches ; le premier situé au premier niveau est de taille 1 TB , et le deuxième situé au deuxième niveau est de taille 100 TB. L'évaluation est effectuée dans le cas d'un partage normal entre contenus, et dans le cas d'un premier niveau réservé au VoD. Les résultats présentés correspondent au trafic Mix de 2011 et 2015.
On remarque qu'un stockage exclusif du VoD au premier niveau améliore la pro-

	Zipf VoD(α)	niveau1	Réduction bande passante au niveau1	Réduction bande passante au niveau1 et 2
2011	0.8	partagé	17%	50%
		VoD	23%	58%
	1.2	partagé	24%	50%
		VoD	23%	58%
2015	0.8	partagé	27%	59 %
		VoD	37%	61%
	1.2	partagé	36%	59%
		VoD	37%	61%

TABLE 9.3 – Réduction en bande passante pour C_1=1TB et C_2=100TB

babilité du hit au premier niveau de cache ; ceci est plus significative pour une loi Zipf(0.8). Dans tous les cas, un stockage discriminatoire des VoD au premier niveau

améliore le taux de hit global résultant des deux caches, que ce soit pour une loi Zipf(0.8) ou Zipf(1.2) ; la différence devient moins importante pour le trafic futur en 2015.

Chapitre 10

Conclusion

Dans ce chapitre, nous avons évalué la nature, la quantité et la popularité des objets échangés sur Internet, le trafic étant presque majoritairement du contenu distribué. Nous avons par la suite évalué les performances d'un cache LRU ; pour ce faire nous avons testé la formule de Che. Cette dernière est facile à utiliser numériquement, valable pour toute valeur de α de la loi de Zipf et pour plusieurs autres lois de popularité. Nous avons expliqué les raisons de son exactitude, en se basant sur la démonstration présentée par Fricker et al. [40]. Nous avons aussi exploré le cas d'un cache Random, et comparé ses performances à celles d'un cache LRU. Nous avons décrit les caractéristiques d'une hiérarchie de caches, et comparé les perfomances des politiques de métacaching et forwarding optimales avec des politiques d'usage. Nous avons constaté à travers un exemple en se basant sur des données représentant l'échange actuel sur Internet que pour des lois de popularité Zipf avec un paramètre α ¡1 il faut choisir des taille de cache très grands pour réduire d'une manière significative l'utilisation de la bande passante, ce qui est techniquement difficile à mettre en place au niveau d'un routeur, ce qui nous mène à penser qu'un déployment au niveau applicatif est plutot convenable. Dans la partie suivante nous comparons les couts des hierarchies distribués à la CCN et des hierarchie centralisées à la CDN qui nous paraissent techniquement plus convenables.

Troisième partie

Coûts d'une hiérarchie de caches

Chapitre 11

Introduction

11.1 Problématique

Dans la partie précédente, nous avons proposé un stockage différencié pour améliorer la probabilité de hit de la hiérarchie. Or, en stockant la majorité des contenus au deuxième niveau, la consommation de la bande passante est plus élevée. Pour les opérateurs, deux critères importants leur permettent de prendre des décisions pour le stockage de données : les coûts et la difficulté de mise en oeuvre. Puisque les réseaux de caches comme CCN sont distribués, il faut prendre en compte, en plus du coût de la mémoire, le coût de la bande passante. Plus l'opérateur investit en mémoire, moins il investit en bande passante et vice versa. L'opérateur est amené à déterminer un tradeoff optimal entre bande passante et caches.

Déployés depuis des années, les CDN deviennent un acteur principal gérant le contenu des sites les plus connus et utilisés à travers le monde. Les CDN, contrairement aux CCN, sont des hiérarchies centralisées et non distribuées. Entre une technologie déjà expérimentée et une technologie en cours de recherche, le choix de l'opérateur est vite fait ; les CDN sont déjà déployés et utilisés, ne nécessitant aucune mise à jour au niveau des routeurs ou des protocoles réseau. Le seul enjeu pouvant encourager les opérateurs à opter pour une architecture distribuée est le coût. Le travail de Ghodsi et al. [5] n'apporte pas cet espoir pour l'avenir des réseaux CCN ; cet article vient, en plein milieu de notre travail de recherche, remettre en cause l'utilisation des CCN à l'avenir.

Les arguments avancés par Ghodsi et al. sont :

– La difficulté de changer tout le modèle Internet de l'orienté IP vers l'orienté contenu. La convergence vers un réseau orienté contenu nécessite un changement au niveau des routeurs et des protocoles. Ceci nécessiterait beaucoup de temps, compte tenu du temps d'attente avant la mise en place d'une mise à

jour mineure telle que l'adressage IPv6. Sans compter le temps nécessaire pour l'implémentation de PKI assurant la sécurité des données.

– Pour des caches de grande taille, il a été prouvé que la coopération n'apporte que très peu de gain ; donc l'utilité de distribuer la mémoire sur tous les routeurs n'est pas forcément bénéfique.

En plus de ces arguments, notre travail présenté dans [40] montre qu'une réduction significative de bande passante nécessite l'utilisation de caches de très grande taille, dépassant largement la taille de la mémoire pouvant être ajoutée à un routeur [53].

Notre problème revient à évaluer les coûts d'une architecture de cache distribuée : qu'est ce qui coûterait plus cher, mettre des petits caches partout ou de grands caches au premier niveau ? Ou plus exactement dans une architecture en arbre, comment faut-il choisir la taille de cache dans les différents niveaux pour avoir un taux de hit cible avec le moindre coût ?

11.2 Etat de l'art

Certaines études abordent le problème de l'optimisation en fixant un ou plusieurs paramètres. Par exemple, Dario Rossi et Giuseppe Rossini [54] se sont déjà penchés sur ce problème. Ils ont conclu que l'utilisation de caches de tailles différentes ne fait augmenter le gain que de 2,5% dans le meilleur des cas, et qu'il vaut mieux utiliser la même taille de cache, vu le gain faible et la détérioration de la rapidité d'un cache quand sa taille augmente. Dans cette étude, la taille globale de caches a été fixée. Le coût de caching est donc fixé et le facteur de performance est le taux de hit. Le coût de bande passante n'a pas été pris en compte. Une configuration est préférable à une autre si sa probabilité de hit est meilleure. On voit que ceci n'est pas suffisant. Par ailleurs, le paramètre α de la loi Zipf de popularité est supposé supérieur à 1, ce qui n'est pas conforme aux mesures.

Sem Borst et al. [52] calculent les coûts de bande passante et proposent un algorithme de coopération pour le placement des données dans une hiérarchie de caches afin de minimiser ce coût. Le coût de la mémoire n'est pas pris en considération, donc le tradeoff mémoire/bande passante n'est pas étudié.

Kangasharju [55] suppose que la localisation des caches et la capacité sont fixes, et dans [47] est exploré le problème d'optimisation avec une taille fixe de cache global comme contrainte.

Notre vision des choses est différente, nous pensons que le coût global inclut le coût de la bande passante et le coût de la mémoire. C'est un critère de comparaison fiable entre architectures.

D'autres études cherchent le tradeoff mémoire/bande passante, comme par exemple Nussbaumer et al. [56]. Ils adoptent une évaluation de coût similaire à la nôtre, en prenant en compte les coûts de la bande passante et de la mémoire pour une hiérarchie

de caches en arbre. Cependant, leurs résultats ne sont pas applicables pour nous. Nous rencontrons le même problème avec l'article [57] qui n'offre aucun résultat numérique exploitable.

Il est clair que la taille des caches détermine le coût de la mémoire qui est croissant en fonction de la taille. Par contre le coût de la bande passante devient plus petit car le taux de hit augmente avec la taille. Notre objectif est de déterminer la hiérarchie de cache optimale, que ce soit une architecture avec, ou sans coopération de caches. On fixe la probabilité de hit global de l'architecture à une probabilité de hit cible.

Avec une hiérarchie de caches, il est toujours possible d'améliorer les gains en utilisant la coopération, comme c'était le cas pour les proxies, et donc rendre la distribution de caches rentable par rapport à une hiérarchie centralisée. Pourtant, Wolman et al. [58] ont démontré, par simulation sur une large population de 10^7 ou 10^8 objets que la coopération n'apporte que très peu de gain ; ce qui remet en question l'utilité de mettre en place un lourd mécanisme de coopération et d'échange de messages entre serveurs. Pour une population réduite, utiliser un seul proxy revient moins cher et est aussi efficace que plusieurs caches coopératifs.

Plus récemment, Fayazbakhsh et al. [59] ont constaté que le stockage à l'edge offre des performances plutôt bonnes en termes de temps de latence, de congestion, et de charge du serveur d'origine. De plus, il offre un écart de 17% par rapport à une architecture CCN presque optimale, ce que les auteurs considèrent comme minime, et compensable en plaçant plus de mémoire à l'edge.

L'article [44] a remis en question cette étude par une contre-étude, en montrant que le fait de coupler une stratégie de forwarding optimale (chercher l'objet dans le cache le plus proche contenant l'objet recherché) avec une politique de méta-caching optimale (LCD pour Leave a Copy Down) augmente considérablement la marge entre un caching à l'edge et un caching distribué à la CCN, ceci en considérant une popularité Zipf avec $\alpha = 1$.

Wang et al. [60] traitent le problème d'allocation de la mémoire et cherchent à trouver l'optimum. Le raisonnement va dans le même sens : l'optimum est obtenu à travers une distribution non équitable des tailles de caches dans une hiérarchie ; cette distribution dépend de la popularité des objets, et de la taille du catalogue.

En somme, un caching exclusif à l'edge est une solution contestée du fait qu'elle n'a pas été comparée à une solution ICN optimale avec une stratégie de forwarding, et de meta-caching optimales. Dans tous les cas, le problème que nous étudions est différent du problème traité dans les articles précédemment cités. En général, les chercheurs fixent une taille de cache global, ou par cache individuel, et cherchent un optimum pour la probabilité de hit globale, ou un optimum selon le nombre moyen de sauts traversés par les requêtes pour trouver l'objet recherché, ce qui mesure implicitement la consommation de la bande passante. Le fait de fixer la taille de cache élimine des

solutions possibles pour les optima, sachant que l'optimum absolu pour les utilisateurs est un stockage entier du catalogue à l'edge évitant ainsi tout déplacement vers les serveurs d'origine.

Entre la taille nécessaire pour assurer le stockage et la consommation de bande passante, le choix des opérateurs devrait se baser sur les coûts, sachant que les coûts de la mémoire décroissent très vite. Un stockage à l'edge éliminerait tous les problèmes liés au routage orienté contenu qui nous paraissent lourdes et posent des problèmes de passage à l'échelle. L'utilisateur demande un objet de son edge. Si le routeur d'accès ne détient pas l'objet, alors la requête est redirigée vers le serveur d'origine. Le tableau 11.1 résume quelques études ayant traité la problématique de l'optimum.

Référence	Contraintes	Optimums
Fayazbakhsh et al. [59]	Les caches ont la même taille	Peu d'avantages à cacher partout dans le réseau, un caching à l'edge est bénéfique
Rossini et al. [44]	Les caches ont la même taille	Avec une politique de meta caching LCD (Leave a copy down) et une stratégie de forwarding menant au cache le plus proche(NRR) contenant l'objet, nous obtenons un écart de 4% en nombre de sauts par rapport à un couplage LCD et une politique de forwarding SPF menant au serveur d'origine le plus proche.
Borst et al. [52]	Taille de caches fixes, recherche de l'optimum en terme de bande passante	L'optimum inter-niveau correspond à un stockage des objets les plus populaires dans toutes les feuilles de l'arbre, et les moins populaires dans le cache supérieur.

TABLE 11.1 – Comparaison des quelques études des coûts des hiérarchies de caches

Chapitre 12

Coût d'une hiérarchie de caches à deux niveaux

Dans ce chapitre, nous nous intéressons aux coûts d'une hiérarchie de caches. Nous souhaitons répondre à la question : est-ce bénéfique d'utiliser des caches distribués plutôt que des caches centralisés ? Nous pensons que c'est la différence de base entre les réseaux CCN et les CDN.

Afin de voir clairement la différence des deux solutions, nous commençons par une hiérarchie à deux niveaux uniquement. Notre hiérarchie est représentée à la figure 12 :

Le trafic généré par les demandes des utilisateurs est de A en bit/s, le requêtes

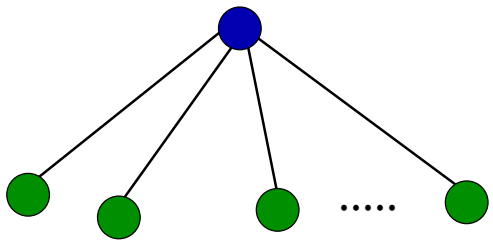


FIGURE 12.1 – architecture à deux niveaux

proviennent de n noeuds d'accès au premier niveau. Tous les caches au premier niveau ont la même taille T_1. Le cache de deuxième niveau est de taille T_2.

On cherche la structure optimale avec le minimum de coût. Les tailles T_1 et T_2 varient afin d'obtenir à chaque fois une structure différente, mais chaque structure devrait réaliser un taux de hit fixé à une valeur cible (afin de pouvoir comparer les coûts

des architectures). Donc pour chaque valeur de T_1, il existe une seule valeur de T_2 vérifiant un taux de hit global fixe.

12.1 Différence des coûts entre structures

Le coût d'une architecture est égal à la somme des coûts de la bande passante, le coût de la mémoire, et le coût de gestion au niveau des caches.

- Le coût de la bande passante C_b est proportionnel (on introduira une constante k_b) au trafic global généré entre le premier et le deuxième niveau. Le coût d'accès est le même pour toutes les architectures, et donc il s'annule en différence

$$C_b = k_b \times Trafic$$

.

- Le coût de la mémoire C_m est proportionnel à la taille des caches

$$C_m = k_m \times \text{Taille de la mémoire}$$

.

- Le coût de gestion au niveau des caches C_g est proportionnel au trafic, soit

$$C_g = k_s \times Trafic.$$

La différence de coût entre deux architectures ayant le même taux de hit global est :

$$\delta_{\text{cout}}(T_1, T_1') = (p_{miss_1} - p'_{miss_1})(k_b + k_s)A + (n(T_1 - T_1') + (T_2 - T_2'))k_m$$

où

- p_{miss_1} est le taux de miss de la première architecture,
- $p_{miss'_1}$ est le taux de miss de la deuxième architecture,
- T_1 (T_2) est la taille des caches au premier (deuxième) niveau de la première architecture
- et T_1' (T_2') est la taille des caches au premier (deuxième) niveau de la deuxième architecture.

Les deux architectures ont le même taux de hit global et le même nombre de caches au premier niveau.

12.2 Estimation numérique des coûts

Nous avons besoin de quelques estimations des coûts unitaires k_b, k_m et k_s.

- k_b : On estime k_b à 15 \$ par Mbps ; c'est une valeur non publiée mais estimée par quelques sources sur le web, et par des échanges privés avec un opérateur. Ce coût unitaire couvre le coût du transport et des routeurs.
- k_m : On estime le coût unitaire de la mémoire à k_m=0.15 \$ par Gigabyte. Cette estimation est dérivée des fournisseurs de cloud comme Amazon.
- k_s : Le coût unitaire de gestion k_s est estimé à 10 c par Mbps. Cette observation est tirée des charges de téléchargement des offres Cloud.

La valeur de k_s est donc négligeable par rapport à la valeur de k_b de sorte que la différence des coûts entre architectures devient :

$$\Delta_{\text{cout}}(T_1, T_1') = (p_{miss_1} - p'_{miss_1})k_b A + (n(T_1 - T_1') + (T_2 - T_2'))k_m.$$

Etudier la différence de coûts revient à étudier la fonction δ définie par :

$$\Delta(T_1) = p_{miss}(T_1)k_b A + (nT_1 + T_2(T_1))k_m.$$

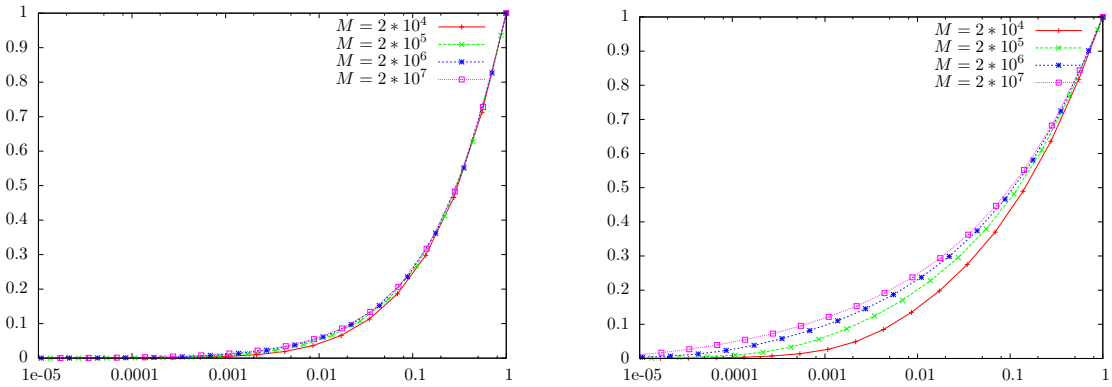
Notre objectif est de déterminer les tailles T_1 (et donc T_2 puisque le taux de hit global est fixé) correspondant au minimum de la fonction Δ. Le trafic à l'accès est de l'ordre de $A = 1$ Tbp, la taille moyenne de chaque objet est de 1 GB et la taille du catalogue est estimée à $M = 2 \times 10^6$ objets.

De plus, notons que la probabilité de hit dépend du rapport entre la taille du cache et le catalogue. Pour vérifier ceci, on considère un cache de taille T et un catalogue de taille M. On trace sur la figure 12.2 la probabilité de hit en fonction de T/M pour différentes valeurs de M et deux lois de popularité, Zipf(0.6) et Zipf(0.8).

On observe que la probabilité de hit pour une popularité Zipf(0.6) est presque la même pour des tailles de catalogue différentes. Pour le cas Zipf(0.8), la probabilité de hit est légèrement plus faible pour un catalogue de taille 2×10^4 mais les probabilités de hit se rapprochent pour les grandes tailles de catalogue. Dans tous les cas, à partir d'une taille de cache normalisée de 10%, les valeurs sont très rapprochées. Puisque les tailles de catalogues sont grandes en réalité, il est clair que la probabilité de hit ne dépend en pratique que du rapport entre la taille du cache et du catalogue.

On note c la taille normalisée du cache $c = T/M$, $\bar{c}$ la taille normalisée du cache au deuxième niveau et $h(c)$ la probabilité de hit au premier niveau de cache. La différence de coût devient :

$$\delta(c) = (1 - h(c))k_b A + M(nc + \bar{c})k_m.$$



(a) Popularité Zipf α=0.6

(b) Popularité Zipf α=0.8

FIGURE 12.2 – Taux de hit en fonction des tailles de caches normalisées T/M

12.2.1 Coût normalisé

Dans un premier temps, on considère que l'opérateur souhaite éviter de chercher les données en dehors de son réseau, et donc la probabilité de hit cible est égale à 1. Dans ce cas $\bar{c} = 0$ (est une différence de tailles de caches au deuxième niveau qui s'annule dans ce cas).

On pose

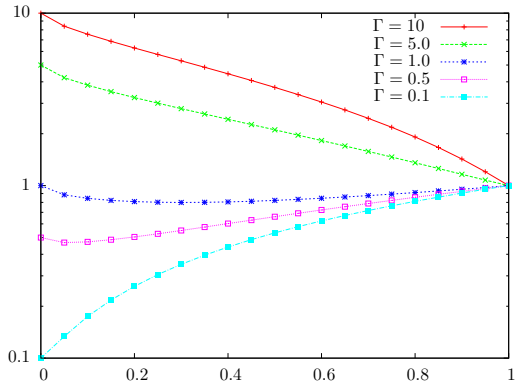
$$\Gamma = \frac{Ak_b}{Mnk_m},$$

le rapport entre le coût maximal de bande passante Ak_b et le coût maximal de mémoire Mnk_m. Le tradeoff entre mémoire et bande passante est visible sur l'équation. On note $\Gamma_{nominal}$ la valeur de Γ correspondante aux valeurs citées dans le paragraphe précédent,

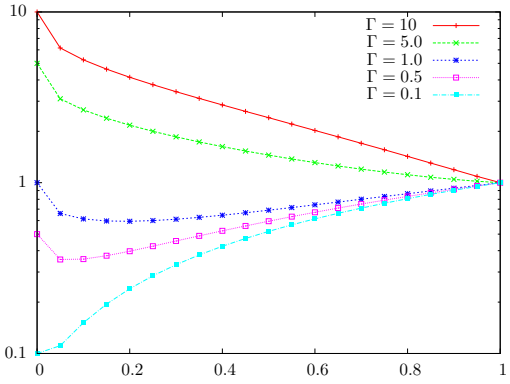
$$\Gamma_{nominal} = \frac{10^{12} \times 15 \times 10^{-6}}{0.15 \times 10^{-9} \times 100 \times 2 \times 10^6 \times 10^9} = 0.5.$$

Avec ce choix taux de hit cible, la différence de coût normalisé peut s'exprimer : $\delta(c) = \Gamma(1 - h(c)) + c$. Cette fonction est présenté à la figure 12.3. On observe que, pour Γ=0.5 (c'est-à-dire la valeur nominale), la valeur optimale du coût correspond à une taille de cache au premier niveau plutôt petite (inférieure à 10% du catalogue) ; plus Γ devient petit, plus on a intérêt à tout cacher au deuxième niveau. Plus la valeur Γ augmente, plus la solution optimale correspondrait à un grand cache au premier niveau. Ces observations sont valables pour les deux valeurs de α comme le montre la figure 12.4.

On remarque que le choix d'une taille de cache au premier niveau égale à la taille du catalogue offre une solution optimale pour une valeur de $\Gamma > 2$ pour $\alpha = 0.6$, et de $\Gamma > 3.8$ pour $\alpha = 0.8$. Plus le cache est efficace (paramètre α plus grand ou politique

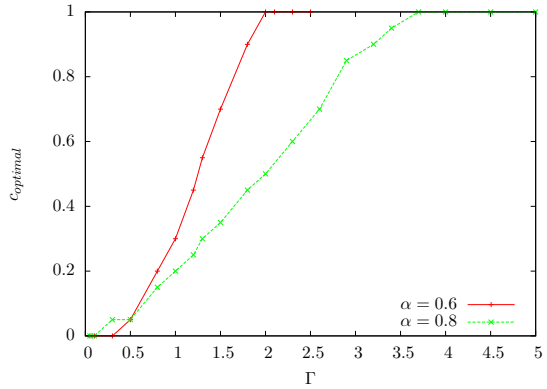


(a) Popularité Zipf α=0.6

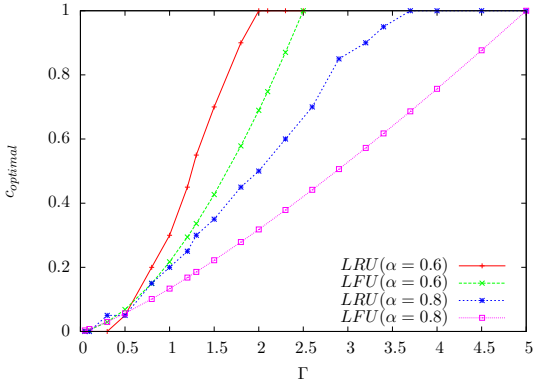


(b) Popularité Zipf α=0.8

FIGURE 12.3 – Coûts normalisés en fonction des tailles de caches normalisés au premier niveau



(a) Popularités différentes



(b) Politiques de remplacement différentes

FIGURE 12.4 – Taille de cache normalisée optimale en fonction de Γ

de remplacement plus efficace), moins il est utile de stocker des données au premier niveau de cache. Mais, même pour un cache LFU et un $\alpha = 0.8$, la solution optimale correspond à un stockage complet du catalogue à partir de $\Gamma = 5$.

La solution optimale actuelle pour le Γ nominal correspond à une valeur de moins de 10 % du catalogue ; mais les valeurs des coûts sont si rapprochées pour $\Gamma = 1$ que le choix d'un c a une valeur égale au catalogue semble meilleur car l'utilisateur peut profiter d'une latence plus faible.

La valeur de k_c diminue de 40% chaque année[1], k_b diminue de 20% chaque année, le

[1]http ://www.jcmit.com/memoryprice.htm

trafic T augmente de 40% chaque année[2] et le catalogue augmente d'à peu près 50%. Le Γ tend alors à augmenter au fil du temps, et peut devenir cinq fois plus grand que le Γ nominal dans 6 ans.

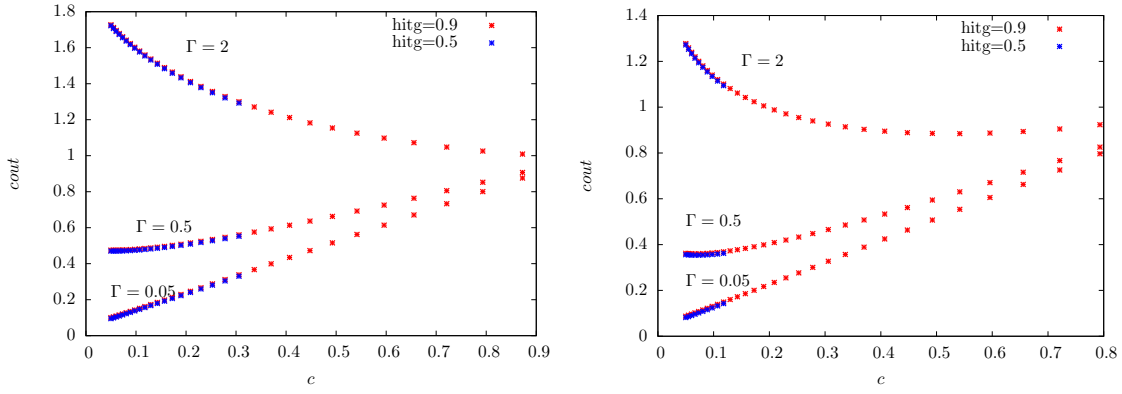
Cependant, les valeurs des coûts varient selon la région géographique [3] et des éléments inattendus peuvent augmenter le prix des mémoires. Le tradeoff reste quand même compliqué à déterminer car il dépend de l'emplacement géographique, et des lois de popularité.

12.2.2 Solution optimale en fonction du taux de hit global

Nous avons considéré la probabilité de hit global égale à 1. Or, pour différentes raisons, le fournisseur de contenu peut manquer de moyens pour payer le prix optimal. La solution consiste à gérer une partie du catalogue (ce qui correspond à la partie la plus populaire) et de confier le stockage des autres contenus à un serveur distant. Le nombre de caches au premier niveau ne change rien à la probabilité de hit global, on peut utiliser la formule de Che à deux niveaux de caches. Donc

$$\delta(c) = \Gamma(1 - h(c)) + c + \bar{c}/n).$$

On obtient les résultats présentés dans la figure 12.5 avec des probabilités de hit global de 0.5 et 0.9.



(a) Loi Zipf(0.6)

(b) Loi Zipf(0.8)

FIGURE 12.5 – Coûts pour différentes valeurs de Γ et de probabilités de hit

On remarque que la courbe des différences de coûts pour une probabilité de hit de 50% coïncide avec la courbe des coûts pour une probabilité de hit de 90%. Ceci s'explique

[2] http ://www.networkworld.com/news/tech/2012/041012-ethernet-alliance-258118.html ?page=1

[3] http ://www.zdnet.fr/actualites/le-prix-du-transit-ip-baisse-encore-partout-dans-le-monde-39775196.htm

peut-être par la valeur de n (n=100) qui implique que la valeur de $\bar{c}/n$ reste petite et n'influence pas la valeur des coûts. Ceci peut mener à nous demander à quel niveau il faut placer les caches dans un réseau.

12.3 Exemple : coûts des torrents

Un moteur de recherche comme *mininova.org* offre un ensemble de torrents. Dan et al. [61] ont analysé des données torrents de cette source et partagé leurs données avec nous. Un premier fichier de données contient 2.9×10^6 torrents actifs et donne le nombre de leechers au moment de la capture. Nous avons conservé 1.6×10^6 torrents correspondant à au moins un leecher actif au moment de la capture. On considère un torrent de taille s et un nombre de leechers instantané de l. En utilisant la loi de Little, et en considérant la durée de téléchargement proportionnelle à la taille s, la fréquence d'arrivée du torrent est :

$$\lambda = \frac{\text{nombre moyen de torrents}}{\text{temps de téléchargement}}.$$

La fréquence d'arrivée des torrents est donc proportionnelle à l/s. Un deuxième fichier correspondant aux tailles des torrents nous a été fourni.

La popularité de chaque chunk est égale à la popularité de son torrent. La formule de Che est plus facile à utiliser au niveau chunk qu'au niveau torrent car les torrents ont des tailles différentes, alors que les chunks ont la même taille. La popularité des chunks des torrents déduite des traces est représentée à la figure 12.6.

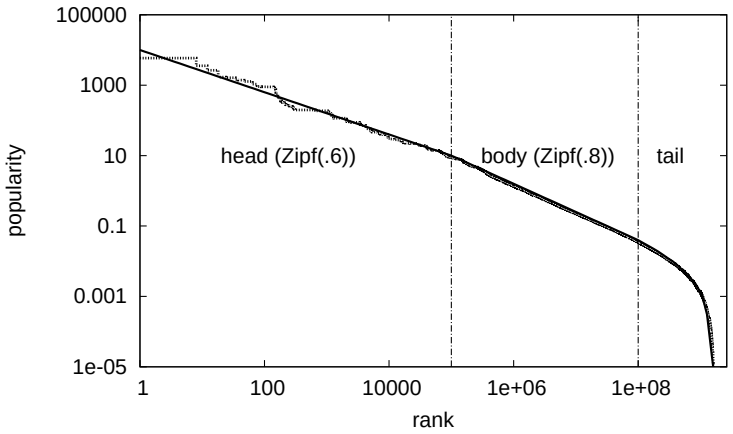
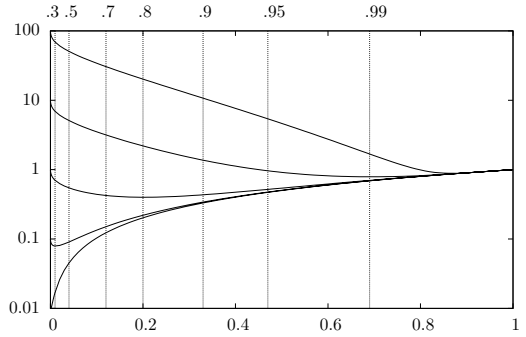
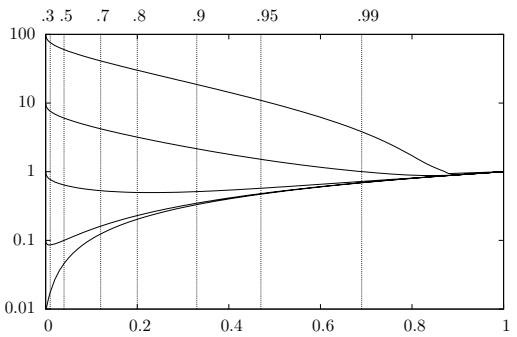


FIGURE 12.6 – Popularité des chunks pour les torrents



(a) coûts normalisés pour un coût de bande passante linéaire



(b) coût normalisés pour un coût de bande passante non linéaire

FIGURE 12.7 – Coûts normalisés pour différentes valeur de Γ

Les courbes présentées dans la figure 12.7 représentent les coûts normalisés dans le cas d'une probabilité de hit globale de 1 pour les torrents. La figure 12.7(b) représente le cas d'un coût de bande passante non linéaire $T^{\cdot 75}k_b$, ce qui permet d'évaluer l'impact d'économie d'échelle. On remarque que les observations sont les mêmes : pour un $\Gamma = 10$, la solution optimale correspond à un stockage complet au premier niveau ; de même pour $\Gamma = 1$ la courbe est presque plate, impliquant qu'un stockage au premier niveau semble plutôt bénéfique.

Chapitre 13

Coopération de caches

Un grand nombre de publications met en avant l'avantage qu'on peut tirer de la distribution de caches et de leur coopération. Mais ces publications ne prennent pas en compte le coût des échanges entre les caches et donc le coût de la bande passante qui est typiquement plus important que le coût de la mémoire. Dans cette section, nous nous intéressons à l'évaluation du tradeoff bande passante/ mémoire, dans le cas de coopération de caches.

13.1 Load sharing

Li et Simon [62] proposent la division du catalogue en P partitions. Chaque cache peut stocker les éléments d'une partition de taille égale à M/P où M est la taille du catalogue.

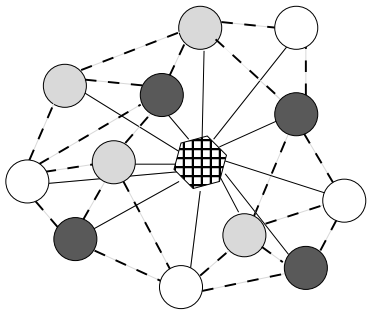


FIGURE 13.1 – Réseau de caches P=3 et S=12

Pour évaluer les coûts de cette architecture, on introduit un nouveau facteur k'_b représentant le coût unitaire des liens entre caches du premier niveau. Une proportion

$1/P$ des requêtes arrivant à chaque cache correspondent au sous-catalogue attribué au cache, les $(P-1)/P$ requêtes restantes étant redirigées vers les autres caches. La consommation de bande passante au premier niveau est donc estimée à :

$$nk'_b \frac{P-1}{P}.$$

Le coût de la bande passante résultante du taux de miss au premier niveau est estimée à :

$$nk_b miss(C)$$

où $miss(C)$ est la probabilité de miss au premier niveau pour cette architecture. Elle correspond au taux de miss d'un cache LRU de taille C pour un catalogue de taille M/P. On pose $c = CP/M$ la taille normalisée des caches pour cette architecture. On écrit $miss(c) = miss(CP/M)$ où $miss(c)$ correspond à la probabilité de miss d'un cache LRU de taille PC et un catalogue de taille M. Tout calcul fait :

$$\delta_{LS}(c) = \delta(c) + (1 - \frac{1}{P})(\Gamma \frac{k'_b}{k_b} - c).$$

Le partitionnement du catalogue est bénéfique dans le cas où $\Gamma \frac{k'_b}{k_b} < c$. Si, par exemple, $\Gamma = 4$ il faut que $\frac{k'_b}{k_b} < c/4 \leq 1/4$. Or, il serait surprenant que le coût d'un lien entre caches de premier niveau soit 4 fois inférieur au coût d'un lien entre le premier et le deuxième niveau de caches, ce qui suggère que ce type de partage de charge n'est guère utile.

13.2 Caches coopératifs

Ni et Tsang [63] proposent un algorithme de caches coopératifs basé sur le load sharing. Chaque cache est responsable d'une partition du catalogue. Toute requête destinée à un cache sera d'abord recherchée dans le cache d'accueil, avant de s'adresser au cache responsable le plus proche. On désigne $q(i)$ la popularité normalisée de l'objet i. Pour un cache du premier niveau, la popularité de chaque chunk appartenant à la partition dont il est responsable est $q'(n) = q(n)(1 + (P-1)e^{-q'(n)t_c})$. En adaptant la formule de Che, on trouve t_c en utilisant la formule (13.1) :

$$\sum_{n=1}^{N} \frac{1}{P}(1 - e^{-q'(n)t_c}) + (1 - \frac{1}{P})(1 - e^{-q(n)t_c}) = C \tag{13.1}$$

On considère un cache k au premier niveau. On note R la fonction déterminant l'identité de la partition du catalogue dont il est responsable : $R(k) = l$ veut dire que le cache k est responsable de la partition M_l. On note A_k l'ensemble de caches au premier niveau sauf le cache k. On observe les échanges entre caches illustrés par la figure 13.2.

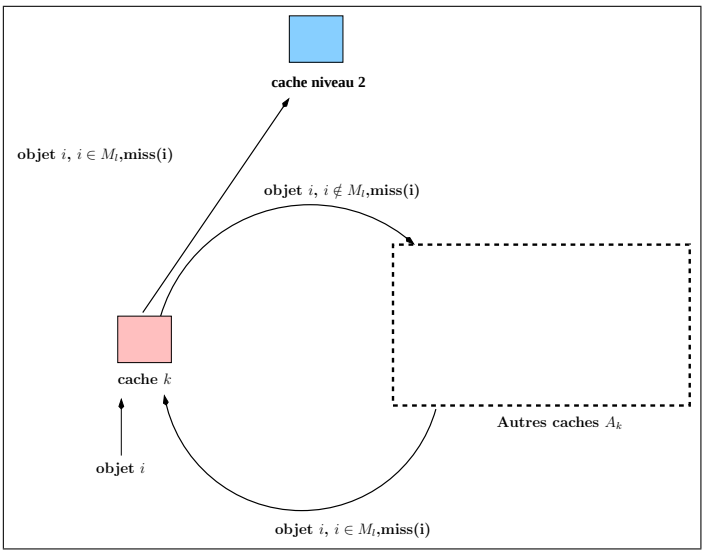


FIGURE 13.2 – illustration des différents échanges pour une hiérarchie coopérative

En utilisant l'équation (13.1) pour trouver t_c, nous pouvons calculer la probabilité de miss au niveau du cache k pour tout objet i

$$miss_k(i) = \begin{cases} \exp(-q(i)t_c) \ ; i \notin M_l \\ \exp(-q'(i)t_c) \ ; i \in M_l. \end{cases}$$

Les caches gèrent le trafic arrivant des noeuds d'accès. On note par θ' la probabilité de hit global de ce trafic :

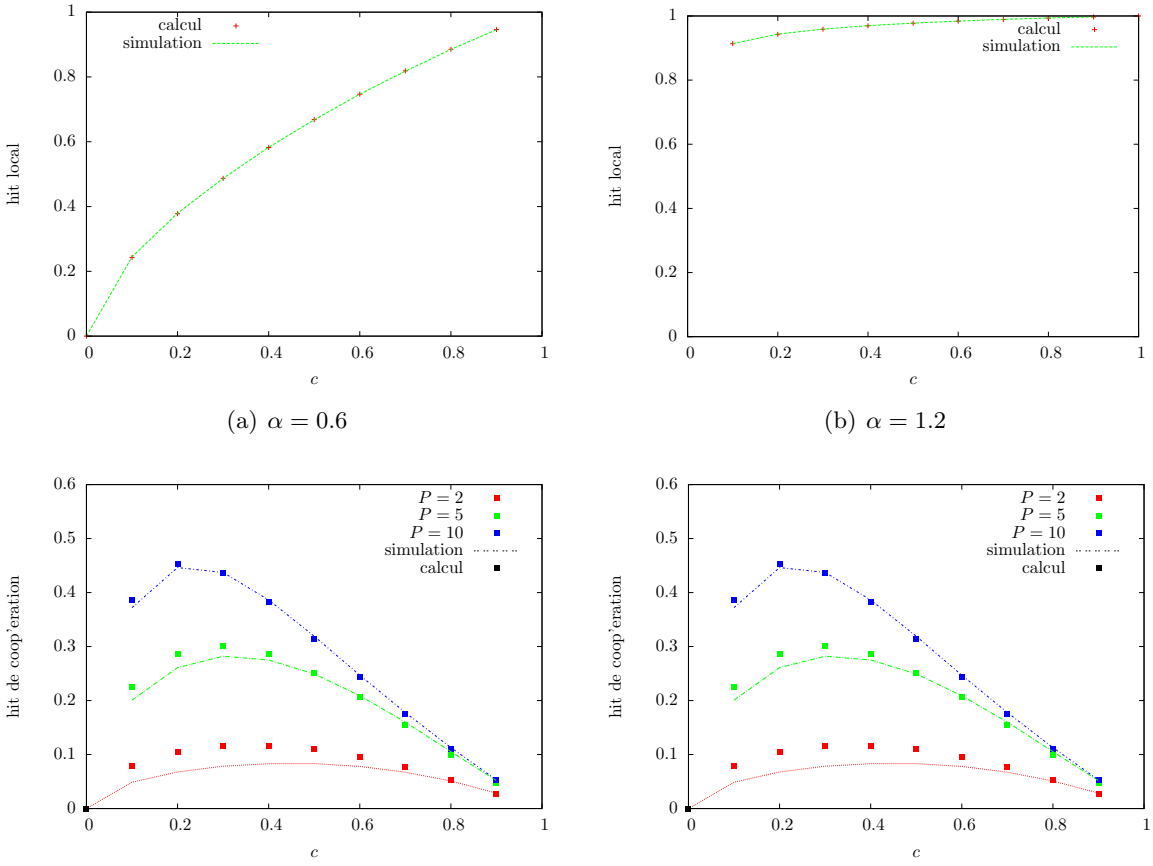
$$\theta' = \sum_{n=1}^{N} q(n) \left(\frac{1}{P}(1 - e^{-q'(n)t_c}) + (1 - \frac{1}{P})(1 - e^{-q(n)t_c}) \right) / \sum_{n=1}^{N} q(n)$$

En cas de miss local, les requêtes seront transmises aux caches responsables de la partition dont l'objet en question fait partie. Le cache k réalise un taux de hit supplémentaire pour ces requêtes. La popularité des objets appartenant à ces requêtes à l'entrée du cache k est $q'(n)$ et la probabilité de hit de chaque objet est $1 - e^{-q'(i)t_c}$. Ces objets arrivent avec un taux $q(i)e^{-q(i)t_c}$ et ils arrivent de $P - 1$ caches. Chaque cache autre que k envoie uniquement une fraction $1/P$ des objets ayant subi un miss au cache k, c'est-à-dire les objets appartenant au sous-catalogue M_l. La probabilité de hit global du trafic débordant des caches après un premier essai (cache local) est exprimée par :

$$\theta'' = \sum_{n=1}^{N} q(n)(1 - \frac{1}{P})e^{-q(n)t_c}(1 - e^{-q'(n)t_c}) / \sum_{n=1}^{N} q(n).$$

On compare à la figure 13.3 les résultats obtenus par simulation et par calcul pour les probabilités de hit locales θ' et les probabilités de hit supplémentaires dû à la

coopération θ'', pour un catalogue de 10^5 objets, pour P égale à 2, 5 et 10 et pour les deux lois Zipf(0.6) et Zipf(1.2).



(a) $\alpha = 0.6$ (b) $\alpha = 1.2$

(c) $\alpha = 0.6$ (d) $\alpha = 1.2$

FIGURE 13.3 – Les probabilités de hit locales et de coopération en fonction de la taille de cache normalisée

On remarque que la probabilité de hit locale obtenue par calcul coïncide avec celle obtenue par simulation, et que cette probabilité est indépendante de P. Pour les probabilités de hit de coopération, une légère différence entre simulation et calcul est à noter pour $P = 2$. Cette différence diminue au fur et à mesure que P augmente. Pour $P = 10$, les valeurs de simulation et de calcul sont identiques. Ces résultats s'expliquent par la précision de l'approximation de Che. Appliquée à un deuxième niveau de caches, on s'attend à ce que l'erreur due à l'hypothèse d'indépendance soit faible si le nombre de caches de premier niveau est assez grand ($n > 5$).

On note θ la probabilité de hit global au premier niveau : $\theta = \theta' + \theta''$. Pour un cache

de taille C, le coût global de la structure est :

$$cout(C) = Ak'_b\frac{P-1}{P}(1-\theta') + Ak_b(1-\theta) + nk_mC.$$

Cet algorithme est inspiré du load-sharing et chaque cache ne peut dépasser la taille de M/P. La taille normalisée du cache peut donc s'écrire $c = CP/M$. On trace la probabilité $\theta(c/P)$ en fonction de c pour différentes valeurs de P pour la loi Zipf(0.6). Les résultats sont présentés à la figure 13.2.

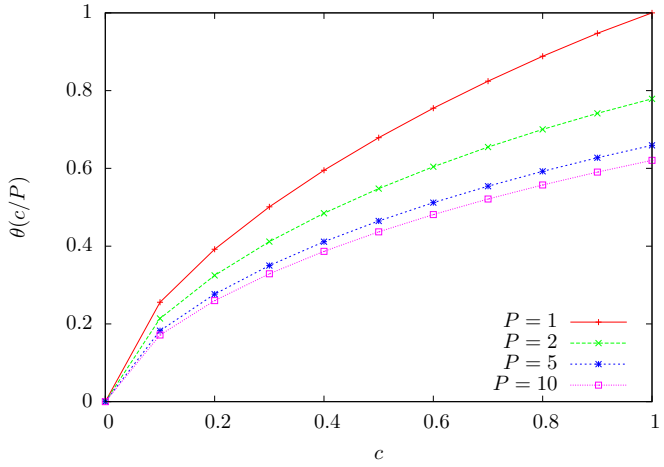


FIGURE 13.4 – Probabilité de hit global $\theta(c/P)$

On remarque que $\theta(c) > \theta_P(c/P)$ pour $P \geq 2$ où $\theta(c)$ correspond à la probabilité de hit au premier niveau sans coopération ($P = 1$). On note $\tilde{\theta}$ la probabilité de hit locale des objets qui ne font pas partie de la partition du cache. On a $\tilde{\theta}(c) < \theta'(c)$ et $\theta_p(c) < \theta(cp)$. Divisant par nk_mM, le coût normalisé peut s'exprimer :

$$\delta_{cc}(c) = \Gamma(1-\theta_P(c)) + \Gamma\frac{k'_b}{k_b}(1-\frac{1}{P})(1-\tilde{\theta}(c)) + c/P.$$

Tout calcul fait :

$$\delta_{cc}(c) > \delta(c) + (1-\frac{1}{P})(\Gamma\frac{k'_b}{k_b}(1-\theta'(c)) - c).$$

Pour savoir quelle architecture est la plus rentable, il faut donc évaluer la quantité $\Gamma\frac{k'_b}{k_b}(1-\theta'(c)) - c$.

On pose $f(c) = \Gamma\frac{k'_b}{k_b}(1-\theta'(c)) - c$. La fonction $f(c)$ est minimale si $\theta'(c)$ et c sont maximales, ce qui correspond à un stockage complet au premier niveau, quel que soit

la valeur de k'_b/k_b et de Γ. La coopération est moins rentable qu'une hiérarchie simple si la taille des caches de premier niveau est grande. La conclusion dépend encore des coûts relatifs k'_b et k_b. Notons que, plus le coût de bande passante augmente relatif au coût des caches, plus il s'avère inutile de faire coopérer les caches.

13.3 Routage orienté contenu

Notre étude est favorable à un stockage complet à l'Edge puisque le tradeoff entre bande passante et mémoire est optimal dans le cas d'un stockage entier au premier niveau. Nos résultats sont soutenus par d'autres études telles que [59] et [5]. Cependant, une autre étude remet en question ces résultats, les jugeant insuffisants car ils comparent un réseau de caches non optimal avec un caching à l'Edge ce qui est pour les auteurs simpliste. Effectivement, [44] met en doute les résultats constatés du fait que la comparaison n'a pas été faite avec une hiérarchie de caches optimale. Avec un couplage meta-caching/stratégie de forwarding, l'écart entre une hiérarchie de caches optimale et un stockage à l'edge peut s'avérer beaucoup plus important.

Nous nous intéressons à l'influence que peut avoir l'utilisation d'une hiérarchie de caches plus efficace que la hiérarchie LRU utilisant SPF (shortest path first) comme politique de forwarding. Nous souhaitons utiliser les caches pour obtenir des taux de hit assez grands (plus de 90%). Dans ce cas, l'efficacité de la politique de méta-caching LCD tend vers celle d'une politique LCE.

La seule question restant à traiter est l'influence de la politique de forwarding sur les coûts. On considère une hiérarchie de caches LRU à deux niveaux. On note n le nombre de caches au premier niveau. Les caches au premier niveau ont la même taille C et on considère une demande symétrique de données. Ces caches sont reliés à un cache de niveau 2 stockant la totalité du catalogue. Nous étudions donc l'optimum dans le cas d'un taux de hit de 100%.

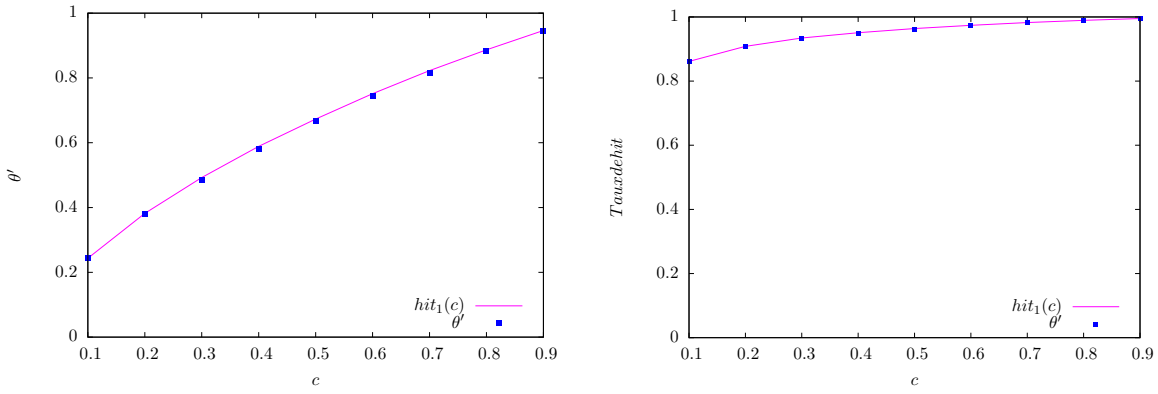
Chaque cache au premier niveau est relié à tous les autres caches par des liens et le coût de bande passante de chaque lien est k_b. Le coût de bande passante des liens reliant le cache du premier niveau au cache du deuxième niveau est k'_b. On note θ' le taux de hit local dû au hit des objets demandés par les utilisateurs et on note θ'' le taux de hit dû à la coopération. Le taux de hit global au premier niveau est $\theta = \theta' + \theta''$. En utilisant les simulations, on trace θ' et on le compare à $hit(c)$ le taux de hit au premier niveau d'une hiérarchie SPF pour un catalogue de $M = 10^4$ objets.

On remarque que θ' est égal au taux de hit d'un cache LRU simple, la valeur de n ne changant rien à ce constat. Le coût de cette hiérarchie peut donc être exprimé par :

$$cost = Ak_b(1 - \theta(c)) + Ak'_b\theta'' + nk_mC.$$

On normalise cette formule et on trouve :

$$cost_N(c) = \delta(c) + A\theta''(k'_b - k_b).$$



(a) Zipf(0.6) (b) Zipf(1.2)

FIGURE 13.5 – Probabilité de hit locale θ' et $hit(c)$ comparés par simulation

Puisque $k'_b < k_b$, $cost_N(c) < \delta(c)$. Donc en utilisant une politique de forwarding NRR, on réalise un coût inférieur.

Nous allons évaluer les optima dans ce cas. La popularité de l'objet i à l'entré de chaque cache au premier niveau peut être exprimé par : $q'(i) = q(i) + \sum_{k=1}^{n-1}(m^{n-l}(1-m)^l)/l$ tel que m est la probabilité de miss de i de popularité $q(i)$ pour un cache LRU.

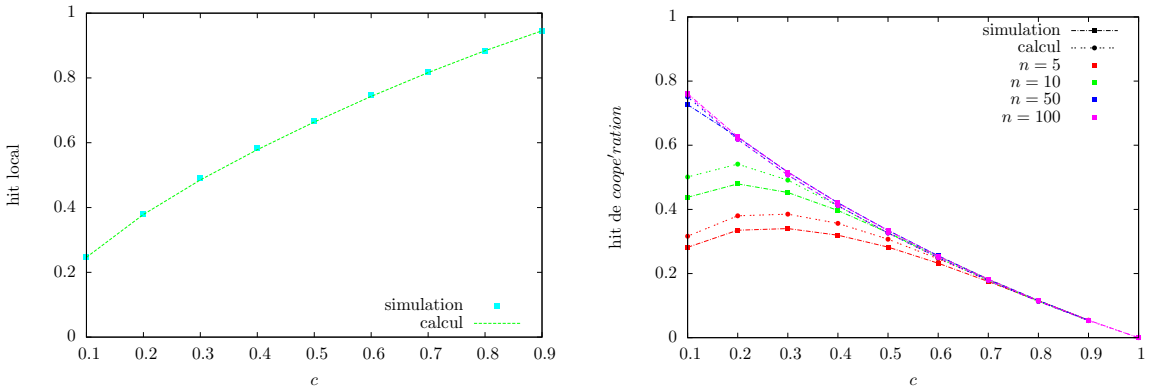
La probabilité de hit locale d'un objet peut être exprimée par : $hit(n) = 1 - exp(-q'(n) * t_c)$ Le t_c est calculé en utilisant l'équation modifiée de Che : $\sum_{i=1}^{M}(1 - exp(-q'(i) * t_c) = C$

La probabilité de hit locale est exprimée par : $\theta' = \sum_{i=1}^{M} q(i) * (1 - exp(-q'(i) * t_c))$
La probabilité de hit de coopération est exprimée par :$\theta'' = \sum_{i=1}^{M} q(i) * exp(-q'(i) * t_c) * (1 - exp(-(n-1) * t_c * q'(i)))$

On compare les résultats de calcul avec les résultats de simulation pour un catalogue de 10^4 objets, n prenant les valeurs de 10,20,40 et 100.

Les calculs sont effectuées pour une loi Zipf(0.6) mais les conclusions sont vrai pour tout α. Les résultats du hit local et du hit de coopération sont représentés dans les graphs 13.6(a) et 13.6(b)

On remarque que les hit locaux calculés par la méthode de Che modifiée coïncident exactement avec les probabilités de hit récupérés des simulations, et ces taux de hit sont indépendants de la valeur de n. D'autre part, on note une différence entre le taux de hit de coopération récupéré des calculs, et le taux de hit obtenu par simulation pour $n = 5$ et $n = 10$, mais cette différence disparait pour des tailles de caches grands. Pour $n = 100$ aucune différence est à noter entre les deux méthodes calcul et simulation. Le cout d'une hiérarchie NRR est estimée à : $cost = n * k_m * C + A * k_b * (1-\theta) + A * k'_b * \theta''$



(a) hit local (b) hit coopération

FIGURE 13.6 – Probabilité de hit comparés par simulation et calcul

le cout normalisé est estimé à : $cost_n = c + \Gamma(1-\theta) + \Gamma * (\frac{k'_b}{k_b}) * \theta''$ On trace la taille de cache optimale en fonction de Γ et k'_b/k_b On remarque que pour les valeurs futures

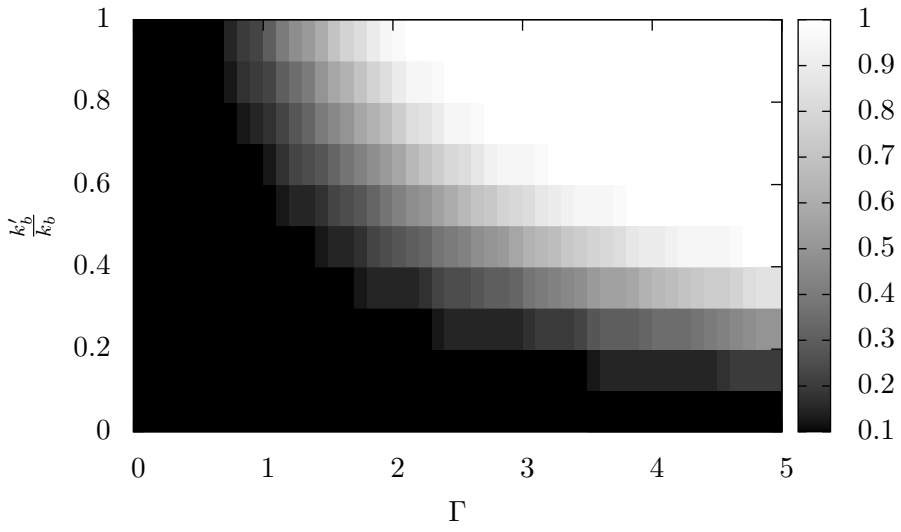


FIGURE 13.7 – Taille de cache normalisée optimale en fonction de Γ et k'_b/k_b

du Γ nominal, et pour un k'_b/k_b=0.8 la solution optimale consiste à tout stocker au

premier niveau de caches.

Chapitre 14

Hiérarchies alternatives

14.1 Coût d'une hiérarchie de caches à plusieurs niveaux

14.1.1 Evaluation des coûts

Sem Borst et al. [52] affirment qu'il n'y a aucun intérêt à distribuer des caches sur plus de deux niveaux. Cette affirmation a attiré notre curiosité. Pour vérifier ceci, on considère une hiérarchie de caches à 5 niveaux.

L'approximation de Che étant bonne pour tous les niveaux de caches, on l'utilise pour calculer la probabilité de hit globale de la structure ainsi que la probabilité de hit à chaque niveau.

Pour simplifier notre analyse, on considère le cas d'une hiérarchie homogène et symétrique en arbre à *nblevel* niveaux. On se restreint dans cette étude à 5 niveaux de caches ($nblevel = 5$). Le cache de niveau 5 est la racine de l'arbre et n'a pas de parent. Chaque noeud a le même nombre nb_f de fils. A chaque niveau i de l'arbre, les caches ont la même taille T_i. On fixe $nb_f = 5$.

On considère 3 cas :

- une répartition équitable de mémoire à tous les niveaux, $T_i = T_{i+1}$ pour tout i,
- une répartition favorisant les caches aux premiers niveaux $T_{i+1} = T_i/2$,
- une répartition favorisant les caches supérieurs $T_{i+1}/2 = T_i$.

Le catalogue a une taille M. Notre objectif est de comparer les différents cas en terme de coûts de mémoire et de bande passante tout en préservant la même valeur de probabilité de hit global.

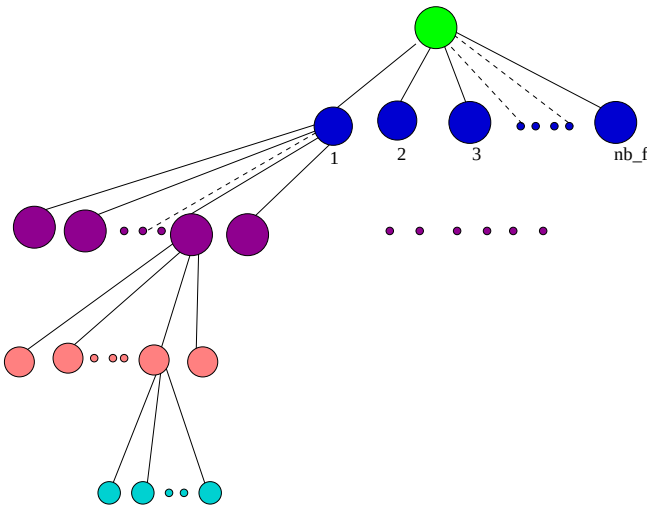


FIGURE 14.1 – Hiérarchie de caches à 5 niveaux

On note p_{miss_i} la probabilité de miss au niveau i. Le coût de la structure est :

$$cout = Ak_b \sum_{k=1}^{nblevel-1} \prod_{i=1}^{k} p_{miss_i} + k_m \sum_{i=1}^{nblevel} T_i (nb_f)^{nblevel-i}.$$

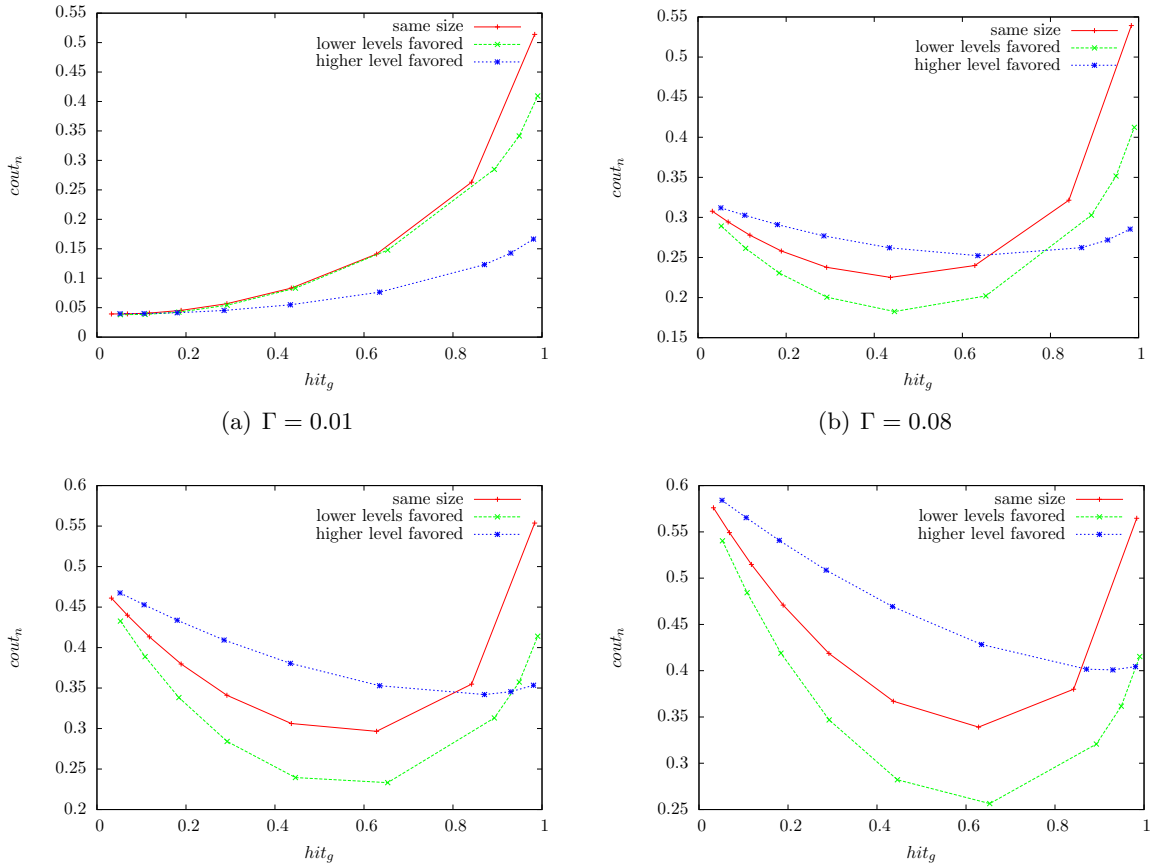
On note $cout_n$ le coût normalisé en divisant le coût par $k_m Mn$, n étant le nombre total de caches. On pose $\Gamma = \frac{Ak_b}{k_m Mn}$. Alors

$$cout_n = \Gamma(\sum_{k=1}^{nblevel-1} \prod_{i=1}^{k} p_{miss_i}) + \sum_{i=1}^{nblevel} t_i nb_f^{nblevel-i}/n.$$

La figure 14.1.1 représente les résultats obtenus pour différentes valeurs de Γ.

La valeur nominale de Γ est de 0.12. Le stockage favorisant les premiers niveaux est optimal juqu'à la valeur d'un taux de hit global de 93%. Le stockage favorisant les premiers niveaux est toujours optimal si $\Gamma \geq 0.16$. Il est probable que cette valeur soit bientôt atteinte puisque Γ est en augmentation permanente du fait de l'augmentation rapide de la demande entrainant une augmentation de la valeur de débit.

Actuellement les coûts de la mémoire baissent plus vite que les coûts de la bande passante et Γ augmente de plus en plus dépassant actuellemnt la valeur de 0.12. Il est vrai que la popularité ne suit pas une loi Zipf(0.6) mais plutôt une loi composée de Zipf(0.6) et Zipf(0.8), mais dans tous les cas, la tendance future est le stockage entier au premier niveau en évitant la maintenance excessive de la bande passante entre routeurs. La répartition des caches n'est pas forcément gagnante pour un opérateur ou un fournisseur de contenu puisqu'il faut prendre en compte les coûts élevés de la bande passante par rapport aux coûts de la mémoire.



(a) $\Gamma = 0.01$ (b) $\Gamma = 0.08$

(c) $\Gamma = 0.12$ (d) $\Gamma = 0.15$

FIGURE 14.2 – Coûts normalisés pour différentes valeur de Γ

14.1.2 Coopération de caches

On restreint notre étude dans cette section au cas d'une répartition équitable de mémoire comme c'est le cas pour CCN. Notre objectif est de déterminer le gain en mémoire et en bande passante en comparant le cas d'une coopération des caches et le cas d'une non-coopération des caches.

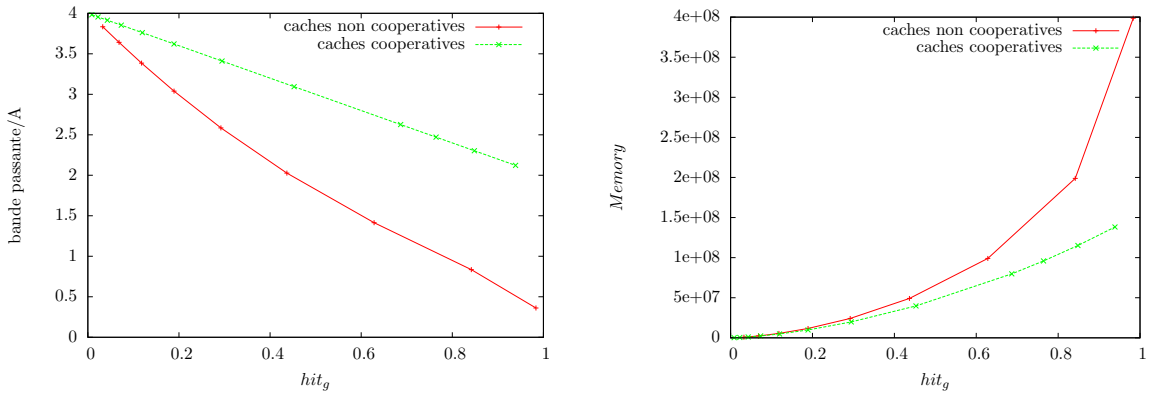
Il est clair que le gain en mémoire qu'on peut avoir avec une coopération est très intéressant mais par ailleurs le gain en bande passante décroit. On choisit la coopération par répartition du catalogue, c'est-à-dire que chaque niveau de cache est destiné à stocker 1/5 du catalogue. Ainsi, on stocke le chunk numéro i dans le niveau $i \pmod 5$.

Puisque les niveaux de caches traitent des catalogues disjoints et que les chunks suivent la même loi de popularité que les objets, il est simple de trouver la probabilité de hit des caches à chaque niveau.

La quantité de mémoire utilisée est : $\sum_{i=1}^{nblevel} C_1(nb_f)^{nblevel-i}$. On note $hit_i(c_1)$ la probabilité de hit d'un cache de taille normalisée c_1 pour le niveau de cache i pour une hiérarchie de caches père-fils et un catalogue de taille M.

La bande passante utilisée pour la hiérarchie sans coopération est :

$$\sum_{i=1}^{nblevel-1} (nb_f)^{nblevel-i}(1 - hit_i(c_1)).$$



(a) Bande passante

(b) Mémoire

FIGURE 14.3 – Comparaison entre caches coopératifs et non coopératifs

On remarque que les caches coopératifs utilisent moins de mémoire pour obtenir la même probabilité de hit global. Cependant, la quantité de bande passante consommée par les caches coopératifs est plus importante que la quantité de bande passante consommée par les caches non coopératifs.

14.2 Coûts et politiques de remplacement

14.2.1 Politique LFU

On considère une hiérarchie en arbre de caches LFU avec les objets stockés du niveau bas au niveau supérieur suivant l'ordre décroissant de leur popularité. La politique de forwarding est une politique de recherche père/fils puisque le fils cherche l'objet en suivant le chemin le plus court menant au serveur d'origine qui est placé au niveau supérieur de la hiérarchie. On considère alors un catalogue de M objets classés selon un ordre décroissant de popularité : $o_1, o_2, o_3, ..., o_M$. La taille des caches au niveau i est c_i . La figure 14.4 représente notre hiérarchie à étudier et le tableau 14.2.1 présente les paramètres.

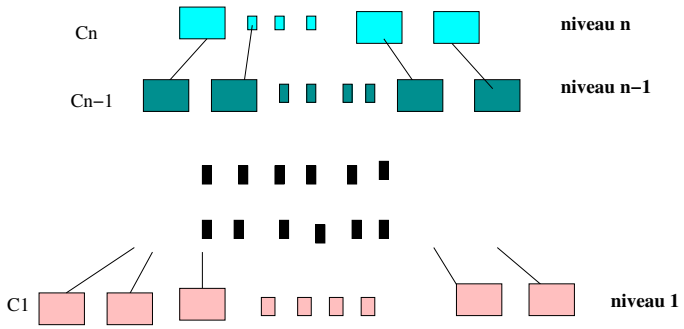


FIGURE 14.4 – Réseau de caches avec une politique LFU

M	taille du catalogue
C_i	taille du cache au niveau i
c_i	taille du cache au niveau i normalisée $c_i = C_i/M$
P_{mi}	probabilité de miss de la hierarchie jusqu'au niveau i
k_b	coût unitaire de la bande passante
k_m	coût unitaire de la mémoire
A	débit d'entrée à la hierarchie
nb_i	nombre de caches au niveau i
P_{mcible}	la probabilité de miss global ciblée

La probabilité de miss au premier niveau P_{m1} peut être exprimé pour LFU par :

$$P_{m_1} = \frac{M^{1-\alpha} - C_1^{1-\alpha}}{M^{1-\alpha}} \tag{14.1}$$

ou

$$P_{m_1} = 1 - c_1^{1-\alpha}.$$

D'une manière générale, la probabilité de miss de la hiérarchie de caches constituée des niveaux 1 à i est exprimée par :

$$P_{m_i} = 1 - (\sum_{k=1}^{i} c_k)^{1-\alpha}. \tag{14.2}$$

On suppose la probabilité de hit globale de la hiérarchie est fixée à une valeur cible :

$$P_{mg} = 1 - (\sum_{k=1}^{n} c_k)^{1-\alpha}$$

d'où

$$c_n = (1 - P_{mg})^{1/(1-\alpha)} - \sum_{k=1}^{n-1} c_k. \tag{14.3}$$

La fonction de coût peut être exprimée par :

$$cout = Ak_b(\sum_{k=1}^{n-1} p_{mk}) + \sum_{k=1}^{n-1}(nb_k - nb_n)C_k k_m$$

On pose $\Gamma' = \frac{Ak_b}{Mk_m}$. On divise la fonction de coût par une constante $k_m M$ et on obtient :

$$cout_N = \sum_{k=1}^{n-1} \Gamma'(1 - (\sum_{i=1}^{k} c_i)^{1-\alpha}) + \sum_{i=1}^{n-1}(nb_i - nb_n)c_i.$$

L'optimum vérifie $\frac{\partial cout_N}{\partial c_1} = 0$ et $\frac{\partial cout_N}{\partial c_2} = 0$, pour une probabilité de hit global de 1 : $c_1 = Min(\left(\frac{\Gamma'(1-\alpha)}{(nb_1 - nb_2)}\right)^{1/\alpha}, 1)$. En posant $\Gamma = \frac{\Gamma'}{nb_1 - nb_2}$, on obtient :

$$c_1 = Min(1, (\Gamma(1-\alpha))^{1/\alpha}). \tag{14.4}$$

A la figure 14.5, on trace la valeur de l'optimum en fonction de α.

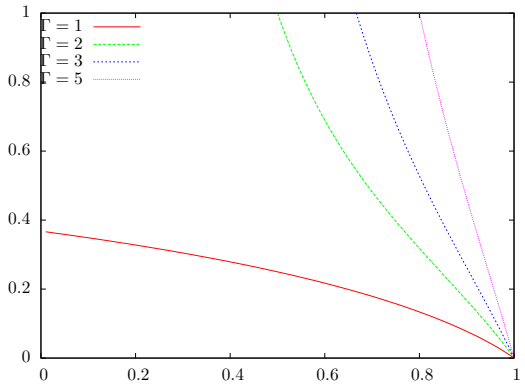
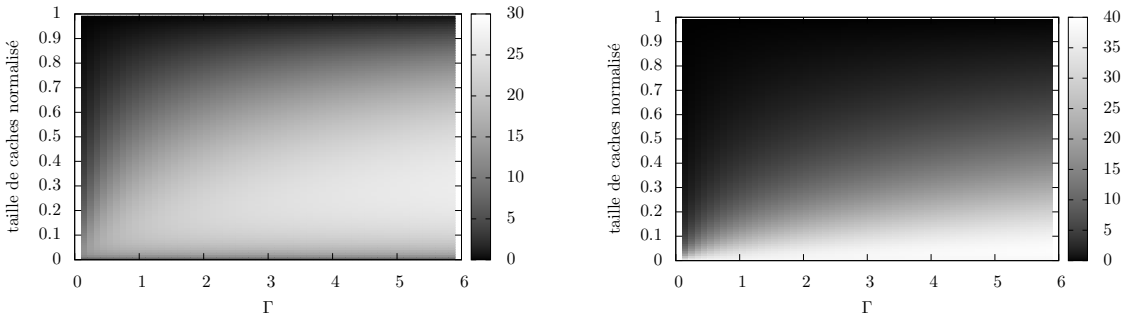


FIGURE 14.5 – La taille de cache optimale en fonction de α.

On remarque que plus Γ augmente, plus l'optimum correspond à un stockage au premier niveau de cache. Pour $\alpha = 0.8$, à partir de $\Gamma = 5$ l'optimum correspond à un stockage entier à l'edge. Nous nous intéressons de plus à la différence de coûts entre LRU et LFU, les résultats sont représentés sur la figure 14.6.

On remarque que la différence de coûts entre politiques LRU et LFU diminue quand la taille des caches augmente. Donc il n'est pas utile d'implémenter une politique matériellement coûteuse comme LFU car LRU suffit.



(a) $\alpha = 0.6$ (b) $\alpha = 1.2$

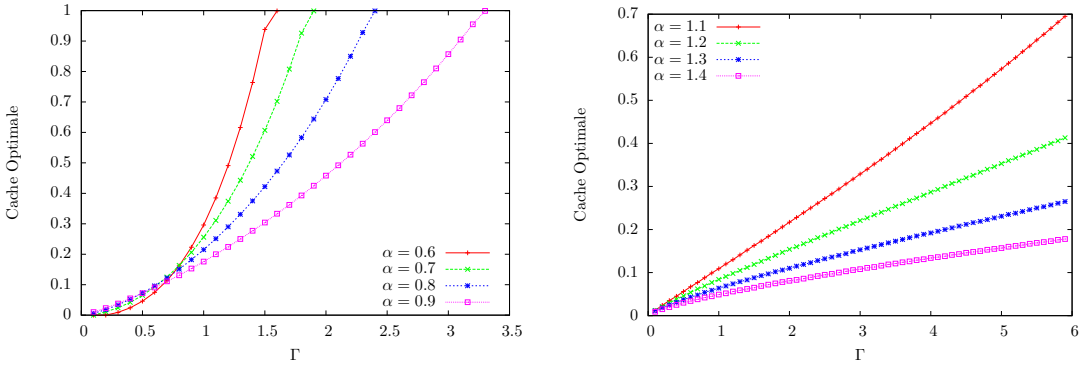
FIGURE 14.6 – Différence entre coûts LRU et LFU en %

14.2.2 Politique Random

La politique Random est une concurrente forte de LRU, son implémentation matérielle est simple et ses performances ne s'éloignent pas de celles de LRU. Dans cette section nous étudions le tradeoff de cette politique. En utilisant la même équation des coûts pour la hiérarchie LRU, le coût d'une hiérarchie de caches Random est exprimé par :

$$cost_N = \Gamma(1-h) + c$$

où h est la probabilité de hit globale de la hiérarchie, et c est la taille de cache normalisée au premier niveau. Avec la politique Random, nous traçons à la figure 14.7 la valeur de l'optimum en fonction du Γ pour différentes valeurs de α. Nous distinguons les cas $\alpha < 1$ et $\alpha > 1$. On remarque que l'optimum correspond à un



(a) $\alpha < 1$ (b) $\alpha > 1$

FIGURE 14.7 – Tailles de caches optimales en fonction de Γ

stockage complet à l'Edge pour des arrivées suivant la loi Zipf(0.6) à partir d'une valeur de $\Gamma = 1.5$. On conclut que même pour une politique Random, et compte tenu des valeurs de Γ selon nos estimations, un stockage complet à l'edge correspond à la solution optimale.

Par ailleurs, quand la taille des caches est proche de la taille du catalogue, les probabilités de hit LRU et Random sont très proches. L'utilisation de la politique Random devient intéressante alors, car cette dernière est moins coûteuse matériellement que LRU.

Chapitre 15

Conclusion

Dans cette partie, nous évaluons le coût d'une hiérarchie de caches en prenant en compte le coût de la mémoire et le coût de la bande passante. Nous utilisons la formule de Che pour évaluer numériquement ce tradeoff. Le coût optimal dépend de la valeur du rapport $\Gamma = k_c A/(Mk_m n)$ ainsi que de la loi de popularité des objets.

Nos résultats suggèrent que dans un futur proche, l'optimum correspondra à un stockage complet à l'edge. Mais cette étude étant réalisée avec une fonction de coût linéaire, et avec des estimations non exacts des facteurs impactant les coûts, on ne peut la considérer que comme une première évaluation. Les opérateurs possédant plus de détails et précisions pourraient adapter cette étude en intégrant des fonctions de coûts réels.

Néanmoins, gérer un grand cache est plus compliqué que la gestion d'un petit cache, et les performances et la rapidité de réponse d'un cache dépendent de sa taille. Dans ce cas, chaque cache à l'edge peut être lui-même distribué localement en un ensemble de caches. L'opérateur pourrait dédier à titre d'exemple un ou plusieurs caches de taille maximale à chaque fournisseur de contenu.

Nous pensons que les fonctions de stockage et de gestion de contenu sont indépendantes. L'opérateur ne peut en plus des fonctions de routage, fournir des réponses aux requêtes d'utilisateurs, vérifier les données en supprimant les données illégales par exemple, en mettant à jour une donnée, etc. Le fait de centraliser les données permet de réaliser rapidement des opérations de mise à jour sur les données.

Conclusion générale

L'Internet devient centré sur les contenus, 90% des échanges sur internet étant des consultations de sites Web, des téléchargements de fichier, ou des streamings audio ou vidéo. La proposition CCN de Jacobson et al. vient donc au bon moment et suscite un grand intérêt.

L'objectif de cette thèse était de rajouter à la proposition CCN des fonctionnalités de gestion du trafic. En effet le protocole de transport proposé au début n'offrait pas un bon débit, puisque la détection de rejet se basait uniquement sur le timeout. D'autre part les paquets Data suivant le sens inverse des paquets Interest ne suffit pas pour assurer un partage équitable de bande passante et la protection des flux sensibles.

Nous avons proposé une solution dite "flow aware networking", inspirée de travaux antérieurs sur IP où cette manière de gérer le traffic par flot a prouvé son efficacité. Nous proposons également l'utilisation de la détection rapide de congestion pour rendre plus efficaces les protocoles de transport. L'interest Discard est un nouveau mécanisme conçu spécialement pour les réseaux CCN permettant de limiter les rejets de paquets Data en supprimant sélectivement des paquets Interest en cas de congestion.

Avec les réseaux orientés contenus, on cherche les données dans n'importe quelle destination possédant le contenu. Dans ce contexte, les protocoles multipaths semblent prometteurs. Nous avons observé cependant que le téléchargement des données de sources multiples "pollue" les caches et détériore ainsi la probabilité de hit globale de l'architecture. Utiliser les chemins les plus courts pour tous les flux tant que la charge des liens ne dépasse pas un seuil maximal s'avère suffisant. Les chemins longs ne doivent être utilisés que lorsque leur charge est faible, sauf bien entendu en cas d'une panne au niveau du chemin le plus court.

Les caches sont des acteurs majeurs dans la gestion du trafic et l'étude de leur performance est essentielle. Il est nécessaire notamment d'évaluer la manière dont devrait être réparti les contenus dans une hiérarchie de caches. La simulation étant coûteuse, puisque toute simulation de catalogue très volumineux consomme beaucoup de temps, nous avons utilisé des calculs approchés à l'aide de la formule de Che. Cette formule

donne le taux de hit d'un cache de taille donnée sous l'hypothèse d'indépendance antre les arrivées de requêtes. Elle s'avère très précise et facile à utiliser étant valable pour tout loi de popularité et toute taille de cache. L'utilisation de cette formule peut s'étendre à deux ou plusieurs niveaux de caches, à condition que le nombre de cache au niveau inférieur soit grand (supérieur ou égale à 5, d'après nos études) afin de garantir l'indépendance des requêtes. En utilisant ce modèle mathématique, et avec des popularités et tailles de catalogues s'approchant de la réalité, nous avons pu étudier les perfomances d'une hierarchie de caches LRU.

Dans un article récent [59], Fayazbakhsh et al. se demandent s'il est vraiment nécessaire de distribuer les caches dans une hierarchie. D'autre part les CDNs rencontrent un succès énorme et sont opérationnelles depuis des années. Pour un opérateur, l'enjeu principale dans le choix d'une architecture orienté contenu est le coût et la difficulté de mise en place. Au niveau de la difficulté de mise en place les CDN sont sans contestation mieux puisqu'ils sont déja opérationnels. Le savoir faire dans ce domaine est fleurissant contrairement aux ICN qui manquent toujours d'une solution claire pour le routage, pour l'identification des objets dans le réseau ainsi que le changement des fonctionnalités des routeurs pour effectuer le stockage.

Au niveau des coûts, nous ne pouvons estimer précisément laquelle des architectures est la moins coûteuse. Il est également difficile d'estimer tous les coûts possibles et d'avoir un modèle de coût fiable s'approchant des modèles utilisés par les opérateurs. Ne pouvant affranchir le pas de la confidentialité des données des opérateurs, nous nous contentons d'estimer les coûts en prenant en compte les coûts de la mémoire et de la bande passante. Nous adoptons un simple modèle de coût linéaire permettant une comparaison chiffrée quoique grossière des différentes options.

Nous considérons qu'une hiérarchie est mieux qu'une autre si le coût global incluant le coût des caches et le coût de la bande passante est plus faible. Notre objectif revient à trouver un tradeoff optimal, puisque une hierarchie distribué est coûteuse en mèmoire mais ne coûte rien en bande passante, alors qu'une hiérarchie centralisée coûte moins cher en mémoire mais est plus coûteuse en bande passante.

Suite aux résultats de nos études, nous recommandons un stockage presque complet à l'edge. Cette solution paraît optimale sutout dans un avenir proche car le coût des mémoires baisse beaucoup plus vite que celui de la bande passante. D'autres études remettent en question cette conclusion, la jugeant rapide puisque certaines politiques de meta caching et forwarding améliorent la probabilité de hit des caches au premier niveau et réduisent donc le coût de la bande passante.

Nous avons approfoni l'étude de certains cas, dont celui de caches coopératives avec une politique de forwarding NRR (Nearest Replica Routing). Dans ce cas pour une hierarchie à deux niveaux l'optimum correspond bien à un stockage à l'edge. Alors que Borst et al. [52] assurent qu'il n' y a aucun avantage à distribuer les contenus sur plusieurs niveaux de caches, cette affirmation n'ést pas accompagné de preuve. Nous avons donc estimé les coûts pour une hierarchie à plusieurs niveaux avec politique LRU. Les constats restent les mêmes, il est plus rentable à moyen terme d'utiliser de

grands caches à l'edge.

On devrait poursuivre cette évaluation avec, par exemple, une politique de remplacement LCD et un routage NRR mais nos résultats préliminaires mettent sérieusement en question la notion CCN de mettre des caches dans tous les routeurs. Nos résultats s'accordent avec ceux de Fayazbakhsh et al. [59], même si les modèles utilisés sont très différents.

Un autre problème posé par CCN est la difficulté de gestion de données dans les caches. Si un fournisseur de données souhaite supprimer un objet, il est difficile de parcourir toute la hierarchie pour supprimer l'objet ou le mettre à jour. Une hiérarchie avec peu de niveaux est plus facile à gérer. Compte tenu des doutes concernant le coût élevé de CCN, de la difficulté de mise en oeuvre nécessitant des modifications majeures, de la nécessité de mettre en place une stratégie de gestion de noms d'objets et leur authentifications, et de la difficulté à gérer les données sur des serveurs distribués, il nous parait préférable de stocker tous les objets à l'edge tout proche des usagers. De grands caches à ce niveau pourraient être spécialisé par fournisseur de contenus pour faciliter la gestion. Chaque objet s'identifierait d'abord par son fournisseur, qui a la responsabilité d'assurer l'unicité de l'objet et son authentification, ainsi que la sécurité de ses objets.

L'opérateur peut prendre en charge la fonctionnalité du caching à l'edge en accord avec les fournisseur de contenus et attribuer à chaque fournisseur un ou plusieurs caches. L'opérateur peut devenir lui aussi fournisseur de contenus. Ceci bien sur peut être conclu à travers des accords commerciaux entre les différents acteurs. La fonctionnalité de l'internet serait alors surtout localisée à l'edge du réseau de l'opérateur. Certaines fonctionnalités proposés dans CCN peuvent bien entendu être utiles dans cette architecture comme, par exemple, l'utilisation des noms d'objets au lieu des adresses IP. Toute utilisateur demanderait un objet avec le nom du fournisseur suivi du nom d'objet et c'est le routeur d'accès qui va, soit envoyer la donné demandé à travers ces caches à l'edge, soit rediriger la demande vers le fournisseur de contenus.

Pour les travaux futurs, nous souhaitons approfondir l'étude du tradeoff mémoire/bande passante en utilisant des modèles de coûts plus réalistes, et pour des hiérarchies optimales NRR/LCD, ou des caches coopératives pour plusieurs niveaux de caches. Nous souhaitons aussi compléter notre étude sur les multipaths et leurs performances, d'analyser l'impact des caches et la manière de les gérer sur les performances des multipaths. Nous souhaitons aussi adapter la formule de Che à plusieurs cas, notament pour le cas d'une hierarchie LCD à plusieurs niveau de caches. Il est aussi important d'évaluer l'évolution des popularités des objets et leurs tailles.

Références

[1] T. Koponen, M. Chawla, G.-B. Chun, A. Ermolinskiy, H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 181–192, 2007.

[2] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *CoNext 2009*, 2009.

[3] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, "Design considerations for a network of information," *Proceedings of the 2008 ACM CoNEXT Conference*, 2008.

[4] "Publish-subscribe internet routing paradigm." `http://psirp.hiit.fi/`.

[5] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking : seeing the forest for the trees," in *Proceedings of HotNets-X*, pp. 1 :1–1 :6, 2011.

[6] S. Fdida and N. Rozhnova, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *in IEEE NOMEN Workshop, co-located with INFOCOM*, 2012.

[7] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," *ACM SIGCOMM Workshop on Information Centric Networking (ICN'12)*, 2012.

[8] G. Carofiglio, M. Gallo, and L. Muscariello, "Icp : Design and evaluation of an interest control protocol for content-centric networking," *IEEE NOMEN Workshop, co-located with INFOCOM*, 2012.

[9] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)." Technical report, Telecom ParisTech, 2011.

[10] P. R. Jelenkovic, "Approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities," *Annals of Applied Probability*, vol. 9, no. 2, pp. 430–464, 1999.

[11] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization : a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, (New York, NY, USA), pp. 15–28, ACM, 2007.

[12] K.Gummadi, R. S.saroiu, S.Gribble, A.Levy, and J.Zahorjan., "Measurement, modeling and analysis of a peer-to-peer file shraing workload.," in *SOSP'03*, 2003.

[13] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes : analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, (New York, NY, USA), pp. 1–14, ACM, 2007.

[14] G. Appenzeller, I. Kesslassy, and N. McKeown, "Sizing router buffers," in *SIGCOMM 2004*, 2004.

[15] J.Nagle, "On packet switches with infinite storage," in *RFC 970*, 1985.

[16] S. Oueslati and J. Roberts, "A new direction for quality of service : Flow aware networking," in *NGI 2005*, 2005.

[17] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate fair dropping for variable length packets," *ACM Computer Communications Review*, vol. 33, no. 2, pp. 23–39, 2003.

[18] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury, "Buffer management schemes for supporting TCP in gigabit routers with per-flow queueing," *IEEE JSAC*, vol. 17, no. 6, pp. 1159–1169, 1999.

[19] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, "Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing," in *ACM Sigmetrics*, 2005.

[20] S. Ben Fredj, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts, "Statistical bandwidth sharing : a study of congestion at flow level," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, 2001.

[21] B. Briscoe, "Flow rate fairness : Dismantling a religion," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 63–74, 2007.

[22] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," *SIGCOMM Comput. Commun. Rev.*, vol. 25, pp. 231–242, October 1995.

[23] T. Bonald, M. Feuillet, and A. Proutiére, "Is the "law of the jungle" sustainable for the internet ?," in *INFOCOM*, 2009.

[24] P. Key and L. Massoulié, "Fluid models of integrated traffic and multipath routing," *Queueing Syst. Theory Appl.*, vol. 53, pp. 85–98, June 2006.

[25] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp," in *NSDI'11 Proceedings of the 8th USENIX conference on Networked systems design and implementation*, 2011.

[26] C. Raiciu, D. Wischik, and M. Handley, "Practical congestion control for multipath transport protocols," in *UCL Technical Report*, 2010.

[27] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini, "Multipath congestion control in content-centric networks," *IEEE INFOCOM Workshop on emerging design choices in name oriented networking*, 2013.

[28] P. Viotti, "Caching and transport protocols performance in content-centric networks." Eurocom Master Thesis, 2010.

[29] Cisco, "Cisco visual networking index : Forecast and methodology, 2010-2015." White paper, 2011.

[30] A. Mahanti, C. Williamson, and D. Eager, "Traffic analysis of a web proxy caching hierarchy," *IEEE Network*, pp. 16–23, May/June 2000.

[31] J.Zhou, Y. Li, K. Adhikari, and Z.-L. Zhang, "Counting youtube videos via random prefix sampling," in *Proceedings of IMC'07*, 2011.

[32] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenke, "Web caching and zipf-like distributions : evidence and implications," *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1999.

[33] Y. Carlinet, B. Kauffmann, P. Olivier, and A. Simonian, "Trace-based analysis for caching multimedia services." Orange labs technical report, 2011.

[34] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *SIGOPS Oper. Syst. Rev.*, vol. 40, pp. 333–344, April 2006.

[35] P. Jelenkovic, X. Kang, and A. Radovanovic, "Near optimality of the discrete persistent caching algorithm," *Discrete Mathematics and Theoretical Computer Science*, pp. 201–222, 2005.

[36] G. Carofiglio, M. Gallo, and L. Muscariello, "Bandwidth and storage sharing performance in information centric networking," in *ACM Sigcomm workshop on ICN*, 2011.

[37] E.Rosensweig, J. Kurose, and D.Towsley, "Approximate models for general cache networks," in *Infocom*, 2010.

[38] A. Dan and D.Towsley, "An approximate analysis of the lru and fifo buffer replacement schemes," in *SIGMETRICS*, 1990.

[39] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems : modeling, design and experimental results," *IEEE JSAC*, vol. 20, no. 7, pp. 1305–1314, 2002.

[40] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for lru cache performance," in *Proceedings of ITC 24*, 2012.

[41] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, "Performance evaluation of the random replacement policy for networks of caches," *ASIGMETRICS '12*, 2012.

[42] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical web caches," *in IEEE ICPCC*, 2004.

[43] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings ICN '12*, (New York, NY, USA), pp. 55–60, ACM, 2012.

[44] G. Rossini and D. Rossi, "Coupling caching and forwarding : Benefits, analysis, and implementation," tech. rep., telecom-paristech.

[45] K. Cho, M. Lee, K. Park, T. Kwonand, Y. Choi, and S. Pack, "Wave : Popularity-based and collaborative in-network caching for content- oriented networks,," *in IEEE INFOCOM, NOMEN*, 2012.

[46] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," *in IFIP NETWORKING*, 2012.

[47] N. Laoutaris, H. Che, and I. Stavrakakis, "The lcd interconnection of lru caches and its analysis," *Perform. Eval.*, vol. 63, pp. 609–634, July 2006.

[48] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown ? : hamlet-like doubts in icn," *in ACM SIGCOMM,ICN*, 2012.

[49] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "Inform : a dynamic interest forwarding mechanism for information centric networking," *in ACM SIGCOMM, ICN*, 2013.

[50] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga, "Catt :potential based routing with content caching for icn," *in ACM SIGCOMM, ICN*, 2012.

[51] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, 2013.

[52] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," *IEEE INFOCOM*, 2010.

[53] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, (New York, NY, USA), pp. 44–49, ACM, 2011.

[54] D. Rossi and G.Rossini, "On sizing ccn content stores by exploiting topogical information," *IEEE INFOCOM, NOMEN Worshop*, 2012.

[55] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," *Computer Communications*, pp. 367–383, 2001.

[56] J. P. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz, "Networking requirements for interactive video on demand," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 779–787, 1995.

[57] I. Cidon, S. Kutten, and R. Soffer, "Optimal allocation of electronic content," *Computer Networks*, vol. 40, no. 2, pp. 205 – 218, 2002.

[58] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, "On the scale and performance of cooperative web proxy caching," in *ACM Symposium on Operating Systems Principles*, pp. 16–31, ACM New York, 1999.

[59] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. C. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain : incrementally deployable icn," *SIGCOMM '13 Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, 2013.

[60] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," *ICNP, IEEE*, 2013.

[61] N.Carlson, G.Dan, A.Mahanti, and M. Arlitt., "A longitudinal a characterization of local and global bittorrent workload dynamics," *In Proceedings of PAM12*, 2012.

[62] Z. Li and G. Simon, "Time-shifted tv in content centric networks : The case for cooperative in-network caching," *In Communications (ICC)*, 2011.

[63] J. Ni and D. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *Communications Magazine, IEEE*, 2005.