Optimization and virtualization techniques adapted to networking

Hatem Ibn Khedher

Spécialité: Informatique Télécommunications Electronique

Ecole doctorale: Informatique, Télécommunications et Électronique (Paris)

Présentée par

Hatem IBN KHEDHER

Pour obtenir le grade de

DOCTEUR DE TELECOM SUDPARIS

Sujet de la thèse :

Des techniques d'optimisation et de virtualisation adaptées aux réseaux

soutenue le 30 avril 2018

devant le jury composé de :

M. Hossam Afifi	Directeur de thèse	Télécom SudParis
M. Emad Abd-Elrahman	Co-encadrant	Institut National des Télécommunications
M. Jean-Pierre Guedon	Rapporteur	Ecole Polythechnique de Nantes
M. Enrico Natalizio	Rapporteur	Université de Technologie de Compiègne
M. Houda Labiod	Examinateur	Télécom ParisTech
M. Hassine Moungla	Examinateur	Université Paris Descartes
M. Ahmed E. Kamal	Examinateur	Iowa State University
M. Abderrezak Rachedi	Examinateur	Université Paris-Est Marne-la-Vallée
M. Jacky Forestier	Invité	Orange Labs

Thèse No 2018TELE0007

Speciality: Computer Science, Telecommunication and Electronics

Doctoral school: Computer Science, Telecommunication and Electronics (Paris)

Presented by

Hatem IBN KHEDHER

For the degree of

Doctor of Philosophy in Telecommunications

Subject :

Optimization and virtualization techniques adapted to networking

defended on 30 avril 2018

Thesis Committee:

M. Hossam Afifi	Thesis Director	Telecom SudParis
M. Emad Abd-Elrahman	Supervisor	National Telecommunication Institute
M. Jean-Pierre Guedon	Reviewer	Polytechnique Nantes
M. Enrico Natalizio	Reviewer	University of Technology of Compiegne
M. Houda Labiod	Examiner	Telecom ParisTech
M. Hassine Moungla	Examiner	University of Paris Descartes
M. Ahmed E. Kamal	Examiner	Iowa State University
M. Abderrezak Rachedi	Examiner	University of Paris-Est Marne-la-Vallee
M. Jacky Forestier	Invited	Orange Labs

Thesis No 2018TELE0007

Abstract

Content Distribution Network (CDN) is a large distributed network deployed worldwide designed to push the content on the edge of the networks. It moves content to mitigate increase in video traffic and enhance user satisfaction. It finds an intermediate point between the origin server and the clients. Network virtualization features, with novel concepts such as Network Function Virtualization (NFV) and Software Defined Network (SDN) solutions can answer user's demands and, subsequently increase the network efficiency. Therefore, we look at the potential benefits of virtualization combining with CDNs.

Therefore, in this thesis, we designed and implemented a tool which performs optimizations that reduce the number of migrations necessary for a delivery task. We present our work on virtualization in the context of replication of video content servers. The work covers the design of a virtualization architecture for which there are also several algorithms that can reduce overall costs and improve system performance. The thesis is divided into six parts.

1) Optimal solutions: In this situation, we presented an exact optimization solution based on the linear programming method that takes into account new criteria based on the desired quality of service of video clients, the capacity of the operator's network links and also the system parameters related to NFV/SDN virtualization such as disk size, available RAM memory, CPU capabilities, etc. A migration cost is introduced. It is the cost of deploying a new instance on a remote site. Once the optimization is complete, it recommended a list of data center sites to which virtual content servers could be migrated and thus optimize the parameters previously introduced.

2) Greedy (heuristic) solutions: For reasons of scalability, we propose a new heuristic method based on graph theory and more precisely on Gomory-Hu algorithm. The idea behind this solution is to reduce the number of links to check before moving to the optimization phase. Thus, we arrive at almost linear resolu-

tion times. Simulations on high density networks are carried out. We evaluate the results and propose possible server migrations as before.

3) Orchestration of services: Given that content delivery network has several virtual components, the proposed (optimal and heuristic) solutions are adapted to this novel context where the optimization objective is to orchestrate and chain a set of delivery components. Simulations on different network deployment scenarios are carried out. We evaluate the results and propose possible server orchestration and chaining between instances.

4) Multi-objective optimization: Still, the main objective in the previous techniques is unique (reduce the overall migration cost). Therefore, we tried in this thesis to propose a multi-objective optimization algorithm that take into consideration different criteria from the involved actors (content provider, network operator, and client) in the delivery chain. The idea behind this technique is to select an optimization layer (i.e., core, aggregation and edge/access) to where a migration algorithm can be executed.

5) Complex Active Networks: For such active network (such as information-centric networking (ICN) or Hadoop), the legacy optimization algorithms are no more valid and they need to be adjusted in this novel context. Therefore, we adapt the optimization constraint to these network types and we coma up by a ”budget” based model. The idea is to set a specific budget to each network node that represent the work that each node can do. Then, active network is modeled and the previous optimization algorithms are adapted and evaluated on complex graphs.

6) Integration of algorithms: Following the architecture derived from ETSI concerning the virtualization of network functions, we propose the integration of the previously designed and evaluated algorithms into a real architecture. A flow diagram is proposed and a prototype based on the open source software OpenStack, OpenDaylight and Open Virtual Switch is implemented and integrated into a real network operator. This part has been demonstrated and it gives very encouraging results.

This thesis is supported by models, implementations and simulations which provide results that showcase our work, quantify the importance of evaluating optimization techniques and analyze the trade-off between reducing operator cost and enhancing end user satisfaction index.

Key words- vCDN; SDN/NFV Optimization; Migration Algorithms; Scalability Algorithms

Résumé

Le Réseau de Distribution de Contenu (RDC) est un large réseau distribué qui est déployé dans le monde entier et est conçu pour pousser le contenu à la périphérie des réseaux. Il déplace le contenu pour atténuer l'augmentation du trafic vidéo et améliorer la satisfaction des utilisateurs. Il trouve des points intermédiaires entre le serveur d'origine et les clients finaux. Les fonctions de virtualisation de réseau, avec les nouveaux concepts émergents tels que les solutions NFV (pour Network Function Virtualization) et SDN (pour Software-Defined Network), peuvent répondre aux demandes des utilisateurs et, par conséquent, augmenter l'efficacité du réseau. Ainsi, nous examinons les avantages potentiels de la virtualisation apportés aux RDC.

Par conséquent, dans cette thèse, on présente nos travaux sur la virtualisation dans le contexte de la réplication des serveurs de contenu vidéo. Les travaux couvrent la conception d'une architecture de virtualisation pour laquelle on présente aussi plusieurs algorithmes qui peuvent réduire les coûts globaux à long terme et améliorer la performance du système. Le travail est divisé en six parties :

1) Les solutions optimales : Dans cette situation, on présente une solution d'optimisation exacte basée sur la méthode d'optimisation linéaire qui prend en compte de nouveaux critères basés sur la qualité de service désirée par les clients de la vidéo, la capacité des liens du réseau de l'opérateur et aussi des paramètres système liés à la virtualisation NFV/SDN comme la taille des disques, des conteneurs NFV, de la mémoire RAM disponible, des capacités CPU, etc. Un coûts de migration est introduit. Il correspond au coût de déploiement d'une nouvelle instance sur un site distant. Une fois l'optimisation est terminée, on propose une liste de sites de centre des données vers lesquels des serveurs de contenu virtuels pourraient être migrés et par conséquent optimiser les paramètres précédemment introduits.

2) Les solutions heuristiques : Pour des raisons de passage à l'échelle, on

propose une nouvelle méthode heuristique basée sur la théorie des graphes et plus précisèment sur Gomory-Hu. L'idée derrière cette solution est de réduire le nombre de liens à vérifier avant de passer à la phase d'optimisation. Ainsi, nous arrivons à des temps de résolution presque linéaires. Des simulations sur les réseaux à très haute densité sont effectuées. On évalue les résultats et propose des migrations de serveurs potentiels comme précédemment.

3) Orchestration de services : Etant donné que le réseau de diffusion de contenu comporte plusieurs composants virtuels, les solutions proposées (optimales et heuristiques) sont adaptées à ce nouveau contexte où l'objectif d'optimisation est d'orchestrer et d'enchaîner un ensemble de composants de livraison. Des simulations sur des différents scénarios de déploiement sont effectuées. Nous évaluons les résultats et proposons l'orchestration de serveurs candidats et le chaînage entre les instances.

4) Optimisation multi critères : L'objectif principal des techniques précédentes est unique (réduire le coûts global de la migration). Nous avons donc essayé dans cette thèse de proposer un algorithme d'optimisation multi-objectif prenant en compte les différents critères des acteurs (fournisseur de contenu, opérateur réseau, client) impliqués dans le chaînage de livraison. L'idée derrière cette technique est de sélectionner une couche d'optimisation (c'est-à-dire, coeur du réseau, agrégation et accès) où un algorithme de placement, de migration, ou d'orchestration est exécuté.

5) Réseaux actifs et complexes : Pour un tel réseau actif et complexe (tel que le réseau centré sur l'information (ICN) ou Hadoop), les algorithmes d'optimisation traditionnels ne sont plus valables et doivent être ajustés dans ce nouveau contexte. Par conséquent, nous avons adapté les contraintes d'optimisation à ces types de réseaux et nous avons appliqué un modèle basé sur le «budget». L'idée est de définir un budget spécifique pour chaque noeud de réseau qui représente le travail que chaque noeud peut faire. Ensuite, le réseau actif est modélisé et les algorithmes d'optimisation précéedents sont adaptés et évalués sur des graphes

complexes.

6) Intégration des algorithmes : Suivant l'architecture issue de l'ETSI qui concerne la virtualisation des fonctions de réseau, on propose l'intégration des algorithmes précédemment conçus et évalués dans une architecture réelle. On propose un diagramme de flot et met en oeuvre un prototype basé sur les logiciels opensources tels que OpenStack, OpenDaylight et Open Virtual Switch. Cette partie a été démontrée à travers une implémentaion réelle et elle a donné de très bons résultats.

Cette thèse est soutenue par des modèles mathématiques, des implémentations et des simulations qui fournissent des résultats qui mettent en valeur notre travail, quantifient l'importance de l'évaluation des techniques d'optimisation et analysent le compromis entre la réduction du coûts de l'opérateur (la migration) et l'amélioration de l'indice de satisfaction de l'utilisateur final.

Mots clés- RDC ; Optimisation ; SDN/NFV ; Algorithmes de migration ; Algorithmes d'évolutivité.

Acknowledgments

I would like to thank the people who have directly or indirectly supported me and contributed to my work during the years of my Ph.D studies.

Firstly, I would like to express my sincere gratitude to my supervisor and thesis director Prof. Hossam Afifi. His knowledge and expertise have guided me all the way through my research.

I would like to thank as well my co-supervisor Dr. Emad Abd-Elrahman for his motivation, dedication, and patience.

Besides, I would like to thank the members of the dissertation jury for accepting my invitation. I am particularly honored by their presence. I thank the dissertation reviewers Prof. Jean-Pierre Guedon and Prof. Enrico Natalazio for their insightful comments and helpful feedback. I also thank the dissertation examiners Prof. Houda Labiod, Prof. Ahmed Kamal, and and Prof. Hassine Moungla for their time and flexibility.

Last but certainly not least, I would like to thank my parents. Thank you for your continued support, kindly prayers and unconditional love.

I sincerely thank my mother Nadia who loved me very much. She was always beside me during the whole thesis and without her I could not finish the thesis. No words can express my appreciation and gratitude. Love you very much.

I thank my brothers and sisters

I thank my friends ...

Thesis Publications

Journaux

1. Ibn-Khedher, H., Abd-Elrahman, E., Kamal, A. E., & Afifi, H. (2017). OPAC: an optimal placement algorithm for virtual CDN. Computer Networks, 120, 12–27.

Conferences and Workshops

1. Ibn-Khedher, H. & Abd-Elrahman, E. (2017). Cdnaas framework: TOPSIS as multi-criteria decision making for vcdn migration. In 12th international conference on future networks and communications (FNC 2017), july 24-26, 2017, leuven, belgium (pp. 274–281).

2. Ibn-Khedher, H., Abd-Elrahman, E., Afifi, H., & Marot, M. (2017). Optimal and cost efficient algorithm for virtual CDN orchestration. In 42nd IEEE conference on local computer networks, LCN 2017, singapore, october 9-12, 2017 (pp. 61–69).

3. Ibn-Khedher, H., Afifi, H., & Kamal, A. E. (2017). Service placement in complex active networks. In 26th international conference on computer communication and networks, ICCCN 2017, vancouver, bc, canada, july 31 - aug. 3, 2017 (pp. 1–9).

4. Ibn-Khedher, H., Afifi, H., & Moustafa, H. (2017). Optimal placement algorithm (OPA) for iot over ICN. In 2017 IEEE conference on computer communications workshops, INFOCOM workshops, atlanta, ga, usa, may 1-4, 2017 (pp. 372–377).

5. Monteiro, K., Marot, M., & Ibn-Khedher, H. (2017). Review on microgrid communications solutions: a named data networking - fog approach. In 16th annual mediterranean ad hoc networking workshop, med-hoc-net 2017, budva, montenegro, june 28-30, 2017 (pp. 1–8).

6. Ibn-Khedher, H., Abd-Elrahman, E., & Afifi, H. (2016). OMAC: optimal migration algorithm for virtual CDN. In 23rd international conference on telecommunications, ICT 2016, thessaloniki, greece, may 16-18, 2016 (pp. 1–6).

7. Ibn-Khedher, H., Hadji, M., Abd-Elrahman, E., Afifi, H., & Kamal, A. E. (2016). Scalable and cost efficient algorithms for virtual CDN migration. In 41st IEEE conference on local computer networks, LCN 2016, dubai, united arab emirates, november 7-10, 2016 (pp. 112–120).

8. Abd-Elrahman, E., Ibn-Khedher, H., & Afifi, H. (2015). D2D group communications security. In International conference on protocol engineering, ICPE 2015, and international conference on new technologies of distributed systems, NTDS 2015, paris, france, july 22-24, 2015 (pp. 1–6).

9. Abd-Elrahman, E., Ibn-Khedher, H., Afifi, H., & Toukabri, T. (2015). Fast group discovery and non-repudiation in D2D communications using IBE. In International wireless communications and mobile computing conference, IWCMC 2015, dubrovnik, croatia, august 24-28, 2015 (pp. 616–621).

10. Ibn-Khedher, H., Abd-Elrahman, E., Afifi, H., & Forestier, J. (2015). Network issues in virtual machine migration. In International symposium on networks, computers and communications, ISNCC 2015, yasmine hammamet, tunisia, may 13-15, 2015 (pp. 1–6).

Contents

List of Figures

List of Tables

Chapter 1

General introduction

1.1 Motivation

Software Defined Networks (SDN) and Network Function Virtualization (NFV) are two paradigms which aim to virtualize certain network functions while adding more flexibility and increasing the overall network performance. ETSI has highlighted virtualized CDN (vCDN) among the major NFV use cases. The main purpose of vCDN is to allow the operator to dynamically deploy on demand virtual cache nodes to deal with the massive growing amount of video traffic. Scaling in/out, caching as a service etc. are also among the key benefits of vCDN. In this thesis we are motivated to review and address the overall virtualization challenges for CDN transition to vCDN.

Further, literature lacks a general virtualization architecture which leads us to study this network and to design recent architectures that adapt with the network type problems (i.e., placement, migration, orchestration, intermediate node activity, etc.). Furthermore, despite the importance of optimization tasks, such functions are missing in the global architecture. We are therefore, motivated in this thesis to contribute with novel optimization algorithms that may solve the major problems of vCDN deployment in current network operator environment. Coming up with a network simulator tool that help network operator to manage the

deployment of vCDN is also incentive.

1.2 Contribution

Our contributions consist of the the design of a novel virtualization architecture for vCDN. Then, the main network issues in virtual machine migration and especially for vCDN use case and in an SDN/NFV optimization context is proposed.

Further, we propose different optimization algorithms for vCDN. Through the first optimization (i.e., Optimal Placement Algorithm for virtual CDN (OPAC)), we are going to formulate an exact algorithm based on a mathematical model for deciding the optimal location to migrate a vCDN or to instantiate (place) a new vCDN on demand to satisfy users quality requirements. Then, a slight modification to the algorithm is proposed in order to solve the vCDN migration problem. As a result an Optimal Migration Algorithm for virtual CDN (OMAC) is formulated. Further, to cope with scalability problems of exact algorithms, we adapt a heuristic algorithm (i.e., Heuristic Placement Algorithm for virtual CDN (HPAC)) to deal with our constraints when large scale networks need to be optimized. In this algorithm, we exploit the well known Gomory-Hu method to find a near optimum point of operation.

Moreover, as CDN is a set of system components, optimizing CDN placement requires considering the vCDN orchestration problem. Therefore, we contribute by a different optimization algorithm. Through this optimization (Optimal vCDN Orchestration Algorithm), we are going to formulate an exact algorithm based on a linear programming model for deciding the optimal locations to place vCDN components to satisfy users quality requirements and minimize the overall load on the main nodes and links. The major objective from the proposed algorithm is to minimize the total orchestration cost of vCDN components(vCDNC) while minimizing the additional extra-costs needed for caching, streaming, and replicating the virtual instances.

Still, these optimization algorithms are mono-objective. Since vCDN collaborate with two other actors (content provider and end-user), we contribute in this thesis by a multi-criteria decision making technique, TOPSIS. It is adapted also in the context of vCDN migration. Finally, as the optimization algorithms (mono and multi-objective) deal with many heteroclite parameters, a clear view on how/where/when they are extracted and how they are reflected on a typical SDN/NFV architecture is diagrammed.

Among the major contributions in this thesis we highlight the implementation of vCDN Infrastructure Optimization Simulator (vIOS). It is a high level implementation of a network tool that uses the previous optimization algorithms on a real virtualization environment, OpenStack. An integration of this tool in Orange operator shows its feasibility and effectiveness to manage high complex graphs as inputs and recommend server migration lists for instance.

Up to now, network optimization can be easily modeled with classical tools or heuristic methods when the graph becomes too large. Introducing active nodes by means of ICN (or the Hadoop) concept introduces a modification in the graph equivalent to a negative resistance in electronic circuits (where active nodes supply an extra capacity/power if the input capacity measured is negative). Indeed, the ICN (or Hadoop) functions provide some kind of work inside the node. This transforms the two dimensional graph to a three dimensional one where the third dimension comes from the interaction between vertical layers within a node. Therefore, the previous conventional methods (OPAC, OMAC, HPAC, OCPA, and TOPSIS) that can be used to optimize the graph are not anymore valid and we need to look for a more realistic optimization method that can take the third dimension into account. Therefore, in this thesis, we contribute by a proposed model and optimization algorithms that target general scenarios over ICN complex active network where active nodes add a third dimension to the traditional network topology representing local work (e.g., augmented reality, data mining, adaptation, publicity insertion, IoT data timing, etc.). Once the network flow of the ICN

active network is defined, optimization algorithms aim to maximize the dynamic data rates under a total ICN budget.

Up to now, the most particular attention is being set to the exact or approximate approaches with performance guarantees on which very significant progress has been made in recent years and to a central problem in network flows which consists of calculating paths to steer/route traffic flows (demands) while minimizing congestion links (ratio of active traffic on capacity). For the best of your knowledge, the current implementation of these approaches are fully decentralized (i.e., a controller that collects topology parameters, live OpenStack statistics/metrics and user demands for a vCDN Service as input and recommends some decisions). In order to bybass the offline optimization that Gomory-Hu introduces when checking potential server migration and decentralize our optimization techniques, we contribute by a distributed maximum concurrent flow algorithm. It is inserted on each network nodes, collect user demands from children nodes and routes the maximum fraction of flow to multi-terminal while not exceeding the neighbor links capacities.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 describes the transition from CDN to vCDN. Chapter 3 analysis the overall networking issues in CDN virtualization and migration. Chapter 4 describes the optimal placement algorithm for vCDN, its parameters, constraints and objective function. Chapter 5 details our exact and heuristic optimization algorithms for vCDN migration. Chapter 6 describes the orchestration algorithm (OCPA). Chapter 7 presents the multi-criteria decision making algorithm. Chapter 8 proposes a novel optimization technique over complex active networks. Chapter 9 concludes the thesis and discusses the future optimization techniques in virtualization. Given the limit size of the thesis, additional document is added at the end which represents further contributions (Annexes A, B, C, D, E, F).

Chapter 2

From CDN to vCDN: state of the art

2.1 Introduction

Content Distribution Network (CDN) is a large distributed network deployed worldwide designed to push the content on the edge of the networks (i.e. Point of Presence (PoP)). It solves two main problems that enhance the network performance: *i*) network latency for service access and *ii*) load balancing. CDN deployment is a major concern for network operators and service providers as video (either VoD or Live TV) represents more than half of the total Internet traffic (66% as per Orange) and will grow to more than 80% of all Internet traffic by 2019 [1].

The CDN moves content to mitigate increase in video traffic and enhance user satisfaction. It finds an intermediate point between the origin server and the clients for the replication. Network virtualization features, with new emerging concepts such as Network Function Virtualization (NFV) and Software Defined Network (SDN) solutions can answer users demands and, subsequently increase the network efficiency. Therefore, we look at the potential benefits of virtualization brought to CDNs. We tailor our state of the art to provide:

- A comprehensive state of the art on the traditional CDNs.
- Strength and weakness of different CDN services when used with different

[1] http:/www.cisco.com/c/en/us/

caching and replication algorithms including virtual ones.

- A new approach taking into account virtual content delivery networks through Network Function Virtualization (NFV) and Software Defined Networking (SDN) concepts.

CDN today is the result of many contributions that have improved the architecture and the design. So, we try to organize those contributions in a categorized way, easy to understand. Also, we include surveys of the most performing caching algorithms used in CDN systems.

The rest of this Chapter is organized as follows: Section 2.2 highlights the common architectures for CDN. Section 2.3 describes the main taxonomies of CDN. We study the evolution from CDN to virtualization aspects in Section 2.4 and the transition from physical cache to virtual ones with details about caching techniques. Section 2.5 introduces the state of the art of the art the main virtual CDN solutions with details about our virtual CDN (vCDN) use cases based on network function virtualization principles: NFV and SDN. Section 2.6 highlights general optimization techniques used in the context of virtualization and video delivery service. Finally, this chapter is concluded in Section 2.7.

2.2 CDN overview

Fig. 2-1 describes in a general way the different components of CDN. A typical CDN includes a content distribution system which has the role of content replication. In addition, a request routing system is necessary to receive requests from end users and redirect them in different ways discussed later. Further, a centralized management plan is used to distribute load over surrogates (it is defined as a device/function that interacts with other elements of the CDN for the control and distribution of content within the CDN and interacts with User Agents for the delivery of the content)[2] (CDN Mgmt). It has the role of cache management.

[2]According to Internet IETF Draft, draft-jenkins-alto-cdn-use-cases-03

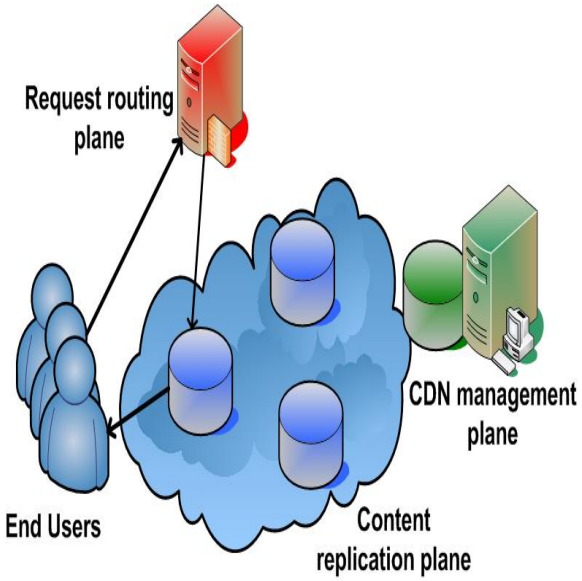


Figure 2-1 – Main components of CDN

Those main components of a typical CDN can be defined as the following:

- Content delivery infrastructure [135]: It manages cache servers responsible for delivering the content to the end users.
- Request routing infrastructure: Its function is to redirect client's queries to the most appropriate CDN cache. Also, it interacts with the distribution infrastructure to make a global view of the up-to-date content stored in edge servers.
- Distribution infrastructure [135] Its function is to move content from the origin server to the CDN edge cache servers ensuring the consistency of cached data.
- In addition to the previous three plans, the accounting infrastructure [135] maintains the logs of client access and records the CDN server's utilization in order to prepare the billing records.

Several works have investigated the topic of content delivery networks. Peng et al. [136] gave an overview of CDN and introduced the encountered problems in

building an efficient architecture. Further, they enumerated and proposed solutions to overcome such issues. Vakali et al. [173] presented a survey on the CDN architecture and gave the benefits of CDNs. Dilly et al. [51] and Akamai [4] gave a state of the art of the existing CDN approaches. They also detailed the infrastructure of the Akamai network (currently, the most developed caching infrastructure) and how it works. Kung et al. [134] proposed taxonomy for CDN classification and gave some proposed approaches for content aggregation and content placement. Saroiu et al. [154] investigated CDN based on four content distributed systems: HTTP web traffic, Akamai, Kazaa, Gnutella and they gave significant implication for large organizations. Honguk et al. [177] proposed a virtualized, programmable content delivery network, in which the CDN is proposed as a service or on demand. CDNs, sometimes also called Content Distribution Networks, are often deployed close to the end user when he makes data requests. Static content may be images, video, CSS files, scripts files, etc CDN customers are mainly Internet Service Providers (ISP) and Over the Top (OTT) service providers like YouTube and Netflix. They push the content to the edge servers over IP. Typically, they are looking to build a large farm of cache servers deployed worldwide in order to reduce the load on the OTT servers, reduce traffic overhead and provide the following services:

- Storage (popularity based)
- Management of cached content (i.e. cache management like replication, replacement, and placement)
- Distribution of content among cache servers (load balancing)
- Content Streaming
- Fault tolerance (i.e. Redundancy)
- Network performance (reduce overloads)

2.3 Taxonomy of CDN

CDN is considered as a center of many interests. The literature treats in priority the following aspects:

- CDN infrastructure and composition.
- CDN delivery and management.
- CDN request routing.
- CDN performance and user satisfaction.

2.3.1 CDN infrastructure and composition

Lazard and al. [110] introduced that CDN infrastructure can follow the overlay approach and cited CDNs deployed by companies as an example, i.e. using Peer to Peer (P2P) applications or an underlay of connected servers. In that case, the original content is always permanently stored in the underlay and copied on occasions to the dynamic substrate. We found several different compositions of CDN as shown in Table 2.1. Indeed, in the underlay method, caching nodes are centralized while in overlay method CDN can be assisted with P2P and other heterogeneous or decentralized methods. Further, we summarize the interaction between the different components of the CDN in two ways: either through the use of surrogates or through caching proxies.

Table 2.1 – CDN infrastructure and compositions

CDN composition issues	Taxonomy	
CDN infrastructure	Underlay	Overlay
CDN servers	Origin	Surrogate (Replica Server)
CDN content type	Static or embedded	Dynamic
CDN cache protocols	Intra-cache: NECP, WCCP, SOCKS	Inter-cache: CARP, ICP, HTCP, Cache digest
CDN Interaction	Caching proxy	Surrogates and subnet

2.3.2 CDN delivery and management

In this section, we investigated the issues of distributing CDN service to end-users and managing the delivery network. Thus, we are trying to study problems like replica placement, content selection in CDN, content outsourcing, content caching techniques, and content update.

Replica server placement

In this issue, CDN administrator decides on how and where he should deploy its own replica servers. In the literature CDN providers develop two main algorithms [173] for determining the optimal number of replica server placement: Single-ISP and Multi-ISP algorithms. In the second case, it is a multi-criteria optimization problem that may be difficult to solve.

Content selection

The selection of content targets to be copied is an important issue in CDN as it affects cache efficiency. Two approaches are proposed: full site and partial site approaches. In full site case, the replica server is responsible for delivering the entire content to the end users. In partial site, replica servers are responsible only for delivering the embedded object such as web page images. It is useful when the partial content is static (such as commercial catalogs).

Content outsourcing

The process of outsourcing content in CDN can be invoked after the decision about how many servers will be deployed and after the selection of content to be copied. Content outsourcing can be achieved through three methods: cooperative push based, cooperative pull based and non cooperative pull based:

- In Cooperative push based, content is pushed to the overlay network and user requests are redirected to the closest replica server or served directly by the

origin server (in the case that there is no content replication in CDN overlay network) [100], [67], and [36].

- In Cooperative pull based approach, content requests are redirected to surrogate servers and when there is a cache miss, replica servers cooperate to serve the end user by the desired content based on distributed index structure such as Distributed Hash Table (DHT) [66].
- In the Non cooperative pull based approach, after one of the following algorithms of request routing (DNS based algorithm, HTTP based algorithm, URL based algorithm), user requests are redirected to their closest replica servers. The main difference between this method and the previous is in case of cache miss. In this method, there is no cooperation between cache servers and replica servers to pull the desired content directly from the origin server [54] [133] [98].

Caching techniques

In caching, we highlight two main approaches: content replication and content update mechanisms. Those techniques are considered as the main role of any CDN network and they define its performance.

1. Content replication and caching [135]: In general, CDN infrastructure is a combination of replicated web server clusters. Once we reach a large scale of distributed surrogates content replication and caching have to be mandatory. Hence, Akamai and other commercial CDNs replicate content across this overlay infrastructure with one of the following approaches
    - In Intra-cache approach, we use one of the following schemes proposed in the literature: query based schemes, digest based schemes, directory based schemes, and hashing based schemes.
    - In Inter-cache approach, we need an inter-routing between different clusters (group of caches).

This algorithm is invoked when the content does not exist in the current cluster. We will detail in the next section the caching techniques in video service delivery.

2. Caching content update [136]: The update process is based on content expiration time (eg., Time-to-Live (TTL)). Four algorithms have been proposed for content updates which are periodic update, update propagation, on demand update and invalidation. In periodic update, content is pushed automatically with a time to live. After time expiration, surrogates must check back the origin server. This approach suffers from high traffic requirement since the update process is static. In update propagation, origin streaming servers deliver always the updated content whenever a video document has been changed. This mechanism generates high update traffic for frequently changing video content. In on demand-update, video document is updated after a prior user demand. The drawback of this cache update mechanism is the excess traffic generation between the surrogate and the origin server in order to ensure that origin server has delivered the updated version of the video content. The latest cache update mechanism is invalidation in which origin server broadcast an invalidation message to all surrogate servers when a video content is changed or updated at the origin server. The disadvantage of this approach is that cache nodes have to seek for an updated version of the video individually after that video has been changed. This mechanism suffers also from inefficiency of content management and coherence among cache nodes.

Table 2.2 summarizes the most important management and delivery issues in CDN as mentioned before.

2.3.3 CDN request routing issues

Request routing redirects content requests from users to the nearest or best replica server based on load, capacity, and other metrics. In the literature, several

Table 2.2 – CDN delivery and management

CDN delivery and management	Taxonomy	
Replica servers placement	Single-ISP	Multi-ISP
Selection of content	Full site content delivery	Partial site content delivery
Outsourcing of content	Cooperative push based	(Non)Cooperative pull based
Caching techniques	Content replication and caching	Caching content update

metrics are used for request routing such as closeness, latency, distance, and load.

In [159] authors showed that the proximity factors used for serving clients don't always give the best performance. Hence, load and context information should be considered before switching to a replica server. Recall from the management of content that there are two ways of replication: full and partial site replication. If we use the first approach, the origin server redirects the content requests to the replica server. In the other approach, the origin server sends basic and embedded objects to the replica server and the latter redirects the response to the client. Request routing systems have two parts:

- Request routing algorithm: how to choose the replica server after that a content provider receives a content request from the client.
- Request routing mechanism: the protocol, procedure or steps to inform the client (user) about the chosen replica server.

In Table 2.3, we show the related issues of CDN request routing procedures to redirect user requests to the optimal (nearby) CDN server.

Request routing algorithms

The main request routing algorithms are enumerated as follows:

1. Adaptive algorithms: The major request routing algorithms found under this technique are:

Table 2.3 – CDN Request routing techniques

CDN routing issues	Routing techniques					
Routing algorithms	Non-adaptive			Adaptive		
Routing mechanisms	GLSB	DNS	HTTP	URL	Anycasting	CDN peering

- Round Robin Algorithm (RR): In this algorithm [130], requests are distributed to the CDN replica servers and each replica has a maximum load for serving requests. Authors showed that this approach assures load balancing, assuming a similarity among all the CDN replica servers (same capacity, same storage, etc...). This approach has a lot of disadvantages because it does not take into account the distance among replica servers and users. Further, authors in [131] showed also that through RR algorithm, client may be served by more distant CDN while the nearest replica server was not loaded. Hence, we conclude that RR algorithm fits well when all the replica servers are in proximity (i.e., replica servers form one cluster).

- Predicted Load Algorithm: Predicted load is based on the number of requests on server recently served (served so far). This algorithm assures load balancing (load is distributed among replica servers) taking into consideration the distance between the client and the replica. Therefore, load and distance are the two essential metrics used in this algorithm [162].

- Random Request: In this algorithm, content provider redirects randomly content requests to replica servers [45].

- Statistics based redirection: In this algorithm, we calculate the percentage of client's request per CDN. Then, we always redirect content request to the CDN which receives the greatest request number [45].

- Geographic location: K. Delgadillo [45][3] mentioned that some Cisco algorithms found in the literature are based on the geographic location of the request origin. In this approach, the redirection to a replica server is based on minimum distance between the provider and the consumer. However, the process of redirecting the user's content request to the closest CDN replica server is not effective because close replica servers may be overloaded. Moreover, when serving cellular users (smart phones, tabs, etc), the origin IP address does not reflect the real mobile location.
- Hash function based algorithm: In [102] authors Proposed a hash function based solution for redirecting requests to replica servers. In their approach, they calculate the hash function of the URL of the content and then redirect the request to the replica server represented by the ID which is larger than the calculated hash function. This algorithm has many variations in P2P file sharing algorithms [17] and intra cluster caching [127] [92].

2. Non Adaptive algorithm: In the literature we can find several works that investigated non adaptive request routing algorithms:

   - Network proximity: Pierre et al. [140] described an algorithm based on network proximity to redirect user requests to replica servers [30]. They measure the path length and update it periodically. This approach based on the distance as a metric is not really effective as explained in [110].
   - Latency: Andrews et al. [8] and Ardaiz et al. [11], proposed algorithms based on the measured latency between the provider and client. In those approaches, client requests are redirected to the replica server reporting minimum latency to users. This algorithm is very efficient but it requires extensive measurements.
   - Weighted combination of latency and distance: Delgadillo et al. [45] in-

[3]Cisco Distributed Director, Cisco White Paper, Cisco Systems

troduced also a combination of three metric called: intra AS distance, inter AS distance, and end to end latency, to decide for the best target. The measurement of those metrics makes the algorithm more complex since it adds an additional signaling traffic.

- Akamai algorithm: Peng et al. [136] And Vakali et al. [173], proposed an adaptive request routing algorithm based on three metrics: load on the replica server, reliability between client and all the replica servers and bandwidth remaining on the replica server. Akamai's algorithm is complex but robust.

Request routing mechanisms

The Request routing mechanisms found in the literature are as follows:

1. Global Server Load balancing (GSLB): In the Global Server [4] mechanism, a group of GSLB switches distributed around the world are responsible for redirecting user requests to a web server in order to assure load balancing among replica servers.

2. DNS-based request routing: In DNS-based solution for request routing, surrogate names are mapped to IP addresses (resolution process) and when a client sends his request, he receives a number of names of surrogate servers that contain the content. Then, he will send his content request to one of them. The choice of the best replica server is currently based on probe requests sending to those replicas to deduct which replica among them is the best (which has the fastest response time). Request routing based on DNS algorithm generates high latency needed for search and lookup process used to build the list of replicas. Despite the proposed solutions to reduce the response time [107], this approach is still insufficient and has many disadvantages. Indeed, it does not take into account the proximity parameter when

[4] https://www.a10networks.com/products/global-server-load-balancing-gslb

redirecting queries (content requests) and cannot guess contextual information about the user. We have to recall that DNS is a simple and straightforward name resolution so it can not provide any sophisticated routing procedures. As shown in Fig. 2-2, the end user should request for the home page of the origin server to receive a response including the desired URI (step 1, 2) that serves to seek for the specified domain name with its local DNS (step 3). Then, the local DNS Server sends a DNS Query by passing a DNS request in order to resolve and get the IP address for the Host Name = csp123.cdn.kt.com (steps 4, 5, 6, and 8). Moreover, the request router selects the optimal cache server based on the geographic proximity based on local DNS server IP and send immediately the IP address of the Cache Server (step 7) to the end user. Finally, the user terminal sends a Content Request (eg. HTTP GET) to the Cache Server Content (eg video files) to download / streaming receive (steps 9 and 10). In this way there is no dependency for the request routing process with a user's location information to select the Cache Server which is a the most important disadvantage as user may be far from the streaming point.

3. HTTP based request routing: In HTTP based request routing [5], clients send their requests to the web server. Then, through the HTTP protocol, the origin server redirects the client queries to another server holding the content. This algorithm incurs some latency because of the many round trips between user, HTTP web server, and replica server which influence the quality of experience. Therefore, user's satisfaction can decrease. It is however the mostly used in video delivery.

4. Anycasting based request routing: In Anycasting based request routing, user requests are sent to one server that manages the anycast address so as to redirect user requests on replica servers. In this algorithm, a group of hosts has the same IP address (global IP address) allocated on demand by the network

[5]https://tools.ietf.org/html/draft-ietf-cdni-redirection-13

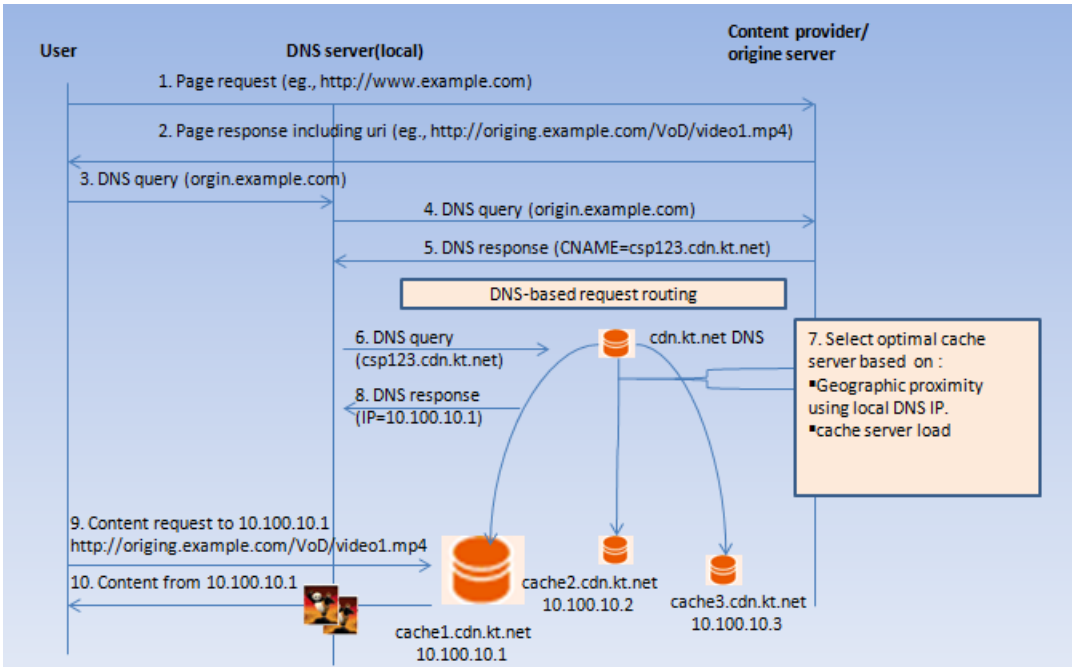


Figure 2-2 – DNS-based request routing

administrator. Furthermore, in this architecture, the routing is made through the closest IP router to a host among the group of replicated servers. This approach needs an extra space of IP addresses, and suffers from routing performance issues (because the anycast) that should be considered for request routing.

5. URL rewriting request routing: URL approach also called navigation hyper link, origin servers redirect content requests to different replica servers holding the embedded object. In this process, URL links of embedded object (images, CSS files, scripts) are dynamically rewritten and generated. Clients are then redirected to those surrogates. This algorithm is adapted to partial site content delivery in which origin server serves the original HTML page while the embedded content is retrieved from the cache servers. Youtube uses a routing technique close to this approach.

6. CDN peering based request routing: In CDN peering based request routing,

Table 2.4 – CDN performance

CDN performance issues	Taxonomy	
Network procurement	Dependent replica server	Independent replica server
Simulation	Simulation tools	

a content delivery network based on symmetric connections between peered CDNs is built in order to serve content to the end user. This approach can be considered as a CDN assisted with P2P overlay network. The goal of this algorithm is to increase fault tolerance and decrease the latency. However, since it is a distributed algorithm, it suffers from scalability and churn problems.

2.3.4 CDN performance and user satisfaction

The performance measurement of the ability of CDN to serve their customers is based on Hardware and software probes or on CDN's logs. Clients when requesting for a video, generate a feedback from the overlay infrastructure andn precisely from the CDN replica servers that indicate CDN performance measurement. Table 2.4. shows the related issues of CDN performance.

We deduce from the investigated state of the art that performance measurement can be achieved either through network procurement (collecting) or through simulation.

Network procurement

In network procurement, performance measurement uses either dependent or independent replica servers that collect the required metrics. In dependent replica servers, the surrogate server is equipped with a process calculating end-to-end metrics. This can be done either by the collection of the logs from the surrogate servers or through the deployment of probes (Hardware or software). In inde-

pendent replica servers performance measurement is achieved by an independent third party that tells clients about the CDN replica servers performance.

In Network procurement, performance is based on the estimation of some metrics such as startup time, time to first byte, server load, jitter, geographic location, and packet loss. More sophisticated metrics can be subjective such as user engagement (i.e. duration of the video viewing by the end-user).

Simulation

In addition to the use of network procurement for measuring network and system performance, researchers resort to simulation tools to measure the performance of CDN because of the proprietary nature of commercial CDNs. Furthermore, many simulation tools are available to measure the performance of CDN [36] and [100]. The use of these tools has some advantages such as the simplicity and the flexibility of implementation in analysis of results. However, those results may be unrealistic since the simulation tools do not take into consideration all the critical factors.

2.4 Network evolution from CDN to vCDN

As our thesis concentrates on virtualization and server optimization, and after investigating the state of the art of CDN, we found that CDNs can be additionally divided into two categories: physical CDN and virtual CDN:

- Physical CDNs: Physical CDNs are divided into commercial and academic development based CDNs. For the commercial CDNs we have Akamai [4], Accellion [2], Lime Light Networks, Appstream, Globix and EdgeStream. For the academic CDNs we must mention CoDeeN [38], COMODIN [40], Coral, and Globule. Physical CDNs have many service types such as file delivery, network service, and video streaming service (live and on demand video) and so on.

- Virtual CDNs: In the literature we have few contributions on CDN assisted with cloud and network virtualization. We have MetaCDN [30] based on edge cloud servers and Active CDN [163] based on edge servers as software modules.

Caching taxonomy for video delivery In the literature, we found many researches and contributions that focus on caching service in order to enhance the performance of the network. For multimedia and video applications, network performance is a function of several parameters such as: video popularity, complexity, congestion, video size, hit rate, quality of experience, cache replacement algorithm and cache update mechanism. According to these metrics, we will compare the work of the state of the art detailed in the two Appendixes A and B.

2.5 Virtualization of CDN

Network Functions Virtualization concept covers several areas. Virtualization of CDNs is important and compulsory especially after the evolution of video traffic delivery as seen in this survey since we have observed the evolution of the CDN architectures till the appearing of NFV and SDN virtualization concepts. For a variety of reasons, such as QoE, QoS, management and scalability, vCDN solutions start to appear.

We need first to differentiate between NFV and SDN. NFV is introduced as a new methodology or concept that decouples network application from the underlying hardware and enables network services based on NFV solutions to run in a virtualized environment. SDN is the networking paradigm that decouples the control plane from the data plane and centralizes the control plane in one entity which is the SDN controller. Both systems can work together to enhance the network operation and management, improve QoE of end users and the overall quality of service and network performance.

So to virtualize the CDN, we believe that the process can be achieved through

one of the following techniques: cloudification, virtualization, and programmability.

2.5.1 Cloudified CDN

In CDN based cloud computing approach we quote MetaCDN [30] that used an existing cloud infrastructure to deliver video service to end users. It implements several CDN services in the cloud to enhance the delivery process such as replication strategies, load balancing etc. However, it still uses preconfigured and physical edge servers deployed on cloud infrastructure through Infrastructure as s Service already deployed in an OpenStack open source project (IaaS).

Further, we cite ActiveCDN [163] that uses the same previous cloud computing architecture to virtualized CDN service such as caching and acceleration techniques mainly for video delivery purpose. However, edge servers in this cloud computing approach are software based solutions. Those network functions are dynamically deployed on CDN caching nodes. The advantage of this approach is that it facilitates the removal, adding and changing of network functions through cloud monitoring systems.

CoDaaS [97] is another solution to enable the virtualization of multimedia services and CDN network function. Further, it lets end users to generate their contents into the cloud. Furthermore, content providers rent a part of the cloud resources such as bandwidth, storage, and so on so as to host their contents in the cloud. This approach is better in the direction of user involvement for service video delivery. Florian et al. [55] proposed a virtual architecture for implementing CDN services in the cloud environment. The main disadvantage is that CDN as a service proposed in their work missed more features like dynamic management of virtual CDNs, routing policy, and so on.

Tai-Won et al. [169] proposed cloud based virtual CDN architecture. They proposed a control plane formed by a cloud broker that configures a set of virtual machines (vCDNs). In their approach, content providers deliver their content to

end users through a virtualization layer. Therefore, they proposed some entities in the control plane to monitor the virtualization layer and fulfill the service level agreement (SLA) needed for achieving media delivery service. We can conclude that these proposals are all cloud oriented CDN solutions strongly dependent on a cloud infrastructure.

2.5.2 Virtualized CDN

In this section, we illustrate the few works that have been done for CDNs based on virtualization (NFV) concepts.

Honguk et al. [177] describe a new strategy for cache deployment. They propose a hierarchical tree structure for virtual content distribution network systems. It is based on a tree construction algorithm. It can reduce content distribution times and minimize source streaming data rates by dividing the network in levels: level1 for local cache and level2 for regional cache. The disadvantages of their strategy are multiples: first of all, they do not give a clear concept for the virtualization. Secondly, the algorithm used for content chunks diffusion is static and do not take into account content popularity or size. Thirdly, it requires a lot of computational overhead since content are streamed per chunk without any consideration of content chunk size.

Niels et al. [28] provide a framework for multimedia delivery (CDN) based on NFV. They gave a structure of the point of presence, PoP deployment within the network. They used layers (core network, aggregation layer, access layer) to seek for the optimal location of cache nodes. Their work missed an optimization algorithm to measure the QoS/QoE. Further, they don not take into consideration the cooperation between CDN edge servers to serve user requests. Also, the optimization equation missed a lot of constraints such as the maximum server storage and the migration of content from one server to the optimal one.

In ETSI [170], authors introduced the virtualization of CDN (vCDN) by giving a principle of different distributed cache nodes deployment in the virtual environ-

ment. However, the overall architecture of vCDN containing the request routing system, the content management, the distribution system, and the accounting system are still missed in this conceptual architecture.

Mangili et al. [116] introduced the state of the art of NFV, CDN, and stochastic optimization techniques. Then, they use a mixed architecture where real and virtual CDN nodes coexist.

2.5.3 Programmable CDN

Honguk et al. [177] introduced an open framework for CDN which relied on SDN. They use this paradigm for CDN to program the overlay network constituted by distributed edge servers. They proposed four components: request router, edge server, controller, and a monitor. They use also SDN for monitoring the distributed cloud resources (data plane) and monitoring the process of content delivery. In control plane, they used control modules for security, caching, and routing that interact with an OpenFlow controller. The contribution has also introduced an optimization mechanism for increasing the network throughput and the overall QoS.

In next section, we are going to study the optimization techniques combined with virtualization and delivery contexts.

2.6 Virtual CDN optimization

In combinatorial optimization (e.g., network flow problem, travelling salesman problem, etc.), we quote exact and heuristic/approximation algorithms as the two principles branches. Most of these optimization problems are Not Polynomial (NP). This means that those problems are difficult to be solved (in polynomial time) but it is easy to check a given candidate solution (yes/no answer). Therefore, heuristic/approximation algorithms are proposed as an alternative to solve large instance numbers of the optimization problem . They are analyzed mainly through the running time in different use cases (empirical, average, and worst

cases). In virtualization and delivery contexts, literature highlights relevant optimization algorithms and techniques.

2.6.1 NFV/SDN common optimization techniques

This subsection highlights the relevant NFV/SDN optimization techniques. Thus, facing the difficult combinatorial optimization problem in the sense of the theory of algorithmic complexity, there are two main solutions for the Virtual Machine (VM)/Virtual Network Function (VNF) optimization problem:

1. Exact approaches (optimal algorithms)
2. Heuristic approaches (e.g., best-fit decreasing, first-fit decreasing, genetic, and meta-heuristic algorithms)

2.6.2 NFV/SDN optimization algorithms in the video delivery context

This subsection highlights the relevant NFV/SDN optimization algorithms in the video delivery context.

I. Trajkovska et al. [71] leverage SDN to build their management framework in order to enhance the overall QoS of the delivery process. They showed that SDN paradigm enhances network management for video service delivery and brings flexibility and programmability to the route calculation while considering the QoS. Moreover, they used a control algorithm based on bandwidth metric calculation. The proposed approach has to take into account the QoE parameters which is necessary in video streaming and it missed different SDN/NFV criteria such as node throughput, service orchestration, and so on.

P. Georgopoulos et al. [71] introduced the OpenFlow protocol assisted with QoE Fairness Framework (QFF) in order to optimize the overage QoE of end users connecting to the home networks. Their framework considers monitoring the video delivery service. Therefore, they take into account the gathered network information and end users profiles in order to build their QFF. In their implementation, they used two standards for audio visual content delivery and management

resources in network operators which are OpenFlow and MPAG-DASH. In their work, there is no correlation between the video bit rate and the QoE perceived by the end user. To optimize the efficiency of the network and assure a high QoE, authors proposed a new framework in which they quantified the non-linear mapping between the perceived QoE and video bit rate through an utility function. Indeed, the utility function used is an exponential curve which binds a video bit rate at a fixed resolution to a perceived QoE. Moreover, they used an objective video quality assessment models (SSIM) in order to build the utility function. For the optimization function used in their work, they exploited the utility function in order to find the optimum set of bit rates that assure QoE Fairness requirements. Further they added two exact optimization algorithms called promote in order to upgrade user with the minimum video quality. In their proposed approach, the utility function does not cover the context parameters but it is based only on video bit rate. Moreover, the approach does not involve the NFV QoE context.

H. E. Egilmez et al. [57] described a new approach for design an SDN controller in order to enhance the multimedia delivery process. They introduced a novel QoS method relying on dynamic routing from the server to the customer. The approach aims to assure the required network performance for multimedia traffic. Beside the IntServ and DiffServ QoS architecture, SDN approach [6], is proposed in their architecture as a new entity to program and adapt the QoS. SDN-Based dynamic routing using OpenFlow protocol achieves resource monitoring, network visibility, and network virtualization. However, it requires a global network view and may requires another system control for system management. Authors showed that the networking problems can be solved through OpenFlow based solution. Therefore, they conceives a new architecture based on OpenFlow to enhance the QoS for service video delivery and introduced an optimization framework for dynamic QoS. they proposed a new design for the controller by adding some algorithms within it (rules, policies) for the communication with the underlying layer decomposed

[6] which decouples the control plane from the data plane and consolidates the control plane to a centralized entity

by OpenFlow switches/devices. Therefore, the QoS based on resource reservation (IntServ) or in priority queuing (DiffServ) is not considered. An SDN based QoS approach is hence proposed as another IP scheme based on dynamic QoS routing. Authors proposed in their SDN controller design, a novel application module that collects network states from switches in order to run the optimization algorithm. The optimization in their approach based on CSP problem and it is resolved by LARAC algorithm.

M. Diallo et al. [46] investigated the state of the art of the QoE and proposed two main techniques for QoE optimization in the context of (SDN/NFV) video which are MDASH and utility based approaches. A global formula is extracted after a subjective process which we will use as a mapping between network resources and perceived quality. Despite this relevant work, it missed the integration of VNF criteria and the is no usage of ICN as an accelerated engine foe the forwarding plane.

In [1] authors introduced the minimum spanning tree (MST) algorithm to optimize the network resources allocation. They enhanced the optimization procedure to deal with the video delivery context in CDNs. Further, an amended MST algorithm by inserting virtual nodes to the initial tree and removing them after algorithm re-execution in order to decide the new streaming sources while keeping the overall cost of the multicasting sub-trees minimized is proposed. Then, they leveraged the QoS as a link metric in their initial graph topology. Then, they bind the QoS to the QoE in a stochastic model in order to approximate the end user experience. Their model is still missed different QoE criteria and does not leverage the virtualization of nodes or links in a delivery network such as CDN. Moreover, the mapping equation of QoE-QoS is not given.

Moreover, optimizing vCDN node deployment and delivery requires QoS/QoE measurement. Hereafter, we quote the relevant work.

2.6.3 QoE measurement for NFV delivery context

In this subsection, we detail the main work investigated in this topic.

H. E. Egilmez et al. [56] formulated an optimization framework for dynamic QoE routing as a constrained short path (CSP) problem. Then, they used the jitter metric for calculating the overall QoS. Further, they minimized the overall cost under the constraint of delay variation in the main intermediate links. Authors affirm that the optimization problem is NP-hard. Therefore, they used a LARAC algorithm based on Dijkstra routing algorithm.

H. Nam et al [124] leveraged also a Constrained Short Path First (CSPF) path selection algorithms over MPLS domain. The algorithm is adapted in order to choose the best routing path for each streaming flow. Authors added a video optimization server to act as an SDN/NFV network application. It aims to minimize the streaming cost and increase the average QoE. It is a simple SDN application that communicates with the network controller. Then, authors consider a delivery network of CDN nodes that send a feedback to the SDN application using CSPF over MPLS in the case of buffering. Moreover, in a congestion situation, the SDN controller redirects the network flows to another CDN node using open-flow based rules. In their approach, CDN delivery nodes send periodically QoE measurement directly from the end user to the SDN controller. The video QoE metrics leveraged in their method are video start-up, latency, buffering rate, and play-out buffer status. Authors used then MPLS Traffic engineering (MPLS-TE) over SDN layer in order to enhance the overall QoS. They used CSPF algorithm with MPLS in order to select the best path from CDN delivery node to the client. They described three constraints which are TCP bandwidth, packet loss rate and jitter. When the QoS becomes very bad, SDN controller runs the CSPF algorithm in order to find the CSP. They added also some rules for the previous algorithm in order to maximize the QoS. Their scheme leverage SDN paradigm with MPLS in an efficient way, however different NFV criteria such as vCDN node resources (vCPU, vRAM, vStorage) and vCDN links are missed. Moreover, optimizing QoE has to consider

vCDN node/link caching and streaming constraints.

S. Ramakrishnan et al. [147] affirmed that QoE measurement is strongly related to the network resources utilization. Further, they proposed a solution based on SDN paradigm in which the streamer, the network, and the consumer are working together to improve the average user's QoE. Furthermore, they added an SDN network application for video quality assessment (VQA). It gathered information from network entities through a specific router. VQA calculates the average QoE required through three main steps: 1) firstly, it measures the average QoE according to the bitrate achieved by streaming flows, 2) then, it uses an objective QoE (e.g., SSIM, VQM, PSNR) method in order to recalculate the QoE, 3) secondly, it collects the previous QoE values and adds some parameters related to the content and the device, 4) finally, it sends the QoE to the network operator represented by SDN controller to update the input of VQA.

In this approach, authors enhanced the QoE metric by considering some characteristics related to data and device types. VQA allocates bandwidth per flow based on device/encoder type in use and video content complexity. We believe that the SDN-based QoE measurement has to consider also the NFV requirements and take into account all the virtualization criteria which is missed in the proposed approach.

M. Diallo et al. [48] introduced an hybrid conceptual perception model (CPM) for streamed video quality assessment. Authors mentioned that QoE of end user can be measured through two main methods which are objective methods based on network parameters and subjective methods based on user's perception of the giving service. Further, they proposed a third method for measuring the QoE that mixes the two previous methods and aims to provide a better QoE. Indeed, they considered different contexts to measure the QoE related to the user, the device, the content, and the network. They provided a simple model that measures the MOS by mean of a simple exponential function.

2.6.4 Discussion and future work

We have compared the CDN architectures and types based on different taxonomies. Despite the very large number of contributions, we believe that many issues are still open for improvement and research. The specification of the OpenFlow protocol brings new potential improvements to the global SDN infrastructure especially in the control/management plane. Moreover, the virtualization has brought its own problems mixing network and system constraints. Literature lacks an optimization algorithm that introduces at the same time system, network, and user quality parameters. Moreover, none of the above work introduced a placement and/or migration protocol for virtual content delivery network functions that include the above constraints. The novel virtualization business model involving new actors (User, operator and content provider) requires additional optimization techniques. We believe that measuring the QoE/QoS is an important task for service video delivery and it required an enhanced model based on these different information context which assures video adaptation and high QoE/QoS for the end-user. Moreover, the QoS/QoE model has to be integrated in the SDN/NFV framework and leverages an accelerated data plane such as network information ICN, virtual caching and routing. In addition, novel optimization techniques that aim to satisfy the main stockholders in the delivery chain (NFV customer, NFV/ICN operator, NFV customer's end user) is then elaborated and solved. Hence, we contribute in this thesis by optimization protocol beside an optimization model.

2.7 Conclusion

This chapter has surveyed research investigated on content delivery systems, mechanisms and algorithms. We classified and detailed the most important CDN systems. Further we followed the evolution of CDN to be virtual (vCDN). We classified then the most important caching techniques in both real and virtual CDNs.

Then we surveyed the virtualization of CDNs. We may conclude that the virtual delivery area is active and supports different caching techniques. There are still many unresolved issues dealing with caching, request routing, streaming and link utilization techniques. To date, and based on the CDN-NFV architecture proposed by the ETSI, aspects such as mobility or routing require extra attention and lack communications protocols.

In the next chapter, we are going to highlight the virtualization challenges in vCDN deployment using an open-source cloud platform.

Chapter 3

Network issues for vCDN migration

3.1 Introduction

The virtualization of resources has addressed the network architecture as a potential target. The basic tasks required in the virtualization substrate are instantiation of new network functions, migration and switching. These basic tasks are strongly dependent on the underlying network configuration and topology in a way that makes them tributary of the network conditions. It means that sometimes it will not be possible or not recommended to accomplish some virtualization tasks if the network is not presenting the minimum requirements. This brings back many déjà vu questions to networks theory but the answers to these questions require an understanding of the new context.

Most of the virtualization architectures such as NFV included in the SDN paradigm are relying on these tools to implement their technical solutions.

In this chapter, we consider the migration of network functions and we study problems that arise from this dynamic process. We present a specific use case for content distribution of multimedia services: CDN.

The rest of this Chapter is organized as follows: Section 3.2 highlights the common architectures for network resource virtualization. Section 3.3 describes the influence of the main network parameters in the virtualization process. We study

network basics such as addressing, quality of service and mobility issues. Simple improvements and technical solutions are presented and demonstrated in Section 3.4. Section 3.5 introduced our virtual CDN (vCDN) use case. Finally, this work is concluded in Section 3.6.

3.2 NFV, SDN and OpenStack

We investigate the state of the art of the recent networking technologies that can meet the virtualization domain. The main architectures used for that purpose are Network Functions Virtualization (NFV), Software Defined Networking (SDN) and OpenStack.

3.2.1 Network Functions Virtualization

NFV [79] virtualizes the network equipment (Router, DPI, Firewall, etc). We will not discuss about hardware. We will rather consider software based NFV architecture. It is a concept that decouples network functions from its underlying hardware. Then, it enables the software to run on virtualized generic environment. Therefore, several virtual appliances can share the single hardware resources.

NFV brings several benefits [82] such as reducing CAPEX and OPEX, promoting flexibility and innovation of the virtual network functions already implemented. Moreover, it has been introduced as a new networking facility that poised to amend the core structure of telecommunication infrastructure to be more cost efficient.

In a virtualized environment, the physical resources (CPU, Storage, RAM, etc.) are emulated and the guest operating system runs over the emulated hardware (vCPU, vRAM, vStorage) as if it is running on its own physical resources. By this virtualization, all the virtual machines share simultaneously the available resources.

This virtualization is based on deterministic framework which is constituted of three main components:

- Host machine: It is the physical machine (physical server) that owns the hardware's resources (CPU, RAM, Storage, Input output interfaces, etc.)
- Hypervisor: It is the controller of the virtual machines that controls their instantiation, dynamic migration, scaling in/out, etc.
- Guest machine: It is the virtual machine or the virtual appliance that is running and attached to the hypervisor.

The standardization of NFV began with ETSI and some use cases described in [170]. The main use cases found are:

- Virtualization of the Radio Network Interface: In this approach we separate the digital function of the radio named Base Band Unit (BBU) from the underlying antenna hardware and distribute the Remote Radio Unit (RRH). The virtualization can be done in a data center that communicates with the RRH distributed functions through optical back-haul networks (optical fiber) in order to accelerate the resource allocation.

  Bhaumik et al. [23] gave an important result utilizing cloud RAN in which they virtualized the BBU and distribute RRU units. They reported that Cloud RAN can decrease the network load by 22%.

- Virtualization of the Home Network: The virtualization of the home network includes the virtualization of the two main components: Residential Gateway and Set-Top Boxes that offer home services (internet access, multimedia service, etc.) to end users. This approach is based on implementing virtualized and programmable software based NFV solution such as: firewalls, DHCP servers, VPN Gateways, DPI Gateways. Then, move them to data centers in order to decrease the cost of devices and increase the QoS.

- Virtualization of Evolved Packed Core (EPC): In this use case, the virtualization targeted several functions such as: SGW, PGW, MME, HSS, and PCRF [79]. The virtual EPC (vEPC) will include all the above functions as

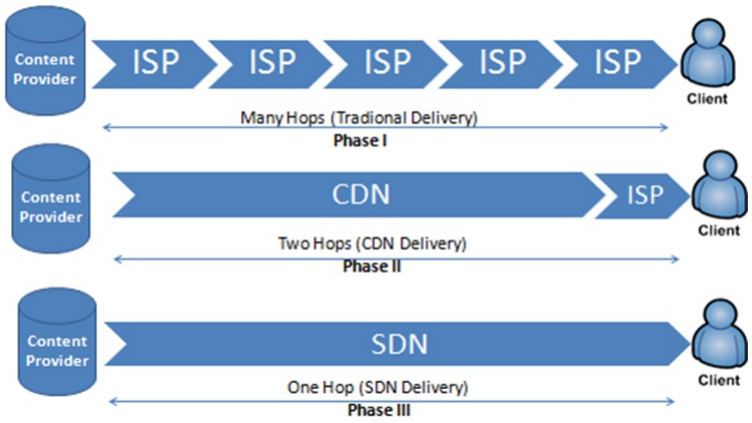


Figure 3-1 – Evolution of multimedia application delivery

software based NFV solutions moved into a cloud EPC. Using this approach we can reduce the network control traffic by 70 % as described in [79].

3.2.2 Software Defined Networking

SDN [20] is a networking concept that shares the same goals of NFV in the direction of promoting innovation; openness and flexibility. We emphasize that they are independent but complementary as explained in [96].

It is characterized by the separation of control plane and data plane and consolidates the control plane in one logical centralized controller to decrease the network overload and increase enhancement the traffic engineering by adding policy and rules to devices enabled OpenFlow protocol.

It can be considered as a tool for traffic engineering to enhance the QoS and meet network issues following the integration of virtualization functions. SDN based on OpenFlow protocol can be used to direct the forwarding layer formed by Open Flow enabled devices such as OpenFlow Switches.

As shown in Fig. 3-1, the multimedia service delivery has evolved three times:

- Phase I: The operators adopt the Best Effort (BE) in service delivery from content providers to clients according to the available bandwidth and without any regards to delay issues, users' satisfaction or perception for QoE (i.e.

no control for QoS).

- Phase II: The operators use the CDN solution to cache the multimedia very near to the clients by using CDN network acceleration. In this phase, there is a remarkable enhancement in service delivery while still some restrictions remain between customers and content providers.

- Phase III: The operators aim to redirect the required services from content providers to selected network locations giving to clients good perception in QoS or QoE measures. In this phase, there is an application layer tunneling defined by SDN (QoS/QoE perception model) [47].

Our concern in this chapter is to benefit from these complementarities in order to virtualize and migrate the multimedia network functions across hosts. Moreover, we aim to gain the advantages of scalability and cost reduction in a dynamic controlled service delivery context. We focus in this work on the mobility issues that arise when achieving the multimedia network function migration.

Mechanism of SDN in video service delivery

SDN will lead to Software Defined Data Centers (SDDC) where the roots of streaming points dynamically move. This dynamic adaptation will pass through 4 phases as follows:

- Resources Virtualization: It concerns the different resources needed during virtualization including bandwidth, system (memory, CPU, I/O).

- Roots & Links Virtualization: The virtual resources gathered and calculated in real time for the virtual node inserted among different nodes in the original tree of operators' networks.

- Network Virtualization: It is a kind of dynamic networks or nodes on demand to serve as streaming points.

- Flow Virtualization: It is a mapping of original root flows to the new elected roots through SDN tunnels between the old root and virtual root to new ones.

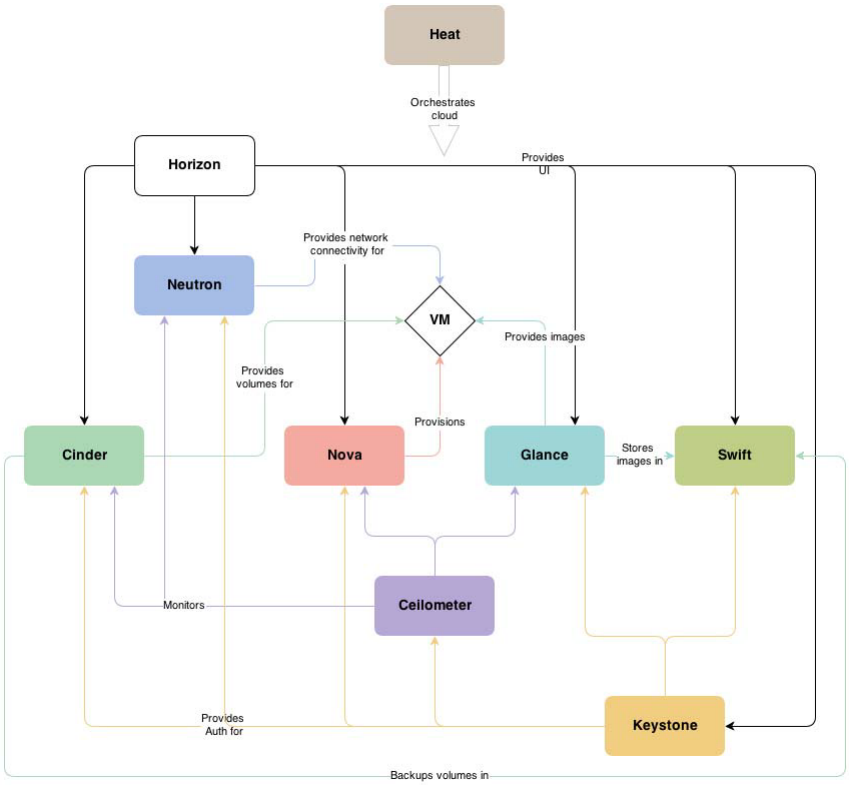


Figure 3-2 – OpenStack architecture

3.2.3 OpenStack

OpenStack [149] is a collection of services and software. It is an open-source project that replaces the Infrastructure as a Service (IaaS) layer in cloud computing domain. It can deal with the two networking technologies that appeared: NFV and SDN. Its conceptual architecture is showed in Fig. 3-2.

However the main components of OpenStack are: Nova, Glance and Swift. Glance service is the creator of image disks and Swift project is the manager of the storage in the private cloud (the equivalent of S3 service in Amazon EC2 project).

OpenStack has also an orchestrator module (Heat) that manages the virtual network functions (VNFs) in NFV based Framework. Their components are well cooperating so as to achieve dynamic instantiation of VMs in the container (VNF) through "Nova" module which is the most important service in this project. It included also Keystone which the Manager of the identity.

3.3 Network constraints in virtualization

We study mobility in the virtualization process. Several basic requirements are necessary before any virtualization procedure can take place. We think that QoS, mobility support and security are among the most important issues to be addressed:

3.3.1 QoS

This parameter is vital. Any virtualization process consumes in its transient state a huge amount of resources. Instantiation or migration of a VM requires very high speed links. Slow links do not only render the virtualization very hazardous but can simply make it impossible to accomplish. QoS is required in many situations. It is to be mentioned that virtual machine migration as of today is not optimized and very trivial approach where everything is copied from the origin to the destination. Hence, we can expect up to 90 % of the copied material is unnecessary because it is unchanged. We expect that efforts in live migration will be done to optimize what is being copied. The QoS of the link can hence be a decisive constraint on whether to migrate or not.

3.3.2 Mobility

Mobility of virtual instances is not a simple task. We can consider moving an NF or a complete instance of a server depending on the desired controller objective. So, from the operating system perspective, there will be a virtual machine, a process, a thread or a session migration as a result of this mobility. As explained before, moving a functionality is required when we want to create a new service in a different location. Off-loading, QoS improvement, interface management [18] etc. may be the reasons for this mobility. When we consider moving a complete virtual machine, it is mainly for load balancing but can also have other reasons such firmware/OS upgrade, instantiation of a new service for a new customer, etc.

3.3.3 Security

Security is an important aspect in VM migration. Many attacks could stop the live or offloading migration at any point. Moreover, a large amount of tunnels has to be setup and configured leading to very load efficiency and redundancy. So, securing this migration either in single domain or inter multiple domains is mandatory. Tools such as OpenStack do not authorize all the possible operations when different domains are involved. This could present a limitation in real deployment.

3.4 Networks issues for virtualized network function's mobility

Virtualized network functions need for an added network requirement in order to assure session moving, open session and live session migration. For basic knowledge, if a virtual machine that resides on a given network, obviously executed as a process, changes the network; it suffers from the continuity of the sessions offered to users which have connected to it. This of course is due to change of VM's IP. Therefore, we must address this issue to ensure the continuity of the session after the migration of the virtual appliance to a remote host and a remote Hypervisor.

3.4.1 Hypervisor overview

The hypervisor is the virtual machine monitor that controls virtual machine issue and specially network issues of VM and during the live migration process. It allows to move virtual machines between different domains. Among the famous hypervisors, we cite VMware [121], Xen [19], Qemu [22], and VirtualBox [44] etc.

In this chapter, we use two tools to design our optimizing network function. Qemu is first tested as an open-source hypervisor that can monitor the running virtual machine and deal with virtualization tools that can't be used in another virtualized environment such as KVM. We developed also a special tool for optimized migration based on OpenStack services [165]. using linear optimization [85]

and [88].

The aforementioned hypervisors enabled VM live migration with continuity of the running service. However, they missed several requirements for assuring live migration and open session. Those requirements are mainly concerning IP mobility when VMs migrate from source host to destination one as it is explained in [12] and [13].

Q. Li at.al [113] introduced that mobility issue can be controlled either by the VM itself, by the host machine (or any intermediate network node) or by the hypervisor. We will focus on those ways of enabling and controlling IP mobility when VMs migrate from one host to another and prove that the best way to control IP mobility is by the hypervisor.

In [99], Kalim et al. introduced another approach to migrate virtual machines between different sub-nets. In their work, they decouple IP address from TCP transporting layer and enabling transport independent flow in order to solve mobility issues and network configuration problems when migrating VMs. This approach describes the requirements and the mechanisms used. However, it misses the description of live migration scenarios that still remain unclear in their prototype.

In live migration processes any of the VM or the network nodes knows the exact time of migration and the destination host. VM can't discover that the hypervisor on which it attached to the network has been changed after the migration. Therefore, the mobility problem in live migration can be easily resolved by letting the hypervisor do this work.

3.4.2 Basic concepts for MIP enabled live migration

The basic components in our conceptual architecture are the customer or the client, the home agent and the home node, the foreign agent, the Hypervisor (Qemu in our case) and the correspondent node. We keep the basis component names as in traditional Mobile IP Protocol (MIP). Fig. 3-3 shows the four steps for

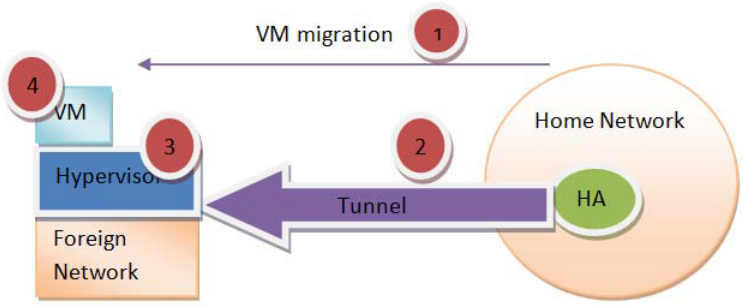


Figure 3-3 – Hypervisor monitored live migration

enabling hypervisor controlled mobile IP for live migration. Firstly, an administrator located at the Home network executes the live migration process. Secondly, a secure Tunnel based on SSH security protocol is created between the home agent and the remote hypervisor on which the migrated VM is running. Thirdly, the hypervisor software redirects the incoming traffic to the migrated VM. Fourthly, this latter keeps alive its session without any interruption.

3.4.3 Networking system design

In our basic architecture, several virtualization tools were used in order to implement network functions and run on a virtualized environment. Fig. 3-4 showed our networking scheme which relies on QEMU/KVM assisted by Libvirt daemon. Those tools manage the running VMs and their issues. The networking scheme used with virtual managers is based on Network Address Translation (NAT) for connecting virtual machines to the physical host. Each VM has emulated interfaces by which they are differentiated.

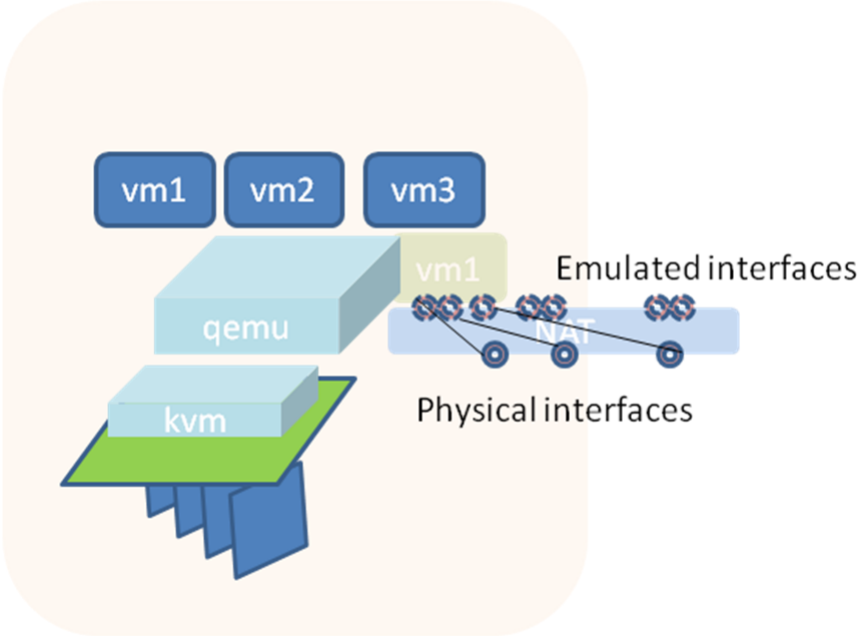


Figure 3-4 – VM networking scheme

3.4.4 Virtual machine migration process

Virtual machine migration in our scenario enables open session, session moving, content moving and virtual machine moving. We differentiate migration into two types: *i*) live VM migration with shared storage area network (SAN) and *ii*) live VM migration with context transfer. Process migration is still an open problem and session migration does not require virtual solutions.

In Fig. 3-5, we show a simple scenario of live VM migration without context transfer. In this scenario, once the virtual machines are executed, the two physical machines share the same disk image.

In Fig. 3-6, we describe a more complex scenario: live migration of a network function with context transfer. Before the migration process, the migrated VM through the hypervisor transfers memory pages and the total disk to the remote host. After the Disk transfer, the hypervisor transfers the virtual machine context (storage, plug-in, packages, libraries, etc.) to the remote host.

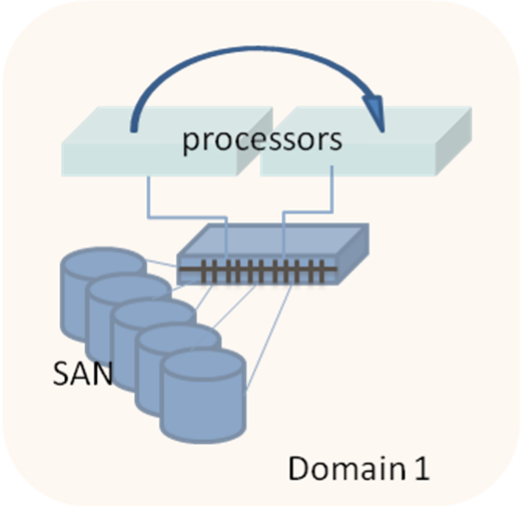


Figure 3-5 – Live migration with shared storage

3.4.5 Evaluation

Starting from our previous networking scheme based on Qemu and KVM virtualization tools, we enabled the two way of live migration either with shared disk images (single SAN or domain) or with context transfer (multiple SANs or multi-domain).

We have focused on network issues that we have faced in live migration in order to more understand the context transfer between the interacted physical and virtual machines. Some system and networking issues are evaluated in this paper. We reported that live migration of running virtual machines needed much more

Table 3.1 – Live migration requirements

	CPU State	Storage content	Network connection	Memory content
With shared SAN	Same context	Shared	ARB broadcast	Coping memory pages
Without shared SAN	Same context	Transferred, Needed for high link speed	Missed for live session	Coping memory pages

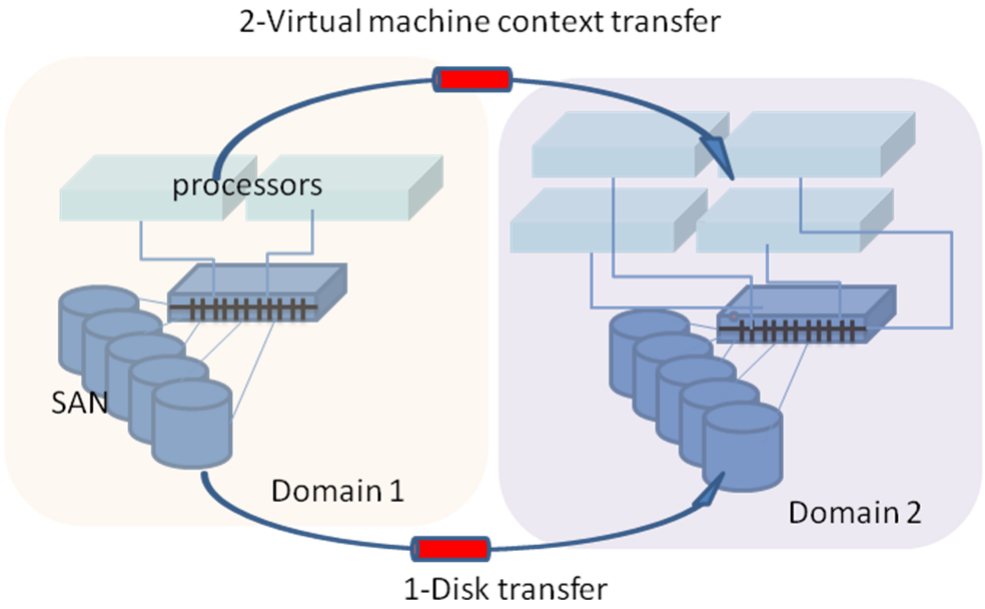


Figure 3-6 – Live VM context transfer

requirements in several migrating components such as CPU state, storage content, network connections and memory content as it showed in Tab. 3.1.

We evaluated some experiment measurements basically related to network parameters such as: link speed that must be higher than 1Gbit/s, live migration time that is not significant with shred storage (4s) and without video session interruption in the case that we moved streaming point from one host to another.

3.5 CDN use case

3.5.1 CDN

CDN is a large distributed network deployed worldwide in order to push content on the edge of the network. It was the solution of two major problems that decreased the performance of the network: congestion within the core network and overload at the origin server. It is often used to host static content so that they will be close to the end user when he made his request for online video streaming. Static contents are: images, video, CSS file, scripts files. CDN customers are mainly

the Internet Service Providers (ISPs) and Over the Top (OTT) service providers like YouTube and Netflix. All those customers want to push their content to the edge servers over IP technology. A CDN is looking to build a large farm of cache servers deployed worldwide in order to reduce the load on the OTT servers and provide the following services: Storage (popularity based), Management of cached content (i.e. cache management like replication, replacement, and placement), Distribution of content among cache servers (load balancing), Content Streaming, Fault tolerance (i.e. Redundancy), and Network performance (reduce load).

3.5.2 Virtualization of CDN

CDN federation can be proposed to improve performance. Virtual CDN, vCDN is a virtual solution for enhancing the utilization of the current CDNs.

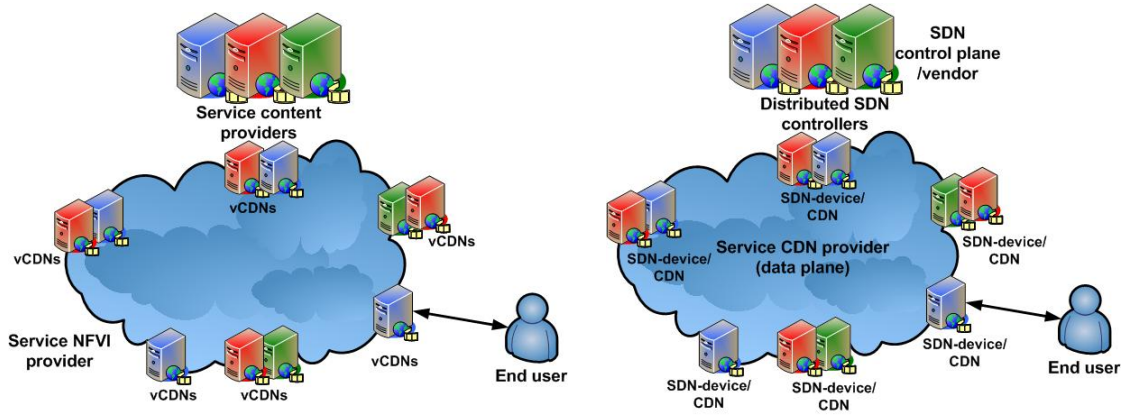
Most of the internet traffic is delivered through commercial CDN by allocating some cache servers to distribute the content to end users. We introduced that this process can be enhanced by three ways of virtualization: classical vCDN based Cloud brokers, vCDN based NFV solution, vCDN based SDN solution and vCDN based NFV and SDN. Among those virtualization ways, we have proposed three new solutions:

1. vCDN based on NFV.

2. vCDN based on SDN.

3. vCDN using the merger between NFV & SDN.

Fig. 3-7a shows our vCDN architecture based on NFV solution. Different services issued from unique or multiple service providers can hence be deployed on virtual machines running as a streaming serves on physical servers (Common-Off-The-Shelf (COTS) Sever). End-user's requests are forwarded to the closest vCDN virtual machine. Several network issues must be resolved to complete NFV and network requirements such as mobility, QoS and QoE.



(a) vCDN based NFV solution

(b) vCDN based SDN solution

Figure 3-7 – vCDN based virtualization solutions

In Fig. 3-7b, we describe our second architecture using SDN. In this approach SDN is only used as a traffic engineering or a monitor of edge servers. Through the control protocol OpenFlow (OF) enabled in OpenFlow Switches (OFS), vCDN based SDN can redirect user's request to the closest surrogate server and enhance then the overall QoS. The Network Operation System (NOS) hosts the SDN controller to communicate with the forwarding layer via OF protocol. Several control modules and applications can be added to our SDN paradigm to enhance the control plane.

In Fig. 3-8, we show our hybrid solution that merges the two former virtual architectures to virtualize and program CDN software. In fact, we keep the design of vCDN based NFV solution and we add the SDN paradigm to control the virtual edge servers (vCDN nodes) based on NFV solutions rather than dedicated hardware.

We conclude this section with a performance comparison for the vCDN solutions against classical ones in Table 3.2.

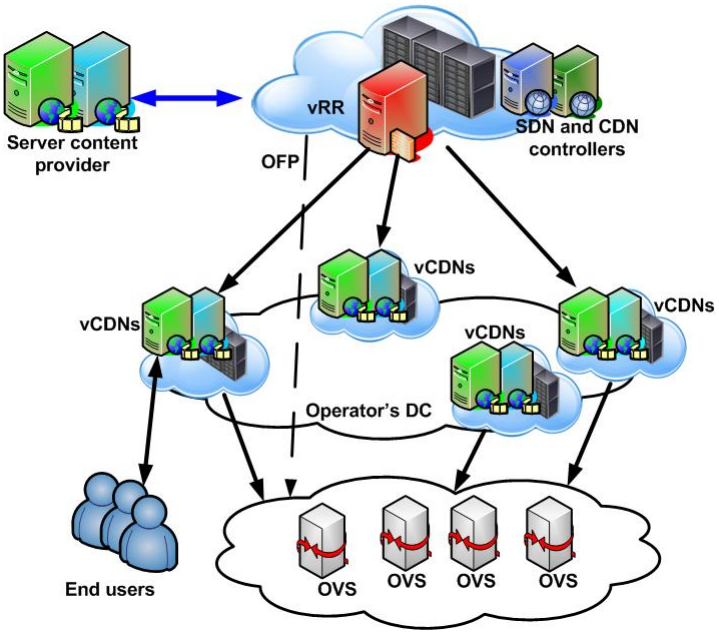


Figure 3-8 – vCDN using the merge of NFV & SDN

Table 3.2 – vCDN comparison

	vCDN based on Cloud	vCDN based on NFV	vCDN based on SDN	vCDN based on SDN&NFV
Programmablity	No	No	Yes	Yes
Migration	Yes	Yes	No	Yes
Protocols	HTTP	None	OpenFlow	OpenFlow, JSON, HTTP
QoS	Low	Low	Medium	High

3.6 Conclusion

In this chapter, we highlighted the impacts of main network parameters and criteria in the live migration process. We proposed some network virtualization concepts like SDN, NFV and OpenStack. We detailed network issues of VM migration in different scenarios. We concluded that NF context should be transferred for full VNF migration and open session. Virtualized network functions can be dynamically instantiated and migrated in a virtual environment but they need for

high requirements to achieve the full context transfer and full virtualization. We explained also a use case of vCDN for multimedia service objective.

In the next chapter, we propose vCDN protocol using the aforementioned virtualization techniques. Then, we contribute by an optimization module that recommends optimal placement points where to instantiate vCDN nodes.

Chapter 4

OPAC: Optimal Placement Algorithm for virtual CDN

4.1 Introduction

Content Delivery Network (CDN) targets the deployment of multimedia content (images, files and videos) over multiple data centers through Internet Service Providers (ISPs). The main goal of a CDN is to serve content to the end users with better availability and higher performance [98] and [132]. It is also proposed to ISPs as an offloading solution for intra and inter data centers updating [64].

Fig. 4-1 depicts the main components of the typical CDN network. It includes: *i*) a content replication plane: it is the system responsible for content caching, streaming and distribution. *ii*) a request routing plane: it is the system responsible for redirecting end-user requests to the optimal surrogate server, and *iii*) a CDN management plane: it the centralized system responsible for cache management processes such as content eviction, placement, etc.. and it does the accounting and billing tasks. Moreover, this network can perform many file management actions like hosting/caching/fetching/swapping in order to satisfy the client's Quality of Experience (QoE) and enhance the overall Quality of Service (QoS) of the network [78].

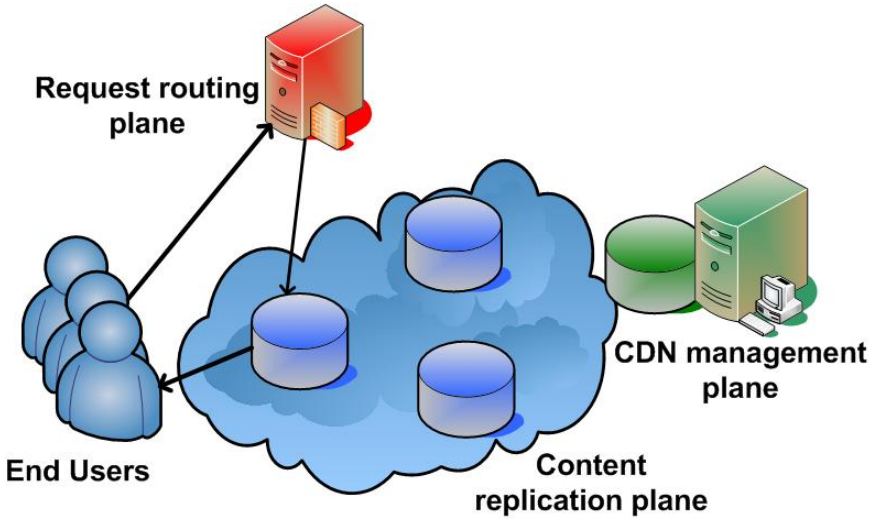


Figure 4-1 – Main components of Content Delivery Networks

In video streaming systems either live-streaming or on-demand services, the CDN placement is a key point in delivery optimization. In [50], the video adaptation was considered in terms of users' perception (user experience). In [49], authors focused on the contextual QoE metrics and their effects on data retrieval or caching costs. The overall performance especially in video services cannot be only based on user perception models. In fact, major deployed data centers over the Internet such as Akamai [4], Amazon [6] or Google [105] have to consider other important parameters. Besides financial issues that are out of our scope, we claim that system and core network parameters are key issues in making content movements over the networks. To the best of our knowledge and despite the existence of placement techniques deployed by the above mentioned providers, the literature lacks optimization algorithms that take into consideration actual system, network, and quality parameters. In fact, each technique is based on a single criterion optimization.

Architectures for virtualizing network functions such as Network Functions Virtualization (NFV), Software Defined Networking (SDN) and OpenStack [1] can add flexibility to the design of network applications as investigated in [41], [95],

[1] It is an opensource cloud computing platform.

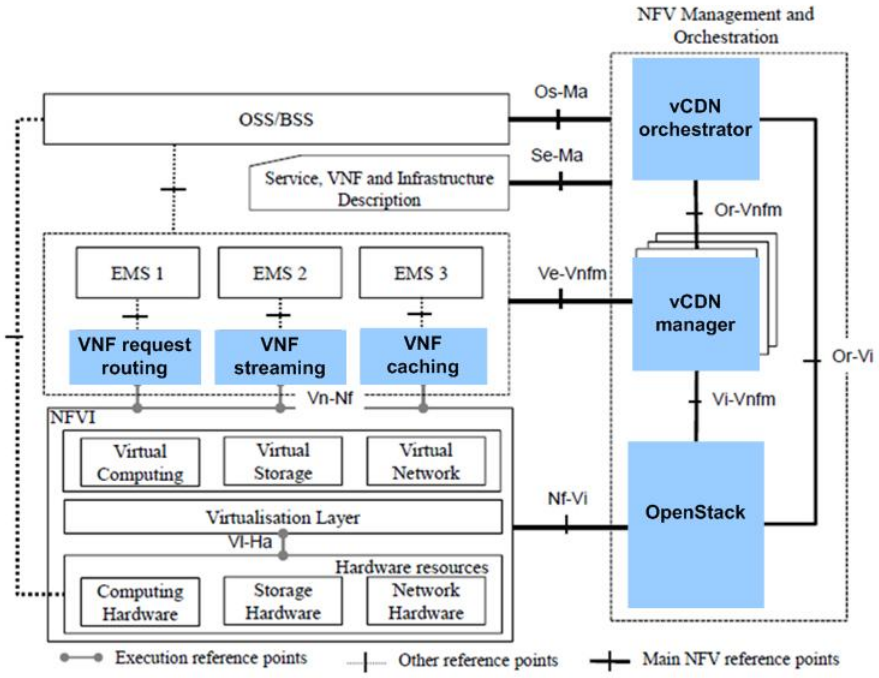


Figure 4-2 – vCDN architecture according to ETSI-NFV [171]

[52], [178], [118], [119], [60], and [68]. They were recently adapted to the CDN context for video content placement. The goal was to decrease the load on the origin content providers (like YouTube, Netflix[2]) and to let the operators/ISPs host the content on their network to decrease as much as possible the load on the external links. Yet, an additional optimization mechanism should be proposed to further improve the overall caching performance combining optimization, placement, and virtualization within the operator/ISP network.

The overall virtualization challenges for CDN transition to vCDN are addressed in the research project DVD2C [129]. In a previous chapter, we have investigated the main network issues in virtual machine migration and especially for vCDN use case and in an SDN/NFV context. Fig. 4-2 depicts the proposed vCDN architecture based upon the ETSI-NFV reference architecture [171]. Note here that different vCDN implementations could exist. Indeed, as depicted in Fig.

[2]It proposes an open CDN using NGINX web proxy server and other open source software: https://openconnect.netflix.com/software/

2-1, a CDN network is composed of different systems such as caching, distribution, request routing, and management. Therefore, by making the transition from classical CDN to the NFV/SDN-based vCDN context, all theses system components can be virtualized and deployed as VNF instances, but given the growth of video delivery, virtualization has to target mainly the caching and streaming components (i.e., VNF caching, VNF streaming, etc.). Further, a VNF request routing system (vRR) is added to the vCDN proposed architecture in order to orchestrate the VNF caching/streaming nodes.

Currently, despite the importance of optimization tasks in the novel virtualization context of SDN and NFV, such functions are missing in the global network architecture. We therefore, try in this chapter to contribute with OPAC (protocol and optimization model) as a potential optimization technique for vCDN placement and explain how it is integrated in the network operator. Although video on demand (VoD), assisted by cascaded caching, enhances the QoE and decreases the load, live video on the other hand can still benefit from our optimization algorithm.

In this chapter, we introduce the concept of virtual CDN as a Service (vCDNaaS) using NFV and SDN tools. Through our proposal, we focus on the framework architecture, design and placement of vCDN use case based on SDN controller, e.g, OpenDaylight [128]. Then, an optimization placement algorithm for vCDN including content caching and request redirecting is introduced with operating system, network, and quality of experience constraints.

The rest of this chapter is organized as follows: Section 4.2 investigates the related work. Section 4.3 proposes the vCDN concept in terms of system requirements and design methodology. Section 4.4 details the steps of the proposed virtual caching protocol. The optimization model is introduced in Section 4.5. The final optimization evaluation is presented in Section 4.6. The update performance if the cache prediction is slightly imprecise is evaluated through a comparison in Section 4.7 and then the work is concluded in the last section 4.8.

4.2 Related work

This section highlights the relevant NFV/SDN optimization algorithms in the virtualization context. Thus, facing the difficult combinatorial optimization problem in the sense of the theory of algorithmic complexity, there are two main solutions for the Virtual Machine (VM)/Virtual Network Function (VNF) Placement Problem (PP): 1) Exact approaches (optimal algorithms) and 2) Heuristic approaches (e.g., best-fit decreasing, first-fit decreasing, genetic, and meta-heuristic algorithms). We try to give an overview of the recent works.

Niels et al. [28] provide a framework for multimedia delivery CDN-based on NFV. They gave a structure of Point of Presence (PoP) deployment within the network. They used the three-layer structure (core network, aggregation layer, access layer) of the network to seek the optimal locations of cache nodes. Their work missed an optimization algorithm to measure the QoS/QoE. Further, they don't take into consideration the cooperation between CDN edge servers to serve user request. Also, the optimization model lacked many constraints such as the maximum server's storage capacity and the migration of content from one server to the optimal one.

Michele et al. [28] after presenting the state of the art of NFV, CDN, and stochastic optimization techniques, use a mixed architecture where real and virtual CDN nodes coexist. Although they present an interesting hybrid solution, their architecture assumes that the distribution of the future traffic demand is known which is not a realistic assumption.

Hendrick et al. [122] gave a model for resource allocation of VNFs within a virtualized environment (NFV Infrastructure). Cloud scaling technologies for VNF instantiation are criticized in order to prepare for NFV use cases inside the Network Service Provider (NSP). They affirm that the NFV paradigm was not designed to be used only in Data Centers (DCs). We agree with their affirmation since in data centers we don't have link or storage capacity shortage. Indeed, cloud administrators provision their deployed clouds with enough link speed, storage, and

computation resources. However, it is not the case in NSP since we face dynamic constraints on network, storage, bandwidth and latency. Authors propose another hybrid solution inside the operator network that mixes between the legacy networks based dedicated hardware for each network function and the virtualized software/instance running on basic and standard hardware in NFVI environment.

The NFV resource allocation proposed by the authors gives different approaches for task and service deployment in hybrid NFV networks such as VM deployment on physical servers, and service deployment either on physical server or on VM. That model named (VNF-P) for virtual network function placement aims to find an optimal way for deploying a service chain characterized by a chain of VNFs taking into account some optimization features to minimize the overall cost of the deployment. VNF-P is a good abstraction of the deployment of NFV in an operator's network. But still, it is too general and does not address CDN specificity. Further, among the shortcomings of the proposed placement model for VNF, authors didn't consider real virtualization constraints such as storage, virtual CPU, virtual streaming costs. Moreover we think that they did not use an adequate distribution of user's requests and that they did not consider any QoE parameter or constraints to deal with user's demands variation.

Bernardetta et al. [3] highlight three aspects in the general NFV placement problem which are: 1) network modeling, 2) generic VNF placement algorithm, and 3) VNF routing problems. They raise an important network flow problem related to (de)compression processes applied on through-traffic passing by VNF instances. This problem of VNF chaining requires high resource allocation. Further, they propose a multi-level objective function for VNF placement optimization. In fact, they present the problem of minimizing the allocated computing resources in a first stage and in a second stage, they perform a second optimization: the minimization of the maximum link utilization. The limitations of their approach are:

They used a prioritization method to solve the multi-objective optimization

problem which leads to sub-optimal results especially when the cost functions are orthogonal as in their case. Moreover, authors didn't take into consideration the virtual RAM which is most important virtualization parameter especially in service migration among virtualized launched instances.

Authors have not properly configured the latency constraint in the core layer (it should be lower than the remaining network layers). Furthermore, the problem was resolved through a math-heuristic algorithm assisted by a prioritization method which has several limitations such as non-optimal cost analysis.

Mathieu et al. [27] propose a placement problem optimization for the Deep Packet Inspection (DPI) networks through designing a vDPI (virtualized DPI) solution. Moreover, they consider the strong relationship between NFV and ISPs in order to host dedicated cloud systems in ISP's Point of Presence (PoP). Their work was concentrated on where to instantiate vDPI in the network to reduce the cost and overcome challenges in the vDPI deployment process. They then gave their general placement model for vDPI cost based placement constrained by OPEX, CAPEX, and cybersecurity costs (since DPI is a security network function). Authors aimed to propose a model capable of minimizing network load and total number of deployed vDPI engines. This contribution is useful for our CDN placement problem. There are however major differences between the deployment of DPIs and CDNs. Moreover, authors assume the knowledge of the traffic flow distribution. This static parameter has limited their solution.

Literature lacks an optimization algorithm that introduces at the same time system, network, and user quality parameters. Moreover, none of the above work introduced a placement and/or migration protocol for virtual content delivery network functions. Hence, we contribute by OPAC protocol and optimization model.

4.3 OPAC: design concepts

We present an efficient solution for optimizing video streaming delivery according to the proposed system design.

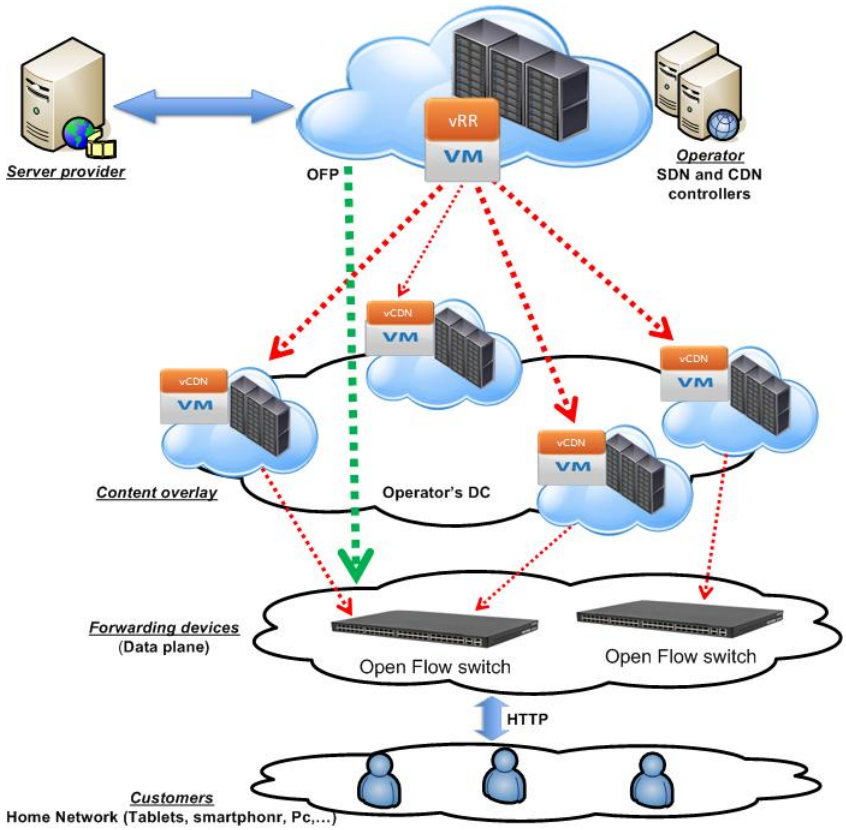


Figure 4-3 – OPAC-based vCDN deployment architecture

We build our system based on the SDN paradigm assisted with traffic engineering rules using Network Function Virtualization Infrastructure (NFVI presented in ETSI standard [170]). We apply virtualization on content delivery networks. NFV concept is used here to virtualize the CDN components (cache/stream nodes and a request router node) while SDN paradigm is used for controlling the behavior of these virtualized instances through open switch-based configurations. Beside vCDN traffic engineering, vCDN programmability, and flexibility, SDN acts as the request routing system of OPAC protocol that communicates with the CDN controller and the virtual instances created with respect to the hypervisor-based virtualization of NFV. We have implemented OPAC algorithm and we have used both paradigms SDN (with OpenFlow protocol) and NFV (via the OpenStack platform enriched by our vCDN services)[3]. Our proposed system model is shown in

[3]A first version of the software is available at: https://github.com/TelecomSudparis-RST/vIOS.

Fig. 4-3 and consists of two new entities (with respect to the SDN basic architecture [105]):

1. vCDN node: It is the software that virtualizes the CDN caching and streaming services on a standard physical server. It runs on a simple VM. vCDN nodes are distributed inside the operator's network. They act as a broker between content provider and end users.

2. Virtual Request Router (vRR): It constitutes the controlling element for a CDN network. We propose an overlay network of virtual nodes (vCDNs) instantiated by the virtual request router which acts as the vCDN controller. The vRR is hence the virtual function in the vCDN controller that redirects end user's requests to the nearest (suitable) vCDN node. The request routing process is based on the OpenFlow Protocol [63] (OFP). It uses a Time To Live (TTL) interval as a cache update strategy.

As shown in Fig. 4-3, OPAC is used for managing and orchestrating vCDN as a service. Further, the vRR proposed here is able to redirect user's requests to the *optimal* vCDN streaming node. Hence, once user requests are directed to the CDN server (VNF), the latter redirects the request to the optimal vCDN relative to the user location and other algorithm parameters.

The event sequence is that a Content Provider (CP) asks the network operator (virtualized CDN provider) to perform a cache process for its videos (referenced by URLs). URLs are classified as *cacheable* or not in the network (see Fig. 4-4). Moreover, each content should be cached (the most popular and requested by end user) into an optimal vCDN node (content placement problem). Therefore, we use a cache function to track the migration of videos as follows:

$$Cache(v,t) = vCDN_{ID} \tag{4.1}$$

Where $vCDN_{ID}$ is the vCDN identifier, v is the video identifier (it is often a key file) and t is the time when we place/allocate the video identified by v.

Algorithm 1 Greedy algorithm for vCDN caching

1: **Input:** $video, user, vCDN$,
2: user-request (user u, video v, vCDN $vCDN_{ID}$);
3: **if** $video\ isCached()$ **then**
4: OPAC Protocol ( );
5: **end if**

We can hence program and deploy our caching algorithm proposing better optimization to enhance the QoE/QoS of end users, decrease the load on the origin content provider (e.g., Youtube, Netflix, etc...), and let the operator host (allocate) the content on its content network ($vCDN_1$, $vCDN_2$,... $vCDN_n$).

OPAC uses a Distributed Hash Tables (DHT) to index the content URLs over the vCDNs. vCDNs communicate periodically with vRR to maintain their connectivity and location.

An additional fast redirection happens as follows: if the video is cacheable, then it should be declared so. And then, the request is redirected based on OpenFlow rules and inserted in OpenFlow switches when the request is intercepted.

We use a greedy algorithm as follows: a user requests a video content. If it is cached. Then, the request is redirected to an optimal vCDN node based on OPAC protocol as detailed in the next section.

Algorithm 1 summarizes the pseudo code of vCDN cache algorithm. Hereafter, we describe these main stages.

4.4 OPAC protocol

The main steps of the proposed protocol are detailed hereafter:

1. The end user i requests a video j , $video_{i,j}$.

2. Network Service Provider (NSP) assisted with SDN network, management layer and NFV deployment (see Fig. 4-4) redirects customer's request to the optimal vCDN cache node (the optimal placement is derived with the help of an exact optimization algorithm, formulated and detailed later).

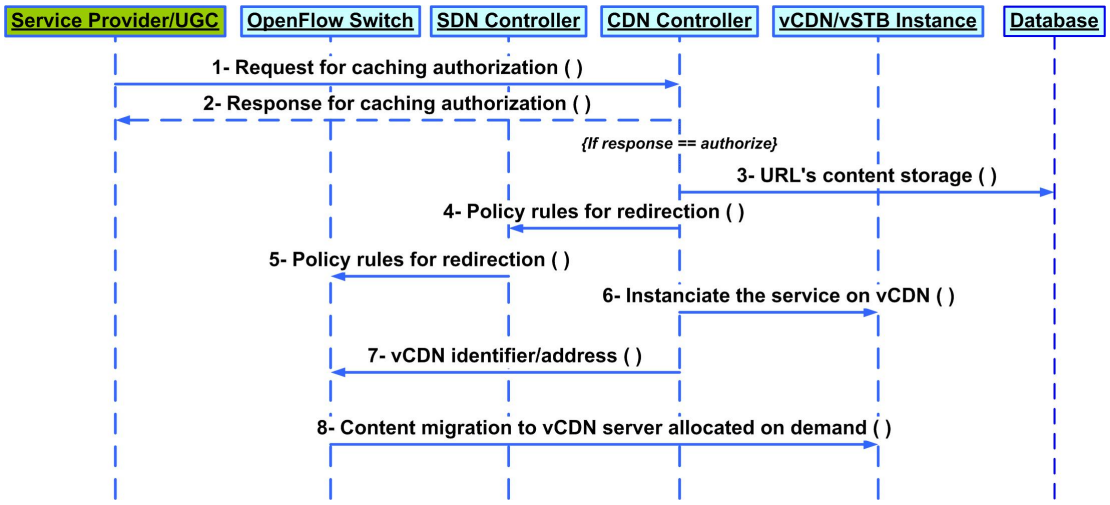


Figure 4-4 – vCDN cache as a service

3. If $video_{i,j}$ is located in the optimal vCDN, then deliver($video_{i,j}$) (see Fig. 4-5).

4. Else VoD origin server delivers the requested video to the user, and optimal vCDN node caches the video (see Fig. 4-6).

5. Notification of the caching controller about the mapping between the new vCDN address and video name.

For the sake of simplicity, we assume that the video service must be cached through the vCDN (we name it a vCDN cache as a service). Once the content is cached, requests from clients are dynamically redirected to the migrated vCDN (the deployment location has been proposed after executing OPAC optimization algorithm). Hence, We can enumerate the steps of the OPAC protocol as the following:

1. vCDN cache as a service (Fig. 4-4):

   - Authorization: Either the Service Provider (SP) or the User Generated Content (UGC) must request authorization (step 1) in order to cache a popular video item (e.g., ”/video.mp4”).
   - SP/UGC takes the authorization (step 2) from the Caching Controller (CC) which is part of the network operator.

- CC initiates policy rules (PRs) for redirection (step 3) by communicating with the database server (DB), then it interacts with SDN controller (step 4) to notify it by the PRs. The latter injects those PRs in OpenFlow switch device (step 5). Then, CC instantiates (if needed regarding to the available resource) the video service on vCDN. This virtual node can be a set-top box (STB) device that supports the virtualization technique (step 6).
- CC sends the vCDN address (@vCDN) to the requester (CP or UGC) (step 7).
- SP/UGC migrates/pushes the desired video content to the vCDN allocated on demand (step 8).

2. Video delivery (Fig. 4-5 and Fig. 4-6):

- Customer requests for a desired video (step 1)
- OpenFlow physical/virtual switch redirects under the previous PRs customer's request to the optimal vCDN streaming server (step 2)
- vCDN streaming server checks for the video (Fig. 4-5, step 3).
- If desired video is located in the local cache of vCDN (hit cache scenario) then it executes service video delivery process (Fig. 4-5, step 4).
- Else (miss cache scenario) it retrieves the video from the origin VoD server (Fig. 4-6, steps 3 and 4).
- vCDN streaming server delivers the video to the end user (Fig. 4-6, step 5).
- vCDN streaming server notifies the CC in order to register the mapping between vCDN node identifier and video item (Fig. 4-6, step 6).
- CC updates/refreshes the database (Fig. 4-6, step 7).

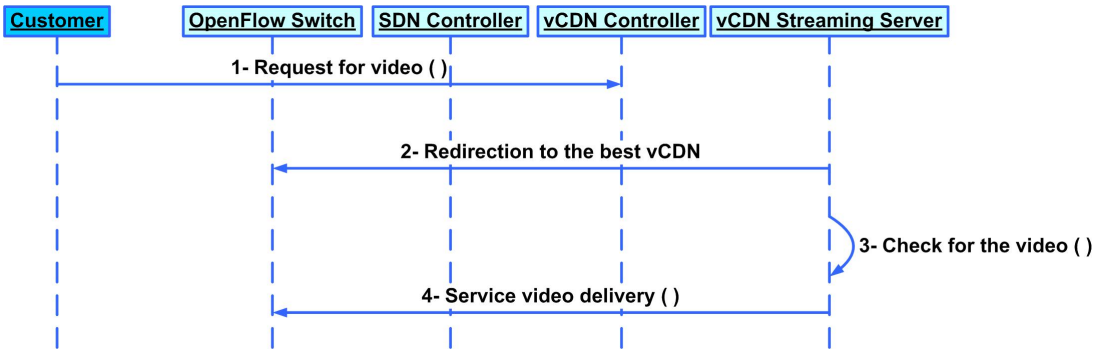


Figure 4-5 – vCDN hit cache scenario

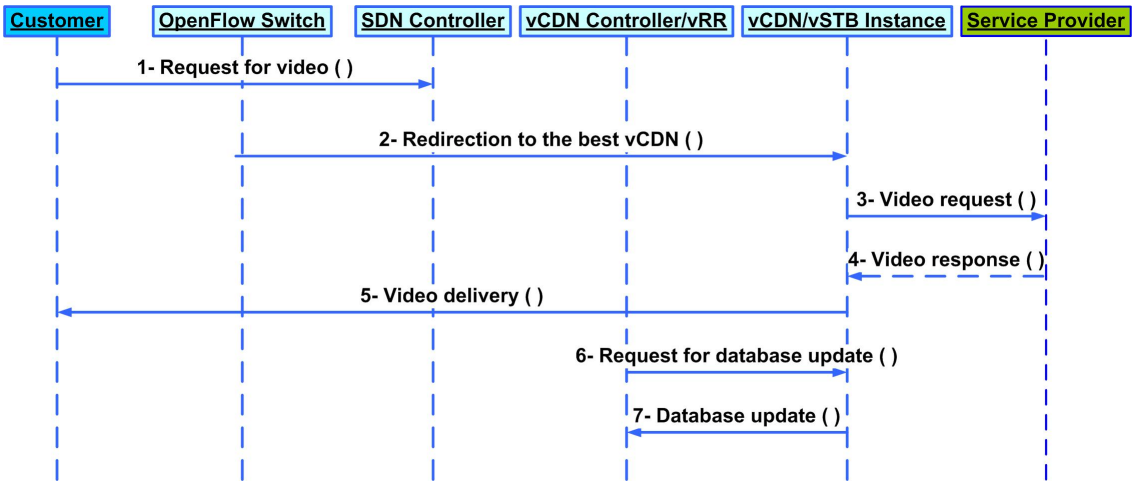


Figure 4-6 – vCDN miss cache scenario

4.5 OPAC optimization model

We propose an exact optimization algorithm that takes as an input the topology of the underlying network. It aims then to optimally place, and migrate the virtual content delivery functions (vCDN nodes) upon the virtualized infrastructure. Those virtual nodes move from one location to the optimal point in order to increase user satisfaction and decrease server and network loads. The optimization algorithm for placement is modeled, implemented, and evaluated in the next subsections.

4.5.1 Problem statement, constraints and main objectives

The problem considers delivering videos on demand or services, e.g., television programs, the movies, etc to a large group of users located in distant zones and served by an operator network. This operator may or may not be the service provider. Moreover, the problem supports users with different resolution requirements.

The statement is as follows: determine where to locate the VoD streaming head-end, and how to migrate the virtual delivery node from one location the optimal one.

We consider different types of constraints in our virtualization process: system type constraints such as RAM, CPU, and storage, and network type constraints such as flow balance (conservation), QoE, link capacity, and server streaming constraints: maximum number of simultaneous connections per server). Recall that none of the previously described contributions considered the QoE parameter. The latter aims to link context information such as the operator delivery network to video streaming. OPAC optimization will maximize the Mean Opinion Score (MOS) of end users having different devices and connected over different technologies (ADSL, WiFi, and Ethernet). Video delivery adaptation mechanisms using utility functions [47] and MDASH (MOS Dynamic Adaptive Streaming over HTTP) can hence be supported by our work.

The objective is to minimize the cost of resources (i.e. bandwidth and migration).

The main goal of our study is to propose an algorithm and the underlying mechanisms to optimize video delivery networks. These mechanisms will guarantee the desired user's satisfaction level and on demand service optimization as:

1. Allocation mechanism and virtual delivery node migration: we introduce an optimization algorithm for vCDN allocation based on the maximum number of connections related to each file and its viewing rate (according to informa-

tion calculated by counters such as DailyMotion/YouTube [4]). Our algorithm enables simple file movement between different domain vCDNs considering minimum cost of content access.

2. Service optimization mechanism (service on demand): The second step is to analyze the virtualized delivery node and migrate them to the optimal PoP headend in order to achieve a better quality of experience (QoE) and better savings for network operator resources.

This implies three levels of optimization:

1. Multi-criteria optimization: this first level aims to satisfy as maximum as possible the involved actors in the delivery process: the content provider that wants to push the maximum content to the CDN provider with minimum cost, the network operator which wants to decrease its server's load and save its resource (storage, bandwidth, ...) as possible, and the client who requests high quality of experience and better satisfaction with minimum price.

2. Network optimization: we optimize the load on the main network links and increase the operator's gain (delay reduction in content streaming).

3. QoE/QoS optimization: we focus on user's QoE by adapting the video streaming resolution to its context. We assume that vCDN users may require different video resolutions (Standard Definition (SD), High Definition (HD) and Ultra High Definition (UHD)). The quality parameter d_v^f corresponds to the resolution requirement per each cluster of end users is defined in Table 4.1. The model can be easily extended to accommodate future ultra higher definition TV services like 8K.

We will detail the optimization problem constraints for vCDN migration and the objective function of our work in the next section.

[4]YouTube MyTop100Videos: https://www.youtube.com/user/MyTop10Videos

Table 4.1 – OPAC mathematical notation

Parameters	***Definition***
V	The set of client group nodes
S	The set of server nodes
S'	The set of optimal server nodes
$D^{s'}$	Maximum throughput of the streaming server $s' \in S'$
F	The set of vCDN nodes
f_{size}	vCDN resource's size $\left(vRAM, vCPU, and\ vDISK\right)\left(f \in F\right)$
$C^{s'}$	Maximum storage capacity of the server $s' \in S'$
$C_{i,j}$	Link capacity between two nodes i and j (from i to j)
d_v^f	Throughput used for streaming the vCDN functionality $f \in F$ to the client group $v \in V$. It represents the user's demand requirements $(SDTV, HDTV, UHDTV)$
$c_{s,s',f}$	The migration cost of the functionality f from s to s'
Decision variables	***Definition***
$x_{s,s',f}$	Placement and migration binary variable which indicates that f should move from from s to s'
$y_{v,f}^{s'}$	Binary variable which indicates the video hit from node v of f in server s
$z_{i,j}^{v,f}$	Binary variable indicating whether the link (i, j) is used to stream f to v

4.5.2 Mathematical formulation

In this section we specify the parameters and the constraints that are defined and proposed in formulating the optimization/analytic model. This formulation determines the migration of vCDN nodes to the optimal location.

- OPAC Parameters: OPAC deals with system, network, and QoS/QoE parameters as quoted in Table 4.1. We represent each by some parameters although it would be better to have them all detailed. For complexity issues, vCDN resource's size (f_{size}) (beside the other system/network parameters) is a normalized function of some relevant values that describes the size of the in-

stance (system). Moreover, we introduce the QoS/QoE metric (d_v^f) in our optimization problem. This parameter represents the consumer demands matrix. Periodically, the network operator feeds OPAC database with the existing demands: requests for vCDN services. A demand comes from a client group ($v \in V$) and targets a vCDN ($f \in F$). This is to link end-users need (i.e., requesting vCDNs) with a specific throughput that is mentioned in their requests as a matching parameter between the requested quality (SD, HD, or UHD in terms of throughput) from the client group (v) and the streaming throughput from the vCDN serve (f). The algorithm tries to satisfy all the user quality of streaming requests. Including all parameters would be too complex to solve in a reasonable time.

- Decision Variables: We quote in Table 4.1 the main system parameters and decision variables.

  1. The binary variable x indicates the placement of the streaming headend, and its migration from one server s to another (best/optimal) location s'. It is defined as:

$$x_{s,s',f} = \begin{cases} 1 & \text{if } f \text{ migrates to } s' \\ 0 & \text{Otherwise} \end{cases} \tag{4.2}$$

  2. The binary variable y indicates client request need for a vCDN service located on the optimal server. It is defined as :

$$y_{v,f}^{s'} = \begin{cases} 1 & \text{if } v \text{ needs } f \text{ and } s' \text{ caches } f \\ 0 & \text{Otherwise} \end{cases} \tag{4.3}$$

  3. The binary variable $z_{i,j}^{v,f}$ indicates whether a link (i, j) is used (from i to j) to stream from a server s' replicating f (the one for which $y_{v,f}^{s'} = 1$) to client v.

- Constraints:

  1. The binary variable y should be less than or equal to x. In fact, y equals 1 if and only if v needs f from the optimal server s' which has the functionality. This means that x should be equal to 1.

$$\forall s' \in S', v \in V, s \in S : y_{v,f}^{s'} \leq x_{s,s',f} \tag{4.4}$$

  2. Only one optimal server should serve the attached node that requests the functionality f:

$$\forall v \in V \mid d_v^f \neq 0 : \sum_{s' \in S'} y_{v,f}^{s'} = 1 \tag{4.5}$$

  3. The cost of streaming f by the optimal server should be less or equal to the maximum server capacity:

$$\forall s' \in S' : \sum_{v \in V} \sum_{f \in F} y_{v,f}^{s'} \times d_v^f \leq D_{s'} \tag{4.6}$$

  4. The storage of the optimal server should not exceed its maximum size:

$$\forall s' \in S' : \sum_{s \in S} \sum_{f \in F} x_{s,s',f} \times f_{size} \leq C_{max}^{s'} \tag{4.7}$$

  5. Flow balance or conservation constraint (which is the laws of Kirchhoff) between the optimal server s' and the client node v should be as the following:

$$\sum_j z_{i,j}^{v,f} - \sum_j z_{j,i}^{v,f} = \begin{cases} 0 & \text{if } i \neq v, i \neq s' \\ y_{v,f}^{s'} & \text{if } i = s' \\ -1 & \text{if } i = v \end{cases} \tag{4.8}$$

  This is the network flow constraint. The sum of incoming flows must be equal to the outgoing ones.

6. Link capacity between source i and sink j should not exceed its maximum:

$$\forall i, j \in V \cup S' : \sum_{v \in V} \sum_{f \in F} z_{i,j}^{v,f} \times d_v^f \leq C_{i,j} \tag{4.9}$$

4.5.3 Mono objective resolution

The objective function: the migration cost (transit) as formulated in equation (10).

- Objective function:

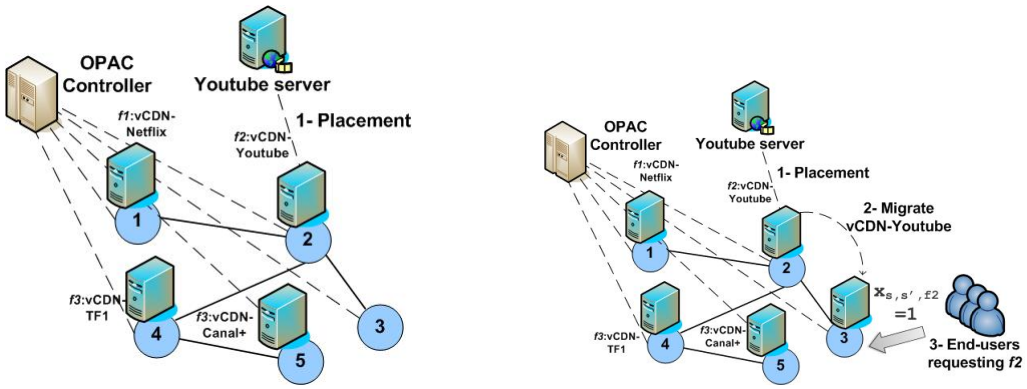
$$\min \sum_{s' \in S'} \sum_{s \in S, f \in F} x_{s,s',f} \times c_{s,s',f} \tag{4.10}$$

  where $c_{s,s',f}$ is the migration cost:

$$c_{s,s',f} = \alpha \times f_{size} \tag{4.11}$$

  Where α is a parameter depending on the position of s' and the position of s (the server initially containing f before migration). α depends also on the operator policy (e.g., the internal policy of the operator like configuration policy).

- Optimization technique: The OPAC-based optimization domain targets a set of large size vCDN nodes to be placed in network operator snapshot (with a relatively low number of NFV servers client group nodes). The goal of OPAC is to intelligently (and dynamically) place and migrate the set of vCDN nodes in response to client group demands in order to increase the in-network caching performance (and hence QoE) while minimizing the total migration cost function as formulated in Eq. 4.10. The migration problem of vCDN to the optimal placements is an Integer Linear Problem (ILP) that can be solved within a few seconds. It (the migration) is modeled through branch and bound optimization technique (method) that minimizes the network virtualization cost (total migration cost) among a set of candidate solutions that



(a) Before migration (b) After migration

Figure 4-7 – Example based OPAC-analytical procedure for vCDN placement

maximize the end user quality of experience. Starting from the previous system, network and content quality constraints, OPAC searches the optimal data center locations, where the binary migration vector $x_{s,s',f}$(defined above) equals to 1.

- Analytical procedure (OPAC-based decision example): For the sake of clarity, we propose an example based on OPAC procedure for vCDN placement/migration context as shown in Fig. 4-7. The algorithm is as follows: 1) As an input, OPAC controller gathers SDN/NFV architectural information such as the initial placement of vCDNs ($f_1 \in s_1$, $f_2 \in s_2$, etc.) and all necessary dynamic parameters (system capacity, network capacity, throughput, location, etc.) as shown in Fig. 4-7a representing the network state before migration, 3) as a result, end-users requesting vCDN YouTube®, to watch a popular video in a live event for example, will be redirected to the videos hosted in the new optimal calculated location as depicted in Fig.4-7b representing the network state after migration. Recall that the network operator hosting the vCDN of YouTube content provider migrates the vCDN cache/stream node to an optimal PoP where resources are available and content quality streamed are satisfied.

Note that in a micro cache deployment (large scale) scenario, OPAC is still accept-

able for vCDN placement/migration application. This is because the algorithm run time is in term of seconds. Therefore there is no need to develop a heuristic algorithm. OPAC algorithm is proposed to be executed in a virtual CDN controller/manager entity (vRR). It has to control/manage/orchestrate the VNFs running virtual CDN nodes. This placement is executed after three main triggers:

- System constraints (through f_{size} and $C^{s'}$ parameters). and network constraint (through $C_{i,j}$, $D^{s'}$, and d_v^f parameters).
- Service Level Agreements (SLA): It is QoS/QoE constraint through a contract between the content provider (the customer of the vCDN solution) and the network operator (the server or the provider of the NFV servers) (d_v^f resolution requirement per each cluster of end users). It (SLA) is considered through a valid range of streaming qualities: SD, HD, and UHD of each couple (client group–vCDN). The optimization process uses this requirement and the result satisfies the SLA.
- Storage/bandwidth prediction: OPAC makes the migration and assumes that the network and system parameters are available (through OpenStack dashboard). We mean by prediction that those parameters may change and cannot be known in advance. Recall that data centers are highly dynamic and shared, so live metrics and parameters are used from the virtual infrastructure manager (e.g., OpenStack data-centers) and update at the same time the OPAC database.

In the current implementation version of OPAC, triggers are not used to execute the optimization. However, as we use a database for our real-time parameters, all the dynamic values can be annotated with a triggering threshold that re-launches the process. The user demands, system parameters and network load are the potential triggers. Moreover, in this version OPACv1, we don't consider individual user demands, only groups (represented by the d_v^f matrix). Second we envision the execution of the algorithm in a predictive scenario. Till now, these demands

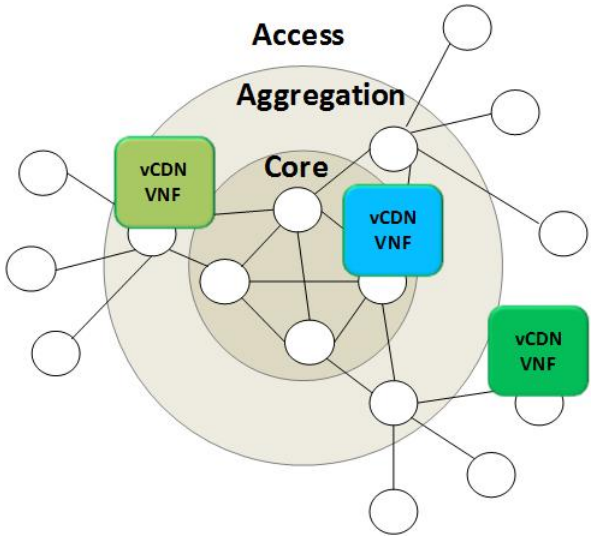


Figure 4-8 – Network topology used for OPAC evaluation

are a forecast of client groups ($v \in V$). The consumer demands are fictional and have been initialized in the OPACv1 database and have the corresponding vCDN request list, PoP request list, etc. New demands are treated in the next optimization round. The optimization operation should be executed in accordance with the data aggregation frequency of the demands and the system/network load based on OpenStack.

4.6 OPAC: optimization evaluation

We evaluate our work in a network that obeys the 3-tier layers architecture (access, aggregation and core layer)[5]. Fig. 4-8 shows the network topology used for the evaluation. Further, for the interest of assessing the efficiency of OPAC algorithm, we used CPLEX [84] as an optimization tool.

We note that servers may be deployed in any network layer while client nodes are deployed only in the access network. We mention also that vCDN cache nodes are deployed on top of the physical servers that among NFVI and migrate from the source s to the optimal server s'.

[5]https://tools.ietf.org/pdf/draft-bagnulo-nfvrg-topology-00.pdf.

Further, we define the NFV cost and the placement time as follows:

- The NFV cost is defined as follows:

$$NFV\ cost = \sum_{s \in S, s' \in S'} \sum_{f \in F} x_{s,s',f} \times c_{s,s',f} \tag{4.12}$$

- The migration time is defined as follows:

$$Placement\ time = \sum_{s \in S, s' \in S'} \sum_{f \in F} x_{s,s',f} \times f_{size} \times \max_{(i,j) \in P_{s,s'}} \frac{1}{C_{i,j}} \tag{4.13}$$

Where $P_{s,s'}$ is a given path from s to s'. In fact we are here assuming that placement is done in sequential way. If placement is performed in parallel, then the placement time would be given by $\max_{s' \in S', s \in S, f \in F} \left(x_{s,s',f} \times f_{size} \times \max_{(i,j) \in P_{s,s'}} \frac{1}{C_{i,j}} \right)$. However, we will only consider Eq. (4.13).

Furthermore, for the sake of assessing the efficiency of OPAC algorithm, three main cost factors are defined as follows:

$$vCache\ cost = \frac{\sum_{s \in S} \sum_{s' \in S', f \in F} x_{s,s',f} \times f_{size}}{\sum_{s' \in S'} C^{s'}} \tag{4.14}$$

$$vStream\ cost = \frac{\sum_{s' \in S'} \sum_{v \in V} \sum_{f \in F} y_{v,f}^{s'} \times d_v^f}{\sum_{s' \in S'} D^{s'}} \tag{4.15}$$

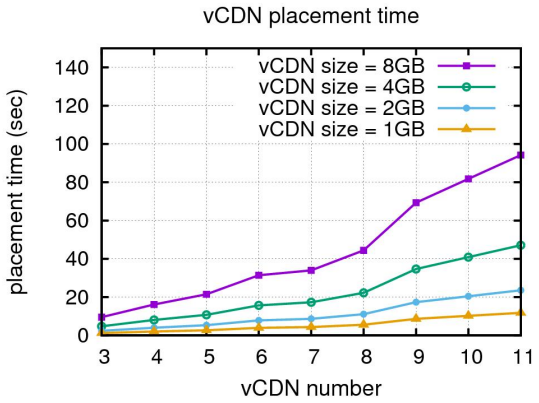
$$Link\ utilization\ cost = \frac{\sum_{v \in V} \sum_{f \in F} z_{i,j}^{v,f} \times d_v^f}{\sum_{i,j \in V \cup S'} C_{i,j}} \tag{4.16}$$

Where the cost UNIT represents the amount of system resources (vCache) or network resources (vStream and Link utilization cost) consumed when migrating a vCDN from a server to another one.

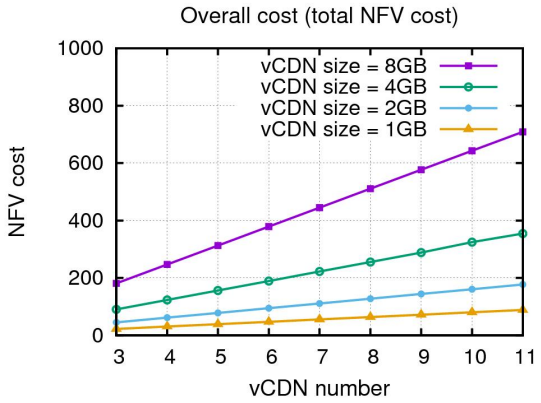
Hereafter, we evaluate the OPAC under different key performance indicators introduced above to assess impact of increasing virtual content delivery number, client group node number. Besides, we evaluate the virtual content delivery reso-

lution impact, delivery capacity impact and delivery storage impact.

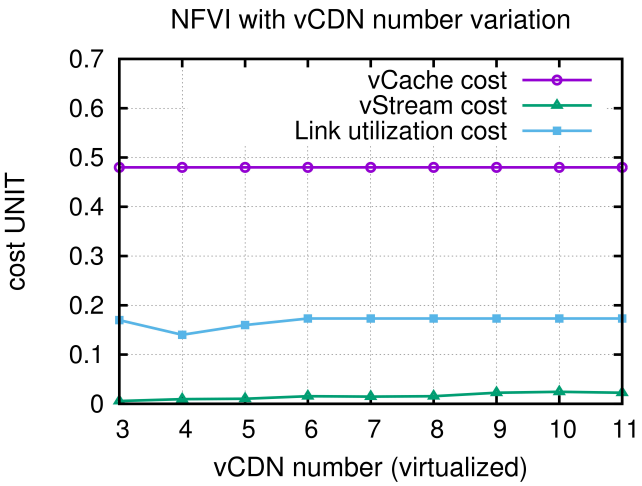
4.6.1 Virtual content delivery number impact



(a) vCDN placement cost



(b) Overall NFV cost of vCDN migration



(c) NFVI cost with vCDN number variation

Figure 4-9 – Network optimization costs: vCDN number impact

Firstly, we evaluate in this subsection the impact of increasing the vCDN number. For this, we choose for the three-tier network the following input parameters: $N = 6$ for client (as aggregation requests) node, $L = 21$ for links (number of aggregated edges), $S = 10$ for severs (as data center zone). Fig. 4-9a shows the migration time needed for placing vCDN in the optimal place taking into consideration the previous NFVI constraints (like computing, storage, and streaming resources) that

are strongly related to CDN caching service and virtualization cost. In Fig. 4-9b, we measure the overall NFV cost which means the overall cost of the migration including all the aforementioned constraints as a function of the vCDN number. We remark that the total NFV cost is linearly increasing. The vCDN placement time has an increasing slope for a vCDN number ranging from 3 to 9. This is due to the client node distribution which is not uniform. In Fig. 4-9c, cost unit [6] is plotted against the vCDN number. The storage cost for vCDNs in the network is constant, the virtual streaming cost and the link capacity cost are variable due to the demand variation of the end user.

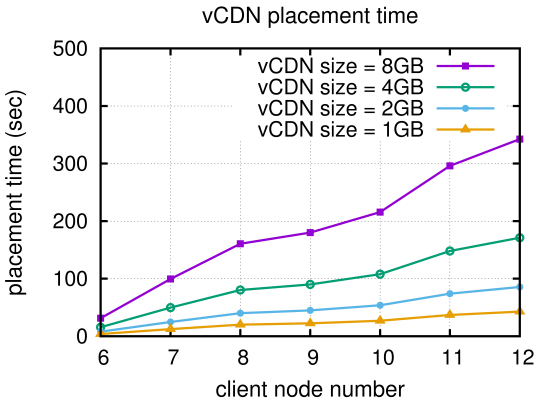
4.6.2 Client node number impact

Secondly, we evaluate the impact of the client number on the overall cost taking into account the storage, network and streaming features. Therefore, we have fixed the total number of the virtualized content delivery networks that should be placed within the network (VNF deployed number = 10), and the same previous three-tier network. In Fig. 4-10a, we plot the placement time against client node number. In Fig. 4-10b, we represent the NFV cost needed for this placement. In Fig. 4-10c, we measure the virtual caching cost. It is in increasing till it reaches a constant value. We represent also link utilization cost, and we conclude that the streaming cost is still increasing because it is affected by the number of location from where clients receive the video service.

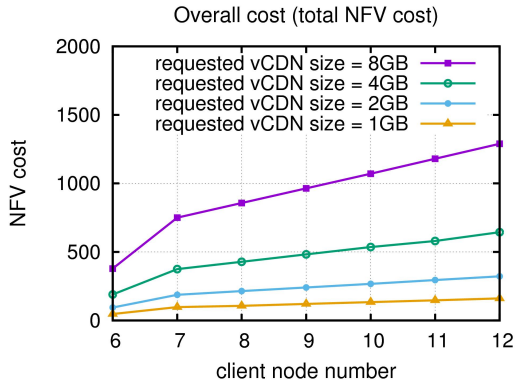
4.6.3 Virtual content delivery resolution impact

Thirdly, we evaluate the impact of virtual content delivery resolution. In Fig. 4-11a, we plot the cost of streaming against vCDN number. The vCDN caching/streaming nodes may serve SD, HD, or UHD content resolution to the clients. Recall that vCDN nodes serve the clients according to their initial requirements. Virtual streaming cost is increasing in each category when vCDN num-

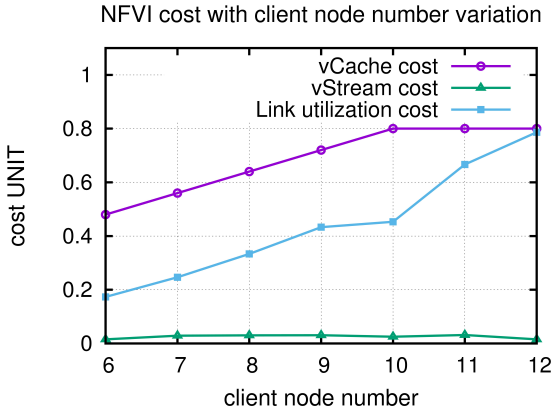
[6] Cost unit is defined by the utilization percentage of the available resource.



(a) vCDN placement cost



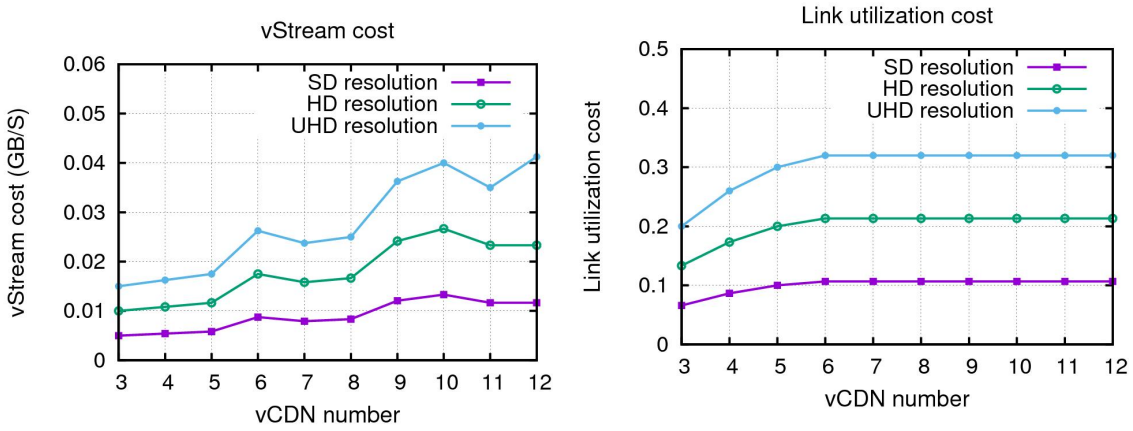
(b) Overall NFV cost of vCDN migration



(c) NFVI cost with client node number variation

Figure 4-10 – Network optimization costs: client node number impact

ber increase. Moreover, the additional cost needed for streaming high resolution video is significant compared to lower definition ones. There is about 100 % increase in cost from SD to HD and from HD to UHD. In Fig. 4-11b, we plot the cost of link utilization against vCDN number. Moreover, this link utilization cost increases with each content resolution when the vCDN number increases. As in virtual streaming cost (Fig. 4-11a), the additional link utilization cost needed for streaming high resolution video is significant. But, the main observation is that cost remains constant when increasing the vCDN number.



(a) Stream resolution

(b) Link utilization cost with different vCDN resolution

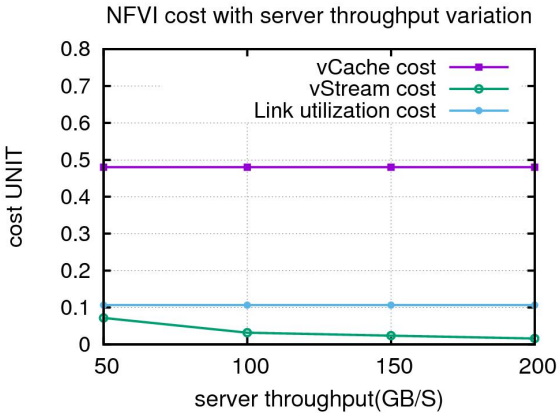
Figure 4-11 – Network optimization costs: vCDN resolution impact

4.6.4 Delivery capacity impact

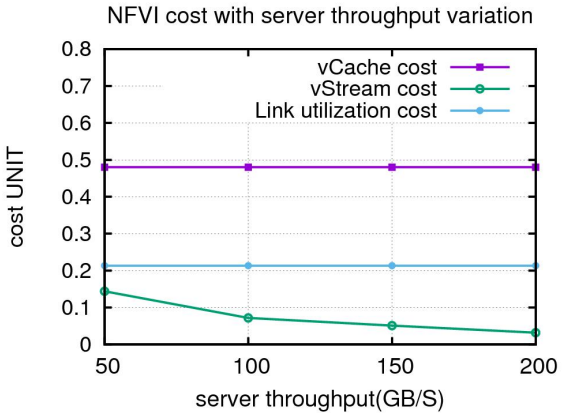
Fourthly, in this subsection we show the impact of increasing the server's throughput on the cost of caching, streaming vCDNs and on the link utilization. Fig. 4-12 includes 3 curves. These curves plot the cost unit against the server's throughput. We assume that clients may request either SD, HDV, or UHDV video quality. Regardless of the quality requested by clients, the caching cost is constant. There is no additional cost to deliver either SD, HD, or UHD to the client in terms of a caching cost in (GB). However, the streaming cost decreases when the server throughput increases. This leads to more efficient streaming and better QoE. This streaming cost increases when clients request high vCDN quality. Link utilization cost on the other hand is constant but it increases with high requested quality (from SD through UHD). In conclusion, in both cost factors (caching and streaming), the increment to get a better quality is not significant. It is only about 10 %.

4.6.5 Delivery storage impact

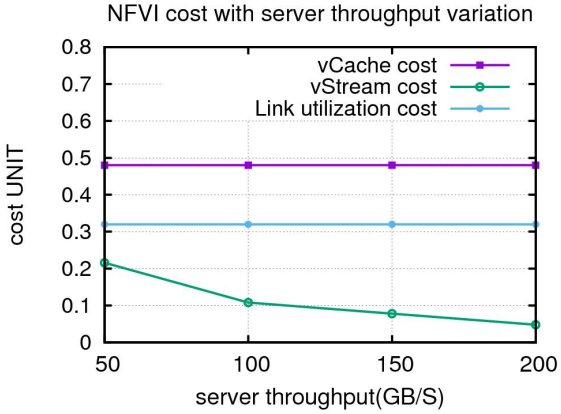
Finally, the impact of delivery storage is evaluated. In Fig. 4-13a, we plot the total migration cost needed for migrating a vCDN with random demand vector per



(a) SD requested quality



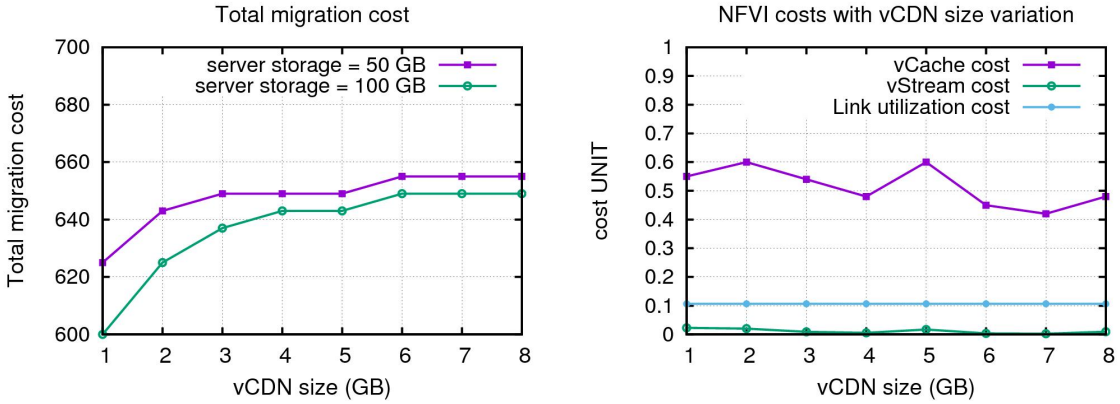
(b) HD requested quality



(c) UHD requested quality

Figure 4-12 – Network optimization costs: delivery capacity impact

client. The additional cost needed for vCDN migration decreases when we increase the server storage. This is an encouraging result since storage is an important service. Storage could be extended up the end user terminal. Fig. 4-13b plots our three proposed costs: vCache, vStream, and Link utilization against vCDN size. Although, the vCDN size increased, there are no important variations in the three cost factors.



(a) Total migration cost

(b) NFVI costs with vCDN size variation

Figure 4-13 – Network optimization costs: delivery storage impact

4.7 OPAC: comparisons

In this section, we introduce two main comparisons. First, we compare between OPAC (our optimal solution) and a non optimal approach. In the second, we compare between OPAC and the related work in two ways: 1) a comparison between OPAC and Bernadetta et al.'s work in term of link utilization cost, and 2) a comparison between OPAC and the state-of-the-art.

4.7.1 Comparison between OPAC and non optimal migration algorithm

The OPAC algorithm as detailed in Section 4.5 searches for the optimal placement to cache vCDN nodes according to the network constraints and users' demands. The network and system constraints are deterministic parameters and can be measured precisely over time. However users' demands can only be predicted as it is difficult to determine a priori what content the user will decide to watch next. Hence, some predictions can be erroneous and lead to non optimal placement of content. Thus, the optimal and the non-optimal definitions for our

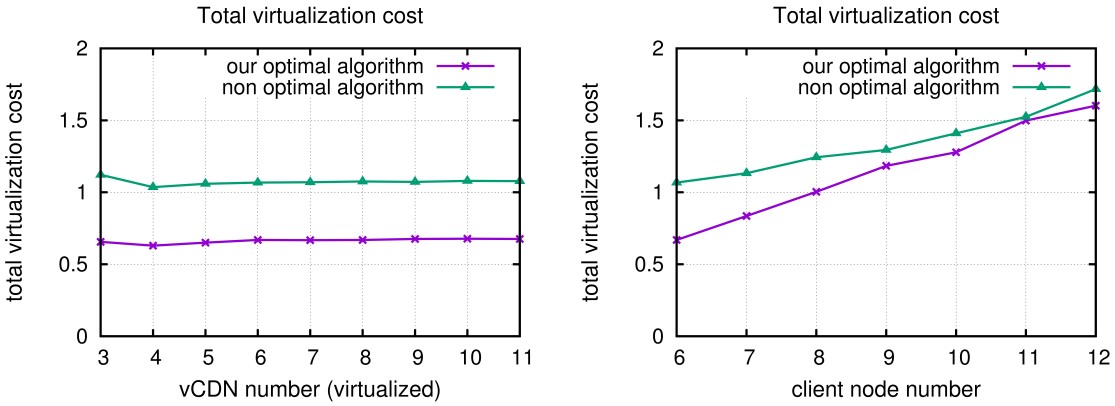


Figure 4-14 – comparison between the optimal and the non optimal algorithms using total virtualization cost parameter and under vCDN/client node variation

implemented algorithm are defined as follows:

$$x_{s,s',f} = \begin{cases} 1 & \text{if } f \text{ migrates to } s' \text{ (optimal)} \\ 0 & \text{if } f \text{ migrates to } S \backslash \{s'\} \text{ (non optimal)} \end{cases} \tag{4.17}$$

In Fig. 4-14 , we made a comparison in terms of the total virtualization cost (accumulated percentage of delay (normalized resource costs used in virtualization of vCDNs))[7] between our optimal algorithm and the non optimal one. It is clear that our exact and optimal allocation/migration algorithm outperforms the non optimal one. The operator's gain which is proportional to the reduction in delay is therefore obvious.

In Fig. 4-15, we plot the computation/execution time gained after the allocation/migration process. The gain is the reduction in delay if our optimal solution is adopted compared to the non optimal approach. The latter is obtained if the target PoPs that will host the virtual CDN is non optimal. It shows that the reduction in delay is noticeable and reaches 20 sec for $vCDNnumber = 11$. This can lead to a better QoE. However, the reduction delay is very important when client node number is increasing and reaches 3 $minutes$ at $v = 12$. Recall that v represents an

[7] Total virtualization cost is defined by the total utilization of the virtual resources (vCache, vStream, and Link utilization).

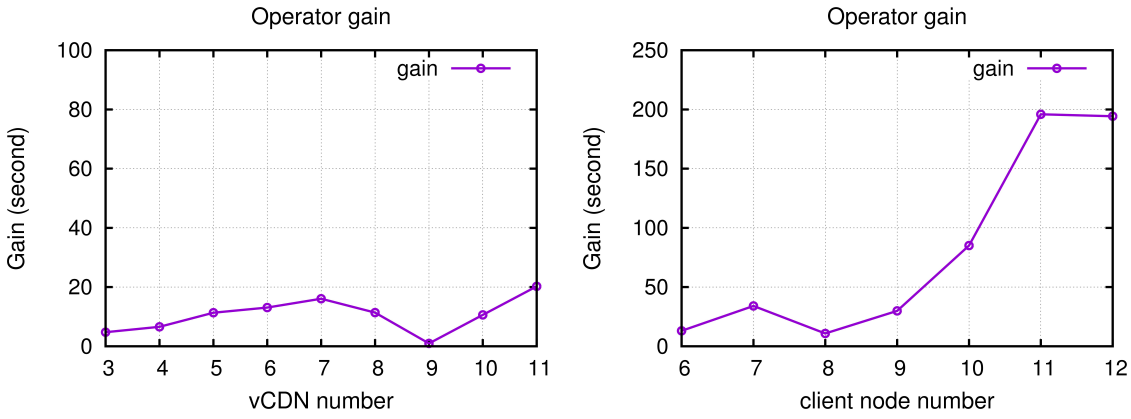


Figure 4-15 – Operator gain after executed the optimal migration algorithm comparing to the non optimal

average demand of a large group of subscribers and a vCDN serves a large number of subscribers as standardized by $(IETF)$[8] .

4.7.2 Comparison between OPAC and related work

Comparison between OPAC and Bernadetta et al. work

Network link utilization is a relevant metric. Thus, to evaluate the network performance of OPAC from the telecommunication operator perspective, we compared our solution with Bernadetta et al [3], who used the same network and NFV parameters.

At first glance, we quote the difference between our approach and Bernadetta approach as follows:

- First of all, Bernadetta protocol uses three flow balance constraints and imposes a single path flow balance. We believe that this increases the link utilization and causes bottlenecks in the network links (congestion).

- The second difference is that Bernadetta algorithm includes compression which we believe is unnecessary since it can be replaced by the VNF size.

[8]https://tools.ietf.org/pdf/draft-bagnulo-nfvrg-topology-00.pdf.

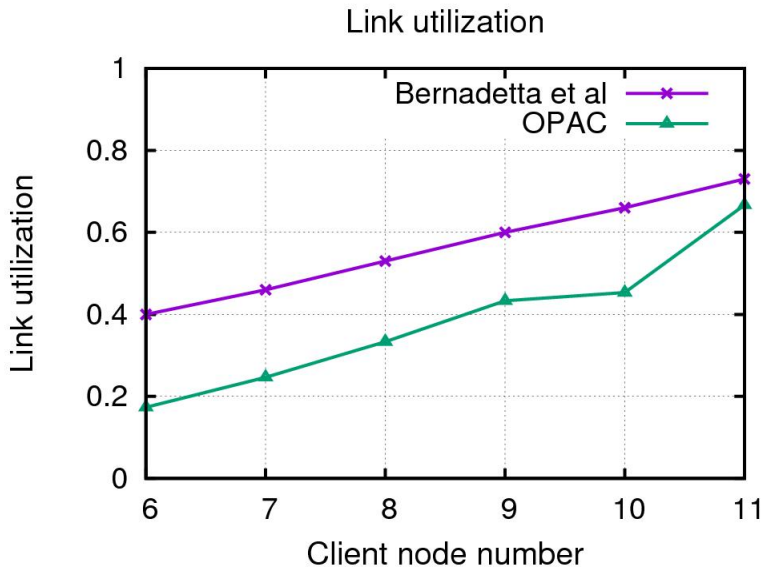


Figure 4-16 – Comparison between OPAC and Bernadetta et al work

- The third difference is that Bernadetta does not use any quality of experience parameter while our approach uses a dynamic one according to user's demand requirement.

Moreover, in Fig. 4-16, we plot the link utilization cost which is clearly greater than our approach. Thus, their proposed algorithm is still efficient for placing but can not deal with user's demand dynamicity ($d_v^f = 10$). In the simulation, we kept the same parameter used by Bernadetta (2 VNF type, node capacities).

Brief comparison between OPAC and the state-of-the-art

We show in Table 4.2 a comparison between our optimization approach and a few recent work:

- Authors in [28] [116] [27] [3] [122] introduced either system or network metrics in order to model the problem of placement of a VNF in general. However, we introduced in OPAC both: system and network metrics.
- Authors also did not introduce any parameter characterizing users' requirement. Therefore, we introduced an additional parameter to do this.

Table 4.2 – Comparison between OPAC and state-of-the-art

Work	***Metrics***	***Limitations***
Niels et al. [28]	Bandwidth, deployment cost	No virtual node cooperation, no QoE parameters, missed capacity and RAM constraints
Michele et al. [116]	Usage cost	Known traffic distribution (static), unreliable
Hendrick et al. [122]	Deployment cost, service request	Virtual CPU, virtual storage, virtual capacity, and QoE constraints are missed
Bernardetta et al. [3]	Allocated computing resources, link utilization	Prioritization method with orthogonal cost functions, virtual RAM and QoE constraints are missed
Mathieu et al. [27]	CAPEX, server's load	Offline measurement, and no migration
OPAC algorithm	System, network, and quality metrics	No limitations

- Authors also tried to solve an initial placement problem of a VNF through static and offline optimization, while our optimization solves the placement problem and adds the migration context in dynamic way and while the vCDN service is running.
- In OPAC, we introduced a virtual cache cost since our VNF (vCDN) is dedicated to streaming/caching services.

4.8 Conclusion

This chapter presents a new optimization technique for cache distribution with underlying virtualization tools. We proposed to integrate new constraints such as QoE, VNF mobility and system design. For this purpose, a redirection/cache/update protocol that defines the messages exchanges between end-user, virtual CDN, operator, and service provider was proposed. Furthermore, OPAC, an optimization algorithm was modeled, implemented and evaluated when demand on different

content changes. It gave a short execution time (i.e., the running time was in terms of seconds). Different comparisons for our placement solution and the state-of-the-art are conducted. The results show a net improvement in cache update. We also evaluate the update performance if the cache prediction is slightly imprecise. Our Cache placement algorithm still gives satisfactory results. Decision time in LP techniques increases exponentially for large network inputs (vCDN instances). They are not appropriate for large complex graphs. Therefore, in the next chapter, we try to propose scalable and cost efficient algorithms for vCDN migration problem in small large graphs.

Chapter 5

Scalable and cost efficient algorithms for vCDN migration

5.1 Introduction

In the previous chapter, we have proposed OPAC, an exact optimization algorithm for the problem of vCDN placement. In this chapter, we bypass the initial vCDN placement step that can be solved by a single execution of OPAC algorithm. Then, we re-formulated the mathematical model to propose an exact algorithm for vCDN migration[1]. It recommends optimal location points where to migrate vCDN on-demand/live instances to satisfy users quality requirements. The major objective from the algorithm is to minimize the total cost of content migration while minimizing the additional extra-costs needed for caching, streaming, and replication number.

Then, we are going to cope with scalability problems of exact algorithms. Therefore, we adapt a heuristic algorithm (i.e. HPAC: Heuristic Placement Algorithm for virtual CDN) to deal with our constraints when large scale networks need to be optimized. In this algorithm, we exploit the well known Gomory-Hu method to find a near to optimum point of operation. Finally, as the optimization algo-

[1] We still call this formulation OPAC.

rithms deal with many heteroclite parameters, a clear view on how/where/when they are extracted and how they are reflected on a typical SDN/NFV architecture is diagrammed.

The rest of this chapter is organized as follows: Section 5.2 details the functioning of OPAC algorithm through a migration use case. Section 5.3 details our heuristic optimization algorithm (HPAC). Section 5.4 evaluates the two algorithms and gives a comparison between them under predetermined metrics besides an integration diagram in Section 5.5. Section 5.6 concludes the chapter.

5.2 OPAC: migration use case (OMAC)

In this section, we specify the parameters and the constraints that are defined and proposed in formulating the optimization model OPAC. This formulation determines the migration of vCDN nodes to the optimal locations. We quote in Table 5.1 the main system and network parameters, and decision variables.

- *The decision variables*:

Table 5.1 – OPAC mathematical notation in a migration use case

Parameters	***Definition***
V	The set of client group nodes
S	The set of server nodes
s_v	The default server that connects the client group node $v \in V$
s_f	The server that initially caches the vCDN node $f \in F$
D^s	Maximum throughput of the streaming server $s \in S$
F	The set of vCDN nodes
f_{size}	vCDN' s size $(vRAM, vCPU, vDISK)$ $(f \in F)$
C^s	Maximum storage capacity of the server s
$L_{i,j}$	Link capacity between two nodes i and j (from i to j)
d_v^f	Throughput used for streaming the functionality f to the client group $v \in V$. It represents the user's demand requirements
m_f^s	The migration cost of the functionality f to s
Decision variables	***Definition***
x_f^s	Placement and migration binary variable which indicates that f should move from origin server to optimal server s
$y_{v,f}^s$	Binary variable which indicates the video hit from node v of f in server s
$z_{i,j}^{v,f}$	Binary variable indicating whether the link (i, j) is used to stream f to v

1. The binary variable x_f^s indicates the placement of the streaming head-end, and its migration from one server to another (best/optimal) location s. It is defined as:

$$x_f^s = \begin{cases} 1 & \text{if } f \text{ migrates to } s \\ 0 & \text{Otherwise} \end{cases} \tag{5.1}$$

2. The binary variable $y_{v,f}^s$ indicates a client v needs a vCDN service, and the server s caches it. It is defined as :

$$y_{v,f}^s = \begin{cases} 1 & \text{if } v \text{ needs } f \text{ and } s \text{ caches } f \\ 0 & \text{Otherwise} \end{cases} \tag{5.2}$$

3. The binary variable $z_{i,j}^{v,f}$ indicates whether a link (i, j) is used (from i to j) to stream from a server s replicating f (the one for which $y_{v,f}^s = 1$) to client v.

- *The constraints*:

1. The binary variable y should be less than or equal to x. In fact, y equals to 1, if and only if v needs f, and f is located on server s. This means that x should be equal to 1. Otherwise, if y equals to 0, then x may be equal to 0 or 1.

$$\forall s \in S : y_{v,f}^s \leq x_f^s \tag{5.3}$$

   Note that a vCDN can be replicated more than once.

2. Only one optimal server should serve the client node that requests the functionality f:

$$\forall v \in V \mid d_v^f \neq 0 : \sum_{s \in S} y_{v,f}^s = 1 \tag{5.4}$$

3. The cost of streaming f by a server s should be less than or equal to the

maximum server capacity:

$$\forall s \in S : \sum_{v \in V} \sum_{f \in F} y_{v,f}^{s} \times d_{v}^{f} \leq D^{s} \tag{5.5}$$

4. The storage of the optimal server should not exceed its maximum capacity:

$$\forall s \in S : \sum_{f \in F} x_{f}^{s} \times f_{size} \leq C^{s} \tag{5.6}$$

5. Flow balance or conservation constraint between the server s and the client node v should be as the following:

$$\sum_{j} z_{i,j}^{v,f} - \sum_{j} z_{j,i}^{v,f} = \begin{cases} 0 & \text{if } i \neq v, i \neq s \\ y_{v,f}^{s} & \text{if } i = s \\ -1 & \text{if } i = v \end{cases} \tag{5.7}$$

This is the network flow constraint. The sum of incoming flows must be equal to the outgoing ones.

6. Link capacity between source i and sink j should be larger than the flow on the link:

$$\forall i, j \in V \cup S : \sum_{v \in V} \sum_{f \in F} z_{i,j}^{v,f} \times d_{v}^{f} \leq L_{i,j} \tag{5.8}$$

The objective function is formulated in equation (5.9) (cost function):

$$\min \sum_{s \in S} \sum_{f \in F} x_{f}^{s} \times m_{f}^{s} \tag{5.9}$$

where m_{f}^{s} is a parameter depending on the position of s, the position of s_f (the server initially containing f before migration), and the position of s_v (the server initially connecting to the client group v). m_{f}^{s} depends also on the size of f and the operator policy.

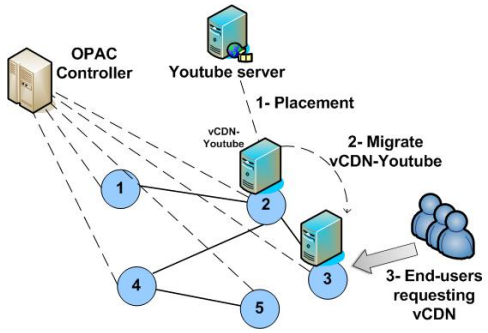


Figure 5-1 – OPAC: vCDN-YouTube migration use case

OPAC based vCDN migration example: For the sake of clarity, we propose an example of OPAC based vCDN migration as shown in Fig. 5-1 . The algorithm is as follows: 1) As an input, the algorithm gathers SDN/NFV architectural information such as the initial placement of CDNs/vCDNs and all necessary dynamic parameters, 2) It needs also a prediction of end-users demands to execute OPAC, and 3) as a result, end-users requesting vCDN YouTube®, to watch a popular video in a live event for example, will be redirected to the videos hosted in the new optimal calculated location.

Recall that the network operator hosting the vCDN of YouTube content provider migrates the vCDN cache/stream node to an optimal PoP where resources are available and content quality streamed are satisfied.

The above problem is NP-hard due to our combinatorial complex system and therefore the proposed exact algorithm is difficult to scale up to decide where to deploy vCDN nodes in a large scale scenario. As a consequence, an efficient heuristic algorithm is proposed in the next section.

5.3 HPAC: Heuristic Placement Algorithm for virtual CDN

Our proposal is strongly based on Gomory-Hu [2] tree transformation [73] of the initial network (represented by access, aggregate and core nodes) as shown in Fig. 5-4). In other words, HPAC transforms the input network into a G-H tree. Then,

[2]noted by G-H in the rest of the chapter

replicating vCDNs is performed thanks to the G-H tree allowing to efficiently reduce the number of edges to be considered when migrating contents.

Algorithm 2 summarizes the pseudo code of HPAC. Hereafter, we describe these main stages.

Algorithm 2 Heuristic algorithm for virtual CDN placement and migration

1: **Input:** V, S, D^s, F, f_{size}, C^s, $L_{i,j}$, d_v^f, m_f^s, $G = (V(G), E(G))$,
2: s_v, s_f
3: **Output:** x_f^s, total migration cost
4: $GHT \leftarrow$ Gomory-Hu transformation $\left(G, L_{i,j}\right)$
5: Tree-Exploration $\left(GHT, s_v, s_f\right)$
6: Migration-process$\left(D^s, C^s, L_{i,j}, f_{size}, d_v^f, m_f^s\right)$:
7: **if** $L_{i,j}, \leq d_v^f$ **then**
8: Migrate-vCDN$\left(\ \right)$
9: **end if**

5.3.1 Gomory-Hu transformation

At this stage, our heuristic HPAC computes the G-H tree of the CDN network or graph. The main advantage of G-H tree transformation is to compact this network graph structure using cuts to retain only feasible candidate topology and consequently lead to a smaller scale vCDN placement problem. We will insert a figure of the example of a G-H transformation.

For sake of clarity, we introduce the G-H tree transformation of our initial graph $G = (V(G); E(G))$. The G-H tree $GHT = (V(GHT); E(GHT))$ of the former graph can be built using the definition in [106]. There exists many algorithms that can build the G-H tree in polynomial times [80] by finding $N - 1$ maximum flow or minimum cuts between each pair of nodes in the graph, where $N = |V(G)|$. The G-H tree can be found in polynomial time according to the algorithm used to find a maximum flow/minimum cut in the initial graph. But, we describe only one selected strategy that relies on the minimum Steiner cut algorithm presented in [39].

The transformation starts by initializing $V(GHT)$ to the set of the graph nodes (i.e., $\{V(G)\}$ and not $V(G)$). Concerning $E(GHT)$, it is initialized to an empty set. Besides, we define a queue list Q to enable the G-H tree construction. Q is initialized to $V(GHT)$ value. Then, while Q is not empty, we pull the first element S from this queue and we apply the minimum Steiner cut algorithm with the current set S (i.e., first Q element).

As the Steiner set in the new graph is obtained by contracting the entire subtree rooted at each neighbor in GHT of S into a single node. Consequently, two new sets of nodes S_1 and S_2 are generated from S and λ_{S_1,S_2} is the cut size. Accordingly, $V(GHT)$ should be updated by removing S and adding S_1 and S_2. Besides, $E(GHT)$ should be enhanced by adding a new edge between S_1 and S_2 with capacity λ_{S_1,S_2} (i.e., the cut size).

Furthermore, the queue Q will be enriched by adding S_1 (respectively S_2) if it includes at least two nodes, $|S_1| > 1$ (respectively $|S_2| > 1$). The former steps will be repeated until the G-H tree construction matches with an empty queue Q state.

The pseudo-code of the following G-H tree building strategy is summarized in Algorithm 3.

Algorithm 3 Gomory-Hu tree transformation algorithm

Input: A connected graph $G = (V(G), E(G))$
Output: A tree $GHT = (V(GHT), E(GHT))$
$V(GHT) = V(G)$, $E(GHT) = \emptyset$, $Q = \{V(G)\}$
while $Q \neq \emptyset$ **do**
  $S \leftarrow$ Pull(Q);//pull the first element from Q
  $\{S_1, S_2\} \leftarrow$ minimum-Steiner-Cut(S, GHT)
  $V(GHT) = \{V(GHT) \setminus S\} \cup \{S_1, S_2\}$
  $E(GHT) = E(GHT) \cup (S_1, S_2)$
  if $|S_1| > 1$ **then**
    5: $Q \leftarrow Q \cup S_1$
  end if
  if $|S_2| > 1$ **then**
    $Q \leftarrow Q \cup S_2$
  end if
end while

We give a simple example of a G-H tree transformation of a graph G as illustrated in Fig. 5-2. It is straightforward to see that the number of edges in the tree

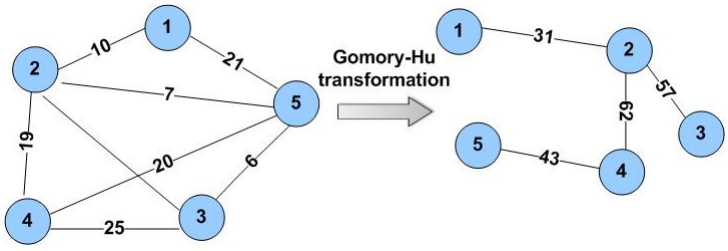


Figure 5-2 – Example of a Gomory-Hu tree transformation.

compared with the initial graph is reduced by 50%. Note that this G-H tree was built using only 6 iterations of Algorithm 3.

5.3.2 HPAC: placement and migration

Based on the G-H tree transformation, the number of nodes is equal to the number of nodes in the initial graph/network. However, the number of links in the derived G-H tree is at most equal to the number of nodes in the graph. Accordingly, the set of all paths can be easily computed. In fact, in the tree structure there is only one path between any couple of servers.

With this tree transformation leading to significantly reduced input sizes, the problem of vCDN placement and migration will be easily solved even for large problem instances (large number of nodes and arcs). This is due to the efficiency of the G-H tree transformation leading to reduce considerably the domain of feasible solutions. So, it allows us to find the near optimal solutions in few seconds. In the following, we give more details on the second stage of the HPAC algorithm to place and migrate vCDNs in a cost efficient manner.

The second stage of HPAC algorithm consists to explore the unique path from an access point to the server containing the vCDN. Thus, if the ingress flow cannot reach the destination (i.e. the vCDN), caused by a rupture node, then we will simply migrate the required vCDN to the rupture node of the G-H tree.

We propose the following example (see Fig. 5-3) to illustrate HPAC algorithm for migrating and placing judiciously vCDNs. It depicts a scenario of con-

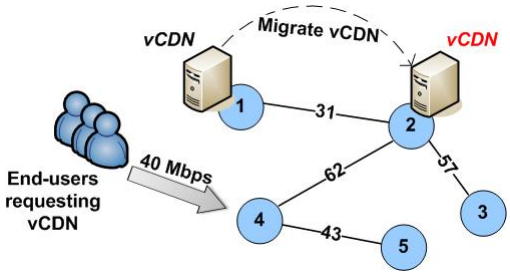


Figure 5-3 – Example of HPAC vCDN content replication/migration.

tent replication/migration. In this scenario, a client cluster (group) representing grouped end-user requests for a vCDN with a specific throughput (i.e., content quality) equal to 40 $Mbps$. The proposed migration method searches to migrate the desired vCDN from its initial deployed position (server 1) to the suitable server (i.e., server 2 which has more than the required streaming capacity) in an efficient way (i.e., with minimum migration cost).

5.4 OPAC versus HPAC (exact versus heuristic)

For the interest of assessing the efficiency of OPAC and HPAC algorithms, we used [84] and [139] as optimization tools. In addition, different metrics/cost-factors can be defined as follows:

$$Migration\ cost = \sum_{s \in S} \sum_{f \in F} x_f^s \times m_f^s \tag{5.10}$$

$$Migration\ time = \sum_{s \in S} \sum_{f \in F} x_f^s \times f_{size} \times \max_{(i,j) \in P_{s_f,s}} \frac{1}{L_{i,j}} \tag{5.11}$$

Where $P_{s_f,s}$ is a given path from s_f to s. In fact, we are here assuming that the migration is done in a sequential way. If the migration is performed in parallel, then the migration time would be given by: $\max_{s \in S, f \in F} \left(x_f^s \times f_{size} \times \max_{(i,j) \in P_{s_f,s}} \frac{1}{L_{i,j}} \right)$.

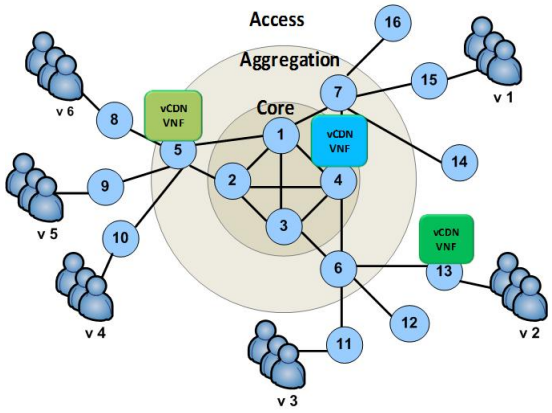


Figure 5-4 – Network topology used for small scale

However, we will only consider (5.11).

$$Replica\ number = \sum_{s \in S \setminus \{s_f\}} \sum_{f \in F} x_f^s \tag{5.12}$$

$$vCache\ cost = \frac{\sum_{s \in S} \sum_{f \in F} x_f^s \times f_{size}}{\sum_{s \in S} C^s} \tag{5.13}$$

$$vStream\ cost = \frac{\sum_{v \in V} \sum_{f \in F} y_{v,f}^s \times d_v^f}{\sum_{s \in S} D^s} \tag{5.14}$$

Moreover, in order to decide which algorithm should we use and when, different network scales (i.e. small and large) and topologies are considered as follows:

5.4.1 Small scale scenario: a network operator snapshot

In this subsection, the optimization targets a small number of vCDNs. Therefore, a snapshot of a three-tier network operator architecture [3] is used for the evaluation as show in Fig. 5-4. Moreover, the main dissimilarities between the two approaches according to the defined metrics are showed hereafter:

Algorithm complexity: The algorithmic complexity of G-H-based HPAC is polynomial while it is exponential in OPAC.

[3]NFVI PoP Network Topology (2016): https://tools.ietf.org/pdf/draft-bagnulo-nfvrg-topology-01.pdf

The run-time: The time convergence of the two algorithms is shown in the Fig. 5-5a . HPAC outperforms OPAC since the latter is a combinatorial-based algorithm and has an exponential complexity.

The total migration time: It is the duration needed to migrate a vCDN to the optimal/near optimal point of the deployment. Fig. 5-5b shows the average migration time needed for migration vCDN delivery functions to the optimal instantiation point taking into account the NFV constraints including system, network, and content quality parameters which are related to the vCDN functionalities. Although HPAC gives the shortest time as depicted in this figure, OPAC is still acceptable. Indeed, This metric is strongly related to the distance between servers as written in (5.11) and the time consumed for the vCDN migration process is less than $1 minute$ for vCDN numbers equals 11.

The total migration cost: It is evaluated in both approaches (exact and heuristic). In Fig. 5-5c , the vCDN total migration cost is measured in terms of *Gigabits* (*Gb*) against vCDN number. The Fig. 5-5c shows that vCDN-migration cost is increasing in both approaches for vCDN number ranging from 3 to 6. This is due to the non uniform client group distribution. It is noticeable that OPAC outperforms HPAC for vCDN ranging from 6 to 11. Nevertheless, HPAC proofs a low variation cost which leads to a better save of operator resources.

The total replication number: Under a random end-users matrix demand, the total replication number in both approaches is not significant (small) as shown in Fig. 5-5d. However, HPAC is the strictest approach when replicating vCDN in small scale scenario.

Other cost-factors: OPAC and HPAC are compared in terms of vCache and vStream cost-units [4]. In Fig. 5-5e cost unit is plotted against vCDN number. The virtual in-networking caching cost increases in both algorithms but they are still insignificant. Similarly, in Fig. 5-5f the virtual in-network streaming cost increases with vCDN number in both algorithms but it is still insignificant. Although the

[4]Cost-unit may be defined by the utilization percentage of the available system or network resource.

Table 5.2 – Gap (migration cost): HPAC efficiency

vCDN number	6	7	8	9	10	11
Gap (%)	0.66	0.42	0.25	0.62	0.62	0.3

HPAC is slightly cost-efficient, both algorithms proofed an efficient network resource saving.

Experiments show that OPAC is still acceptable and saves about 96 % of the system resources and 94% of the network resources at $|F| = 11$. Nevertheless, it is still the most costly comparing to HPAC in terms of system and network resource usage.

For the sake of clarifying the effectiveness of the HPAC, we define the Gap metric as follows:

$$Gap(C) = \frac{C_{HPAC} - C_{OPAC}}{C_{OPAC}} \tag{5.15}$$

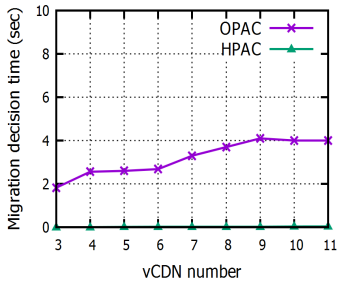
Where C is a cost factor (e.g., migration cost).

Table 5.2 depicts the Gap metric calculation formulated in (5.15) . It demonstrates the efficiency of the HPAC algorithm in terms of total migration cost since the Gap values are close to zero.

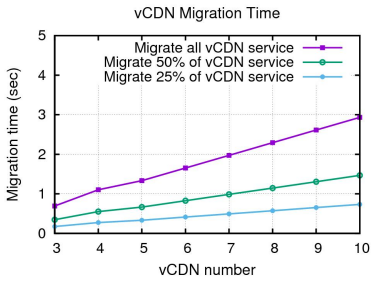
5.4.2 Large scale scenario: an Erdos-Renyi graph-based network operator

In this subsection, the optimization targets a large number of vCDNs (µ vCDNs or containers). The scenario relies on the well known Erdos-Renyi undirected and weighted graph. The graph has 100 vertices (nodes) and 200 edges as shown in Fig. 5-6a. Its G-H-based transformation is shown in Fig. 5-6b which has only 99 edges (49.5%). These figures depict the topology used for evaluating HPAC algorithm in a large scale scenario. The OPAC algorithm could not be used here with reasonable resources.

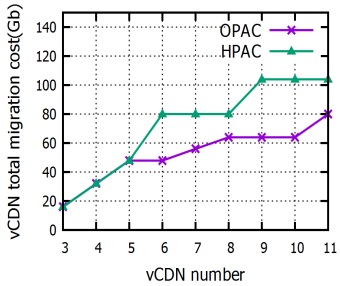
Fig. 5-7a and 5-7c depict the total migration delay time and the total migration cost respectively. Further, Fig. 5-7b shows the total replication number. These



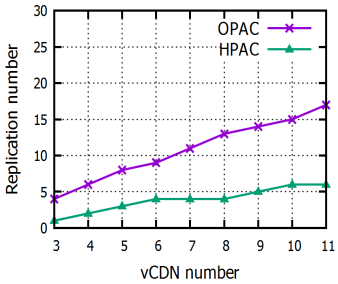
(a) Migration decision time



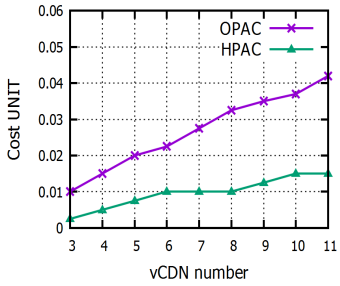
(b) Migration time



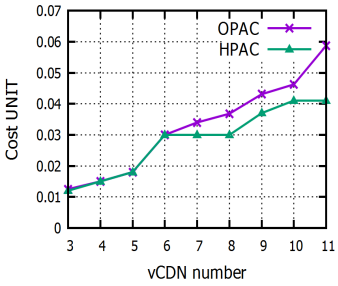
(c) vCDN total migration cost



(d) Replication number



(e) Virtual cache



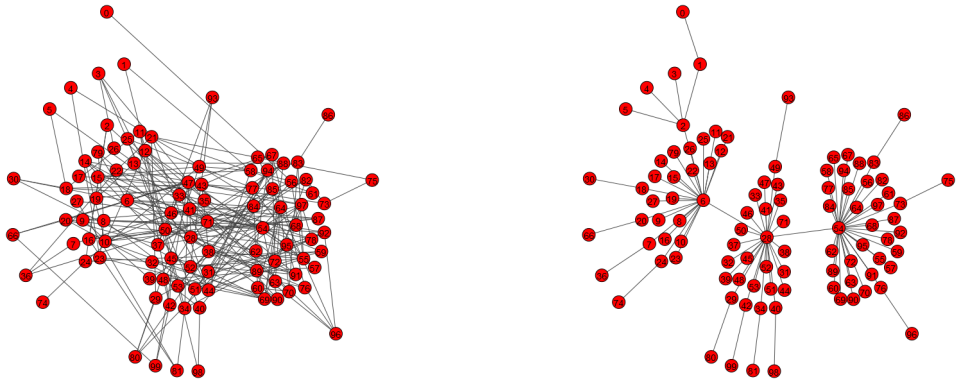
(f) Virtual stream

Figure 5-5 – OPAC-HPAC comparison in the small network scale scenario.

metrics are increasing until a decreasing slope at $|F| = 80$ representing the average vCDN number stabilizing the system. Furthermore, Fig. 5-8 shows the run-time of the OPAC and HPAC approaches in a large scale scenario. It demonstrates the feasibility/efficiency of HPAC since its run-time is in terms of a few seconds (6 *sec*) while OPAC run-time explodes from vCDN number equals 20.

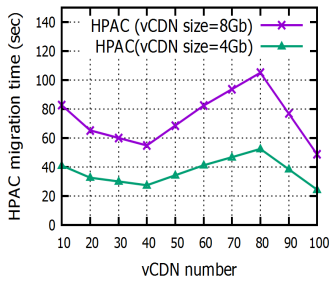
5.4.3 Interpretations

In this subsection, we quote the interpretations that can be revealed from the results above:

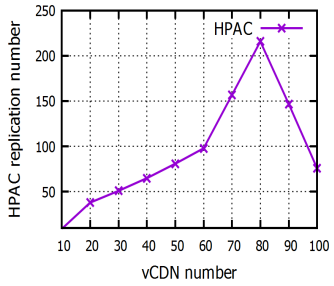


(a) Network topology (b) G-H-based transformation

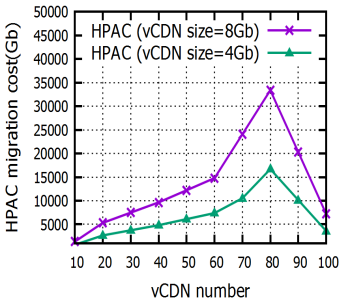
Figure 5-6 – Network topology used for large network scale.



(a) Migration time/delay.



(b) Replication number.



(c) Migration cost.

Figure 5-7 – HPAC in large network scale.

In small scale scenario: (the order of ten servers, ten client groups, and ten vCDNs), the exact approach is suggested to be used because its run-time is in terms of a few seconds (see Fig. 5-5a. OPAC is what we should choose as an optimization mechanism because it gives the lowest migration cost as shown in Fig. 5-5c .

In large scale scenario: (the order of hundred servers, hundred client groups,

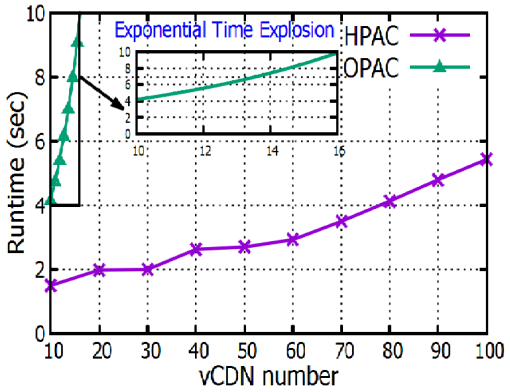


Figure 5-8 – OPAC and HPAC run-time in large network scale.

Table 5.3 – Efficiency comparison between OPAC and HPAC; SS: Small Scale,LS: Large Scale $N = |V(G)|$, $M = |E(G)|$

Metrics	***OPAC***	***HPAC***
Algorithm complexity	Exponential	$\mathcal{O}(N^3 * M^{1/2})$
Run-time	A few seconds in SS; a few minutes in *LS*	A few second in *SS/LS*
Total migration cost	Excellent in SS;unfeasible in *LS*	Good in *SS/LS*
Total migration time (i.e.,delay)	Good in *SS*;unfeasible in *LS*	Excellent in both *SS/LS*
Replication number	Free/loose in *SS*; unfeasible in *LS*	Free/loose in *SS/LS*
vCache/vStream cost	Good minimization in *SS*	Excellent minimization in both *SS/LS*

and hundred vCDNs), the exact approach is suggested to be useful and the G-H-based HPAC algorithm is the alternative to solve the vCDN migration problem since its algorithm run-time is in terms of a few seconds as shown in Fig. 5-8 .

A brief comparison between the two approaches which answers implicitly to the question when to use OPAC and when to use HPAC is given in the Table 5.3 .

5.5 Integration of the Algorithms

Practically, the proposed optimization algorithms have to be integrated in a vCDN controller (as seen Fig. 5-1). It interacts with the vCDN manager software in a legacy NFV-MANO framework [172]. Further, as the migration problem deals with different heteroclite parameters and variables, a clear view about how they are extracted and reflected on an SDN/NFV framework is necessary.

In Fig. 5-9 and for the sake of simplicity, the two main use cases needed for integrating OPAC/HPAC algorithms by the network operator are depicted. Indeed:

- Use case 1: The network operator checks the current placement of vCDNs. To do this, he queries a Database using SQL structured language and an NFV/SDN controllers (e.g., Openstack [149]/Opendaylight [128]) using API interfaces. Openstack horizon and Opendaylight DLUX providing respectively the system and network resource information needed to complete the operator database.

- Use case 2: The network operator requests for the vCDN optimization placement/migration result using either OPAC or HPAC according to the actual network scale (small or large).

From an operator perspective, integrating the proposed algorithms may follow the sequence diagram represented by the Fig. 5-10 . Indeed, the main stakeholders are:

- User (Operator): Operator of the infrastructure, deciding the migration or not of a vCDN instance according to the current state of the environment and the result given by the optimization algorithms OPAC/HPAC.

- System: This system, offering the Users an interaction with the OPAC/HPAC algorithms inputs and results. It still maintains the operator database updated by the necessary information about the operator's servers (*getHypervisor()*) and the provider's vCDNs (*getStatus()*).

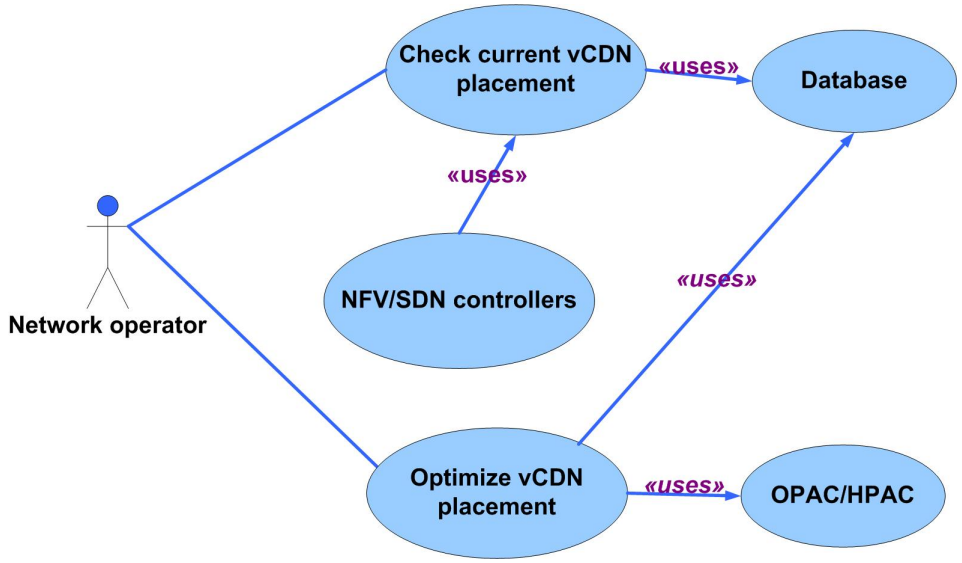


Figure 5-9 – Main use-cases for OPAC/HPAC in an SDN/NFV framework.

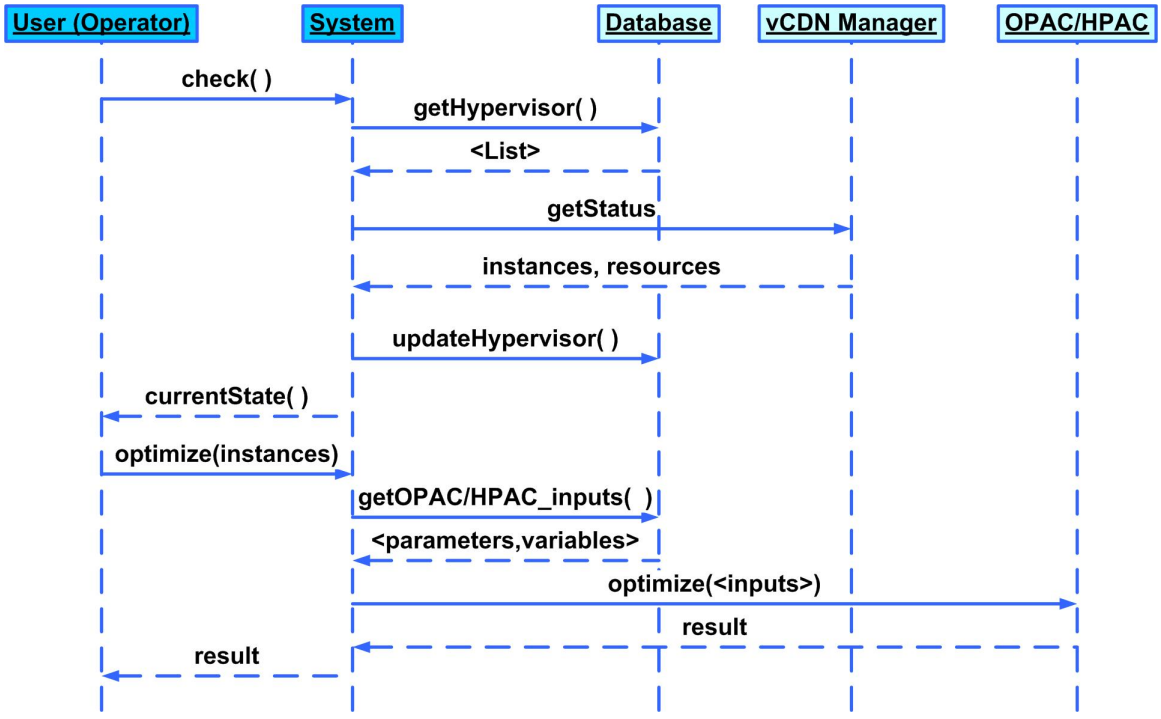


Figure 5-10 – Sequence diagram for OPAC/HPAC integration.

- Database: It contains the information about the SDN/NFV infrastructure status and values. In this case, it is an SQLite database.
- vCDN Manager: It takes care of the interaction with the software managers running in the architecture, to poll from them the required information for the OPAC/HPAC algorithm. It will use internally API calls to the different software managers (Openstack Horizon, Opendaylight, etc.).
- OPAC and HPAC Algorithms: Implementation of the optimization algorithms that optimize the vCDN placement and migration based on exact/heuristic optimization approaches. A file is given as an input and a file is returned as an output of the selected algorithm. The algorithm selection depends mainly on the network scale.

For simplicity, when the user (operator) executes the optimize command, the system fetches the operator databases to get the required information ($get\_input()$) in order to launch in turn the optimize command. Then, OPAC/HPAC algorithm decides where to place and migrate the vCDNs and provide the system/operator the result. Finally, the system executes the migration process according to our results and releases the dedicated resources in case of Service Level Agreement (SLA)-expiration/vCDN-delete-request.

Recall that the proposed algorithms involved also the content provider who wanted to rent a cloud of vCDN for its customers. Therefor, from a content provider perspective, the main exchanges between the content provider, the vCDN manager and the user (operator) are depicted:

- The content provider (e.g., YouTube) requests a vCDN creation during a specific time (e.g., 2 hours) and with a specific QoE (e.g., excellent) and to cover a specific region (e.g. Paris).
- The network operator, representing the owner of the NFV infrastructure, checks the current state of its NFV resources and call the proposed optimization algorithms through the provided *system*.

- This system interacts with a operator database server to retrieve the required information about network topology, NFV resources and update another database dedicated to the placement/migration decision.
- The NFV Infrastructure (NFVI) administrator interacts with the decision's database to migrate the resources while keeping the signed SLA between the content provider and the network operator valid and standing.
- Optionally, the NFVI administrator may let the content provider as the manager of the video content.
- The decision database is updated by the operator for security issues.
- In case of SLA expiration (e.g, covering time expiration, vCDN delete-request, etc..), the operator releases the dedicated NFV resources.

5.6 Conclusion

This chapter presents two optimization solutions either for the placement or the migration problem of virtual CDNs. Multiple constraints that are strongly related to the CDN virtualization are taken into account such as vCDN size, content resolution and system/network requirements. In addition, the two optimization algorithms OPAC and HPAC target respectively different network scales. Then, they are modeled and implemented. In small scale networks, results show that $|F|$ have more significant impact on the average migration cost in the case of using HPAC rather than OPAC. Nevertheless, it is noticeable that HPAC converges to the optimal solution and outperforms OPAC in specific metrics. In large scale, HPAC gives a short execution time (e.g. the run-time was in terms of a few seconds) which proofs its efficiency and scalability. In this network scale, The HPAC's results show that $|F|$ have insignificant impact on the migration time, the migration cost and the replication number since the system reaches quickly its stability. Moreover, the two approaches are integrated in an SDN/NFV framework and the

main use case, sequence diagram from either an operator or content provider perspective.

In the next chapter, we extend the aforementioned placement algorithms to service orchestration problems. In this case, vCDN components should be orchestrated (chained) and service flows should be mixed together in order to satisfy end-user requirements (e.g. a content with ad flows).

Chapter 6

Optimal and cost efficient algorithm for vCDN orchestration

6.1 Introduction

In the previous chapter, we have investigated the main optimization techniques for vCDN placement and migration problems in small and large network scales. Currently, despite the importance of vCDN optimization tasks, service orchestration module is still missing in the global architecture. We therefore, try in this chapter to contribute with vCDN management and orchestration architecture beside an optimal orchestration algorithm and explain how it is integrated in the operators virtual LAN.

The proposed optimization algorithm in this chapter considers vCDN orchestration problem inside the mobile network operator. Moreover, the major objective from the proposed algorithm is to minimize the total orchestration cost of vCDN components (vCDNC) while minimizing the additional extra-costs needed for caching, streaming, and replicating the virtual instances. Through this optimization (i.e. OCPA: Optimal vCDN Placement Algorithm), we are going to formulate an exact algorithm based on a linear programming model for deciding the optimal locations to place vCDN components to satisfy users quality requirements

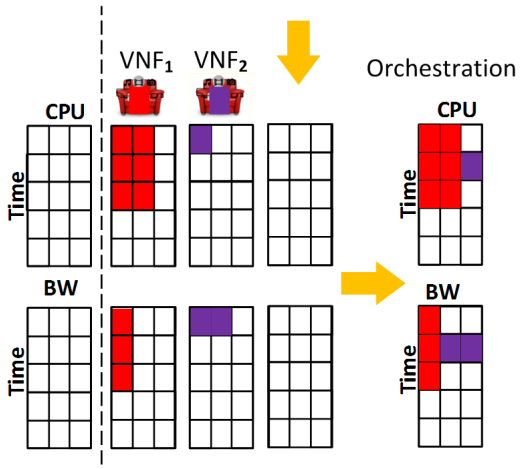


Figure 6-1 – vCDN network orchestration

and minimize the overall load on the main nodes and links.

The rest of this chapter is organized as follows: Section 6.2 describes the proposed vCDN architecture in the NFV operation field. Section 6.3 details OCPA parameters, constraints and its objective function. Section 6.4 evaluates the proposed algorithm and gives a comparison between them under predetermined metrics. Section 6.5 quantifies the algorithm behavior under different scenarios. Section 6.6 concludes the work.

6.2 Virtual CDN orchestration architecture for the NFV deployment

6.2.1 How to orchestrate ?

vCDN is not a single network component. It is a complex network of virtual functions such as caching, streaming, routing, advertising, and so on. Orchestrate a vCDN means the placement of these functions and the scheduling of all software resources in time and space. From an optimization point of view, this corresponds to a new problem in which we have several services to chain and orchestrate. It is necessary to pass the flow by several virtual servers. In Fig. 6-1, we show a vCDN

network that has two components (VNF_1, and VNF_2). They represent respectively a content and ad servers. We suppose also that we have two software resources (CPU processing and BW throughput capabilities). The first component requires (2 CPU, 1 BW) for three time slots and the second component requires (1 CPU , 2 BW) for one time slot. As depicted in the figure, an orchestrator module allocates system and network resources to VNFs.

6.2.2 ETSI-MANO-based vCDN orchestration architecture

ETSI-MANO standardizes a framework for deploying different Virtual Network Functions (VNFs) [1]. In our case, CDN is the target VNF. We followed this standard and proposed our specific design and architecture for CDN. This latter (vCDN) requires those virtual network functions (VNFs): request routing, caching, streaming, cryptography, migration, advertisement, measurement, content adaptation, and an orchestration. Those VNFs are controlled by a Virtual Network Function Manager (VNFM). This entity executes the proposed OCPA algorithm for orchestrating and optimizing the placement and the migration of vCDN VNFs on top of NFV physical infrastructure (NFVI). Indeed, according to the customer requesting vCDN creation and orchestration service, a dynamic placement (orchestration) of the vCDN components is handled in an intelligent way.

NFVI is composed of three domains: i) virtual computing domain, ii) virtual storage domain, and iii) virtual networking domain. NFVI is managed by cloud management platform (eg., OpenStack [149]) which corresponds to the Virtual Infrastructure Manager (VIM) that manages. The proposed vCDN architecture has a global orchestration that manages and orchestrates the CDN VNFMs (if there is multiple), OpenStack, and OSS/BSS which manages QoS, network failure, and security.

We propose a simplified vCDN architecture With respect to the ETSI-MANO standard as depicted in Fig. 6-2. Indeed, it has the following communication

[1] It is software that implements a network function.

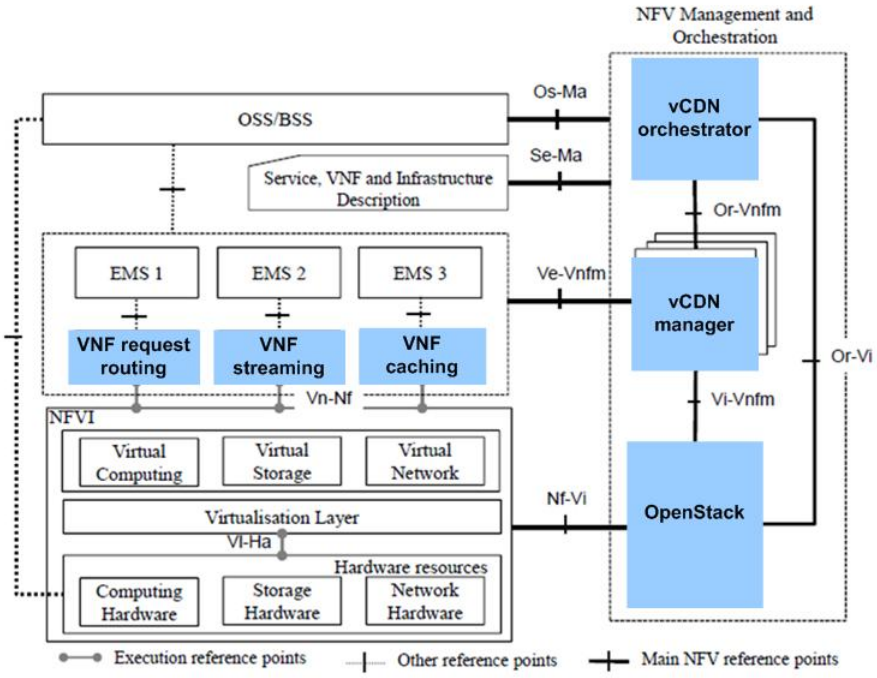


Figure 6-2 – virtual CDN architecture according to ETSI standard

interfaces:

1. *Vertical interfaces:*

   - vCDN orchestrator–|–vCDN manager: It instantiates the vCDN manager and then manages the global CDN network services through resource management rules.
   - vCDN manager–|–OpenStack: It has the role of virtual resource provisioning .
   - OSS/BSS–|–vCDN's VNFs: It guarantees the end-to-end security of the allocated vCDN VNFs and provides the streamed QoS.
   - vCDN's VNFs–|–VMs(NFVI): It assures the instantiation and deployment of the corresponding VNFs of vCDN (i.e., request routing, streaming, and caching).

2. *Horizontal interfaces:*

Table 6.1 – Mapping between simplified vCDN and ETSI-MANO

ETSI-MANO	***Our mapped entity***
NFVO	CDN orchestration module
VNFM	CDN Manager, OCPA
VIM	OpenStack
EM/VNFs	Caching and streaming nodes with a request routing node
NFVI	VMs
OSS/BSS	Not used here

- vCDN orchestrator–|–OSS/BSS: It manages the CDN network services (NSs).
- vCDN manager–|–EM/VNF: VNF life-cycle management including VNFs scaling, VNFs creation, VNFs deleting, VNFs stopping, VNFs releasing.
- OpenStack–|–NFVI/VMs: It has the role of VM resource allocation and provision. It manages VM creation, VM deleting, and VM importing.

In Table 6.1, we map the ETSI-MANO design to propose a practical vCDN architecture. Further, we propose different algorithms for orchestrating vCDN components, placing, and migration of the vCDN caching and streaming nodes.

vCDN must follow the VNF model. However, its specificity resides on the type of the used components, the OCPA orchestration algorithm that should be used, and its layer of integration (vCDN manager). In the next subsection, we define our enhanced vCDN architecture comparing to the traditional ETSI MANO model.

6.2.3 Global virtual CDN architecture

The global vCDN architecture is enriched by several computing, networking and management modules. In this section, we quote the baseline components as follows: NFVI space, VNF space, Orchestration space, and management space.

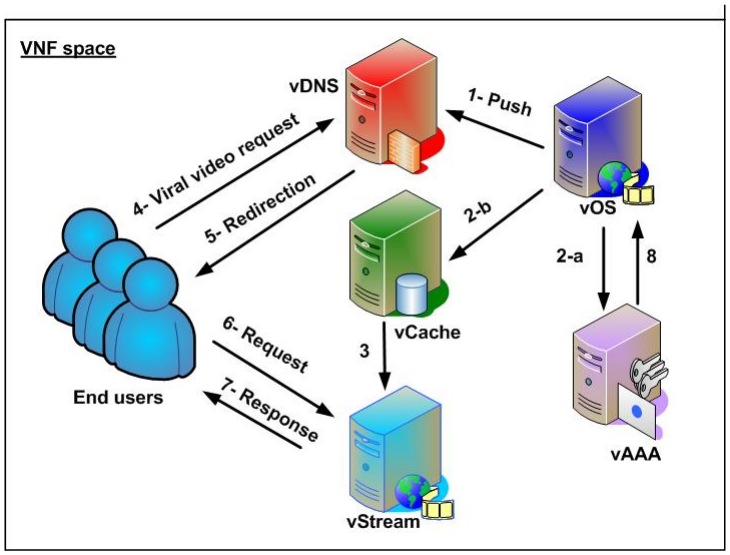


Figure 6-3 – VNF space for virtual CDN

NFVI space

This space includes three domains:

1. Virtual computing domain: The computing domain is the area of virtual machines computation.

2. Virtual storage domain: It is the area of allocating VM states, contexts and files.

3. Virtual networking domain: It is the area of virtual network interfaces (VNIs) configuration.

VNF space

In the VNF space as shown in Fig. 6-3, we propose these virtual functions for vCDN:

1. Request routing: This function has the role of selecting a suitable cache node and redirecting end-user requests to it. It can be:

- DNS instance (vDNS): It is a DNS server software instance running on one virtual machine. This approach of redirection relies on DNS client IP address and not end-user IP address. It cannot be an optimal redirection.
- HTTP instance: It is a HTTP server software instance running on one virtual machine.
- Anycast instance: It is an Anycast server software instance that represents a group of caches nodes by a single IP address.

2. Cache instance (vCache): It is a storage domain, memory zones, or a simple data base server/client that caches the video content temporary. This video is delivered locally to another end user for a future request. As an example for caches: MYSQL.

3. Authentication Authorization Accounting instance (vAAA): It is a security software instance for the cryptography issues. Any foreign user cannot decipher the video content either in the origin server, or in the cache nodes.

4. Streaming server instance (vStream): It is a video streaming server running on a virtual machine. It uses RTP or RTCP protocol for the delivery.

5. Origin server (OS) Instance: the origin server is the entry point to the CDN customer (content provider). The latter ingest its popular video content inside the origin server. We have one origin server per content provider.

In our vCDN design, we propose offering one request routing instance, one AAA instance, one origin server. And, we propose also n cache instances, n streaming servers instances per content provider.

Note here that other VNFs can be added easily to the VNF space and still controlled by the VNFM. For instance, we can quote:

- Dynamic MIP.v4 or PMIP.v6 server instance: In order to enable IP mobility of the virtual machine, we propose to use Dynamic MIP.v4. Virtual machine, server, or session mobility is enabled through this protocol. The use of this

protocol is mandatory for live virtual machine migration between different data-centers. Other tunneling protocols can be used such as VPN, GRE and Cisco LISP mobility.

- SDN controller instance: It is software for controlling video traffic between streamers, load balances, and cache nodes. It can use virtual switches for linking the VMs and enable monitoring and traffic engineering. Floodlight and Opendaylight are examples of SDN controllers.
- Load balance instance: It is software instance for balancing end user request between active streaming servers.
- Advertisement server Instance: It acts a mixer of the origin content and popular or desired advertisement.
- Mobility and prediction instance: It has the role of an entering point to the OPAC instance. It feeds it by a user requests traces and user location traces.

Those VNFs related to vCDN architecture should be interconnected, controlled, and orchestrated by the VNFM.

For the sake of simplicity, Fig. 6-3 depicts the low level design of VNF space for vCDN. First of all, content provider pushes his content in an origin server (OS). Secondly, the latter requests for distributing the content between distributed vCaches. This operation starts after a cryptography processing through AAA server (2-a, 2-b). Thirdly, vCaches push content is vStream virtual nodes (3). Fourthly, user device sends a request to a virtual request routing server (4). Fifthly, vDNS redirects user's device request to a suitable vStream node.

Orchestration space

In the orchestration space, we propose to execute an optimization algorithm as an NFV instance in the centralized entity: vCDN Manager as designed in Fig. 6-2. The vCDN Manager has to orchestrate the aforementioned VNFs already proposed for vCDN network function according to the output of the optimization algorithm.

The proposed OCPA orchestrator instances are optimal orchestration algorithm for vCDN that map vCDN VNFs to physical servers and dynamically migrate vCDN nodes to the optimal point of deployment. In other words, these instances are executed as NFV instances of orchestration on top of a VNFM[2]. The OCPA algorithm provides a service instantiation graph, SIG representing the candidate NFV servers to where a set of vCDN components should be placed. The algorithm's execution is then triggered according to these factors:

- Constraint: It represents system, network, and quality constraints of the orchestration algorithm.
- Service Level Agreement (SLA): This trigger is translated from the upper layer.
- Storage Prediction: The available server storage at each time triggers the execution of the algorithm.
- Bandwidth Prediction: The available server storage at each time triggers the execution of the algorithm.

Recall that the output of the aforementioned algorithms is a mapping graph between the optimal NFV servers and each candidate VNF. This output information can be succeeded by a decision-making process giving place/migrate the VNF or not information.

Management space

In the management space, we propose a centralized entity for managing the CDN. It has three main roles:

- The creation of one or more orchestration spaces.
- The management of virtual machines allocation for each vCDN service or VNF.

[2]It is the software that implements the management function (migration, scaling in/out, etc.) of different VNFs already deployed in the network.

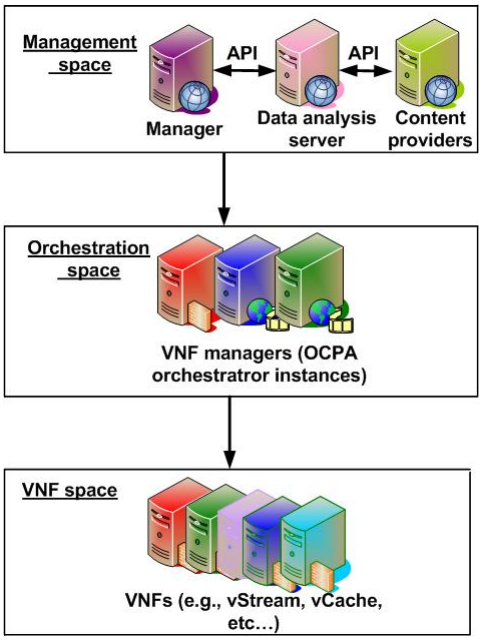


Figure 6-4 – Management, orchestration, and VNF spaces for vCDN

- The resource provision and allocation for the VMs.

In Fig. 6-4, we propose adding a *Manager* entity to the above NFV and VNF spaces. This entity defines the management space. It interacts with the content provider (i.e., the vCDN customer) for content pushing inside the vCDN network provider. The Content provider signs with the manager the SLA, provides the vCDN service time (duration of the service), and the specifies the desired regions that it would cover. Moreover, for adaptive data center activation, end-user demands can be predicted through a data mining server. It communicates between the content provider and the manager entities.

In Table 6.2, we compare the proposed vCDN architecture to the ETSI-MANO reference. It is clear from the table that our design is more flexible as we added an API interface between CPs and NFVI.

Table 6.2 – Comparison between vCDN and ETSI-MANO

Our design	***ETSI-MANO correspondence***
NFVI space	NFVI
VNF space	VNF/EM
Orchestration space	VNFM, NFVO
Management space	No entity, this space is specific for virtual CDN service. It interacts directly with a content provider which is the customer of CDN service.

6.3 OCPA: optimal vCDN orchestration algorithm

In this section, we specify the parameters and the constraints that are defined and proposed in formulating the optimization model, OCPA. This formulation determines the extension of vCDN component to the optimal locations. We quote in Table 6.3 the main system and network parameters, and decision variables. The OCPA optimization model considers three NFV server levels where vCDN components may be extended in the orchestration process: i) the PoP level, ii) the home level, and iii) the user device level.

- *The decision variables*:

  1. The binary variable x_f^s indicates the placement (instantiation) of the vCDN component on the optimal location s. It is defined as:

$$x_f^s = \begin{cases} 1 & \text{if } f \text{ is instantiated on } s \\ 0 & \text{Otherwise} \end{cases} \tag{6.1}$$

  2. The binary variable $y_{v,f}^s$ indicates a customer v needs a vCDN component, and the server s instantiates it. It is defined as :

$$y_{v,f}^s = \begin{cases} 1 & \text{if } v \text{ needs } f \text{ and } s \text{ instantiates } f \\ 0 & \text{Otherwise} \end{cases} \tag{6.2}$$

Table 6.3 – Mathematical Notation

Parameters	***Definition***
V	The set of customer's clients
S	The set of servers ($s \in PoP, H, U$)
D^s	Maximum network capacity of the server $s \in S$
F	The set of vCDN components in terms of vCDN types
f_{size}	vCDN's size ($f \in F$)
C^s	Maximum system capacity of the server s
$L_{i,j}$	Link capacity
d_v^f	The set of customer demands
p_f^s	The placement cost of f on s
Decision variables	***Definition***
x_f^s	Placement binary variable which indicates that the ($f \in F$) should be placed on the (optimal) server $s \in S$
$y_{v,f}^s$	Mapping binary variable which indicates that consumer group ($v \in V$) needs ($f \in F$) placed on the server $s \in S$
$z_{i,j}^{v,f}$	Flow balance binary variable which indicates whether the link (i, j) is used

3. The binary variable $z_{i,j}^{v,f}$ indicates whether a link (i, j) is used (from i to j) to stream from a server s replicating f (the one for which $y_{v,f}^s = 1$) to customer's clients v.

- The constraints proposed in this model are:

1. Storage and bandwidth (S/B) capacity: the end user device level has less S/B capacity than the STB/RAN and the STB/RAN level has less S/B capacity than the PoP/EPC level.

2. The storage of the optimal server should not exceed its maximum size:

$$\forall s \in S : \sum_{f \in F} x_f^s \times f_{size} \leq C^s \tag{6.3}$$

3. The cost of streaming f by a server s should be less than or equal to the

maximum server capacity:

$$\forall s \in S : \sum_{v \in V} \sum_{f \in F} y_{v,f}^{s} \times d_{v}^{f} \leq D^{s} \tag{6.4}$$

4. Each replica server among the distributed NFV enabled infrastructure has also to serve one optimal user device.

$$\forall s \in S : y_{v,f}^{s} \leq x_{f}^{s} \tag{6.5}$$

5. Flow balance or conservation constraint (which is the Kirchhoff laws) between the optimal server s and the customer's client node v should be as follows:

$$\forall i, j \in V \cup S : \sum_{v \in V} \sum_{f \in F} z_{i,j}^{v,f} \times d_{v}^{f} \leq L_{i,j} \tag{6.6}$$

6. Network flow constraint that implies that incoming traffic equals to the out-coming traffic at any network link:

$$\sum_{j} z_{i,j}^{v,f} - \sum_{j} z_{j,i}^{v,f} = \begin{cases} 0 & \text{if } i \neq v, i \neq s \\ y_{v,f}^{s} & \text{if } i = s \\ -1 & \text{if } i = v \end{cases} \tag{6.7}$$

Beside these constraints, we have considered Device-to-device (D2D) communication between end-users devices. Each end-user device acts as a cluster head and feeds a D2D cluster.

$$\forall v \in V, \sum_{f \in F} d_{v}^{f} = D2D_{n} \tag{6.8}$$

Where $D2D_n$ is the number of D2D that requesting a vCDN service through the end-user device $v \in V$. The end-user device may act as a cluster head in a typical D2D-assisted model.

Moreover, NFV/SDN criteria are considered in the sense that each vCDN may serve either Standard Definition (SD), High Definition (HD), or Ultra HD (UHD) video quality:

$$\forall v \in V, f \in F \; d_v^f \in \{SD, HD, 4K\} \tag{6.9}$$

In order to maximize the average QoE of the customer's clients, extending the vCDN components in the orchestration process is important and reduces the end-to end response time and the access delay. Therefore, the objective function has to minimize the response time of all the clients. This is equivalent to minimize the total migration cost. Hence, the proposed objective function is formulated in equation (6.10) (cost function):

$$\min \sum_{s \in X = \{PoP, H, U\}} \sum_{f \in F} x_f^s \times p_f^s \tag{6.10}$$

Where: $(x_f^s)_{|S| \times |F|}$ is the decision binary variable matrix indicating that a vCDNC $f \in |F|$ should be migrated to the server s $\in X$ and p_f^s is a parameter depending on the position of s, the position of s_f (the server initially containing f before orchestration), and the position of s_v (the server initially connecting to the client group v). p_f^s depends also on the size of f and the operator policy.

For the sake of clarity, we propose an example of OCPA vCDN orchestration as shown in Fig. 6-5 . The algorithm is as follows: 1) As an input, the algorithm gathers SDN/NFV architectural information such as the components of vCDN (vCDNc for service caching, vCDNs for service streaming, and vCDNr for request routing) and all necessary dynamic parameters, 2) as a result, end-users requesting vCDN service streaming will be redirected to the optimal edge cloud location.

Recall that the network operator hosting the vCDN content provider orchestrates the vCDN instances (or components) to an optimal service instance graph (SIG) where resources are available and content quality streamed are satisfied.

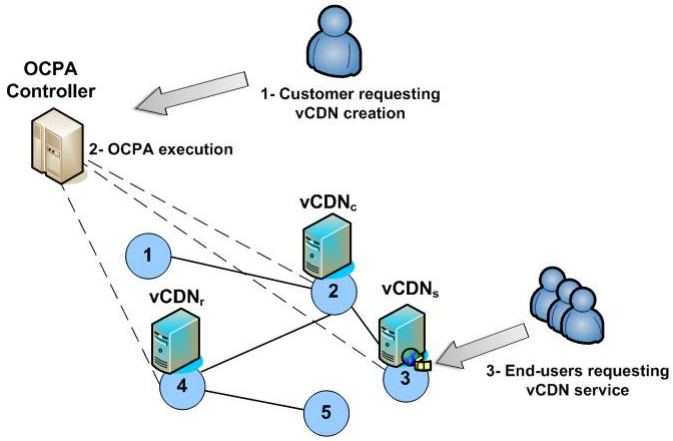


Figure 6-5 – OCPA: vCDN orchestration example

6.4 Performance evaluation

For the interest of assessing the efficiency of OCPA algorithm, we used three-tier as network topology (see Fig. 6-6) suggested by IETF standard for data center [74] and [84] as optimization tools. In addition, different metrics/cost-factors can be defined as follows:

$$vCDN\ caching\ cost = \frac{\sum_{s\in\{PoP,H,U\}}\sum_{f\in F} x_f^s \times f_{size}}{\sum_{s\in\{PoP,H,U\}} C^s} \tag{6.11}$$

$$vCDN\ streaming\ cost = \frac{\sum_{v\in V}\sum_{f\in F} y_{v,f}^s \times d_v^f}{\sum_{s\in\{PoP,H,U\}} D^s} \tag{6.12}$$

$$vCDN\ replica\ number = \sum_{s\in\{PoP,H,U\}\setminus\{s_f\}}\ \sum_{f\in F} x_f^s \tag{6.13}$$

In Fig. 6-7, the vCDN total migration cost is measured in terms of *Gigabits* (*Gb*) against vCDN number (components) and under different deployment scenarios or flavors. Result shows that vCDN-migration cost is increasing in all flavors (vCDNTTP, vCDNTTH, vCDNTTU) for vCDN number ranging from 3 to 10. This is due to the non uniform client group distribution. It is noticeable that vCDNTTP outperforms vCDNTTH and vCDNTTU for vCDN ranging from 6 to 10. Indeed,

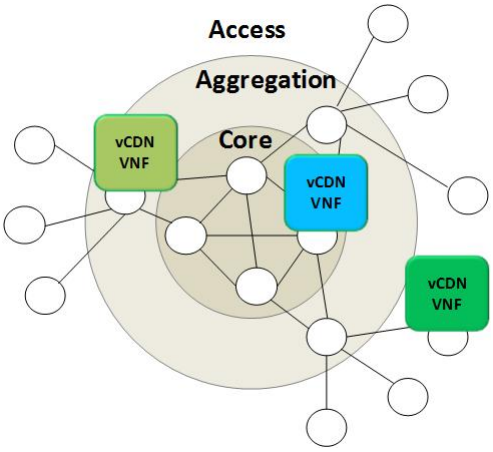


Figure 6-6 – Network topology used for the OCPA evaluation

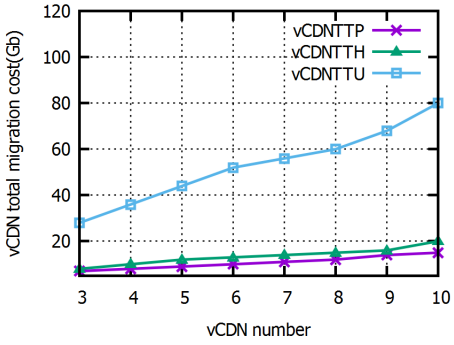


Figure 6-7 – vCDN total migration cost

vCDNTTU requires additional cost and brings more gain to the network operator as the content is extended outside the core network. Recall that each client group ($v \in V$) requests different vCDN components depending on the D2D cluster members.

Fig. 6-8 depicts the total migration time needed to orchestrate vCDN components to the optimal points of deployment taking into account the NFV constraints including system, network, and content quality parameters which are related to the vCDNC functionalities. Although vCDNTTP and vCDNTTH give more shorter time than vCDNTTU, this latter is still acceptable and feasible. This is due to the high required cost required for opening another instance in the user device layer. Recall that the proposed OCPA model focuses mainly on minimizing

the total orchestration cost.

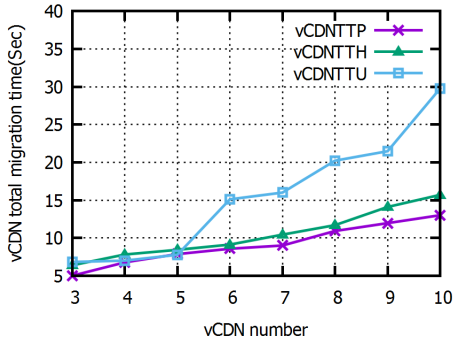


Figure 6-8 – vCDN migration time

OCPA is quantified under vCDN caching and streaming cost factors [3]. In Fig. 6-9a and Fig. 6-9b, cost factors are plotted against vCDN number. The vCDN caching and streaming cost factor increases with vCDN components in all flavors but it still insignificant. Although the orchestration flavor, vCDNTTP is slightly cost-efficient, OCPA algorithm proofed an efficient network resource saving.

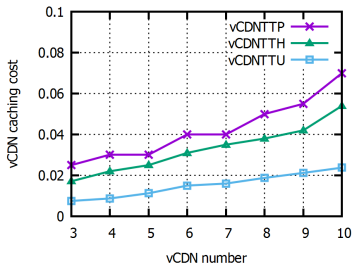
In Fig. 6-10, the total replication number of vCDN is plotted against $|F|$ under a random client matrix demand. Result shows that replica number is not significant. However, vCDNTTP is the strictest approach when replicating vCDN in small scale scenario.

6.5 OCPA: scenarios

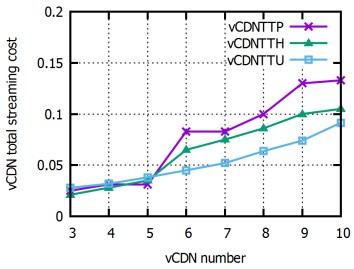
In order to compare OCPA, we proposed two scenarios: *1) User-side* and *2) Network-side* perspectives. The two approaches are defined as follows:

1. In user-side scenario, vCDN components are orchestrated toward the access level represented by home devices (STB, RAN, and and end-user equipment). In other words, network operator outsources vCDNC functionalities. Recall that the network operator is still the manager of its resources despite its outsourcing to the user-side domain.

[3] Cost factor may be defined by the utilization percentage of the available system or network resources. It is the amount of resources (in GBytes) consumed when orchestrating vCDNC to optimal locations.



(a) vCDN caching cost



(b) vCDN streaming

Figure 6-9 – vCDN network optimization costs

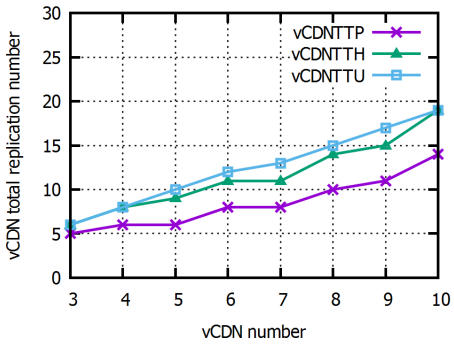
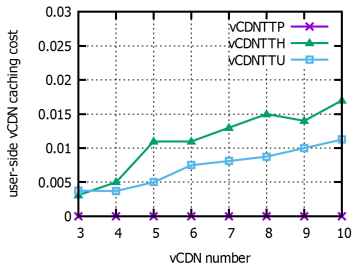


Figure 6-10 – vCDN replica number

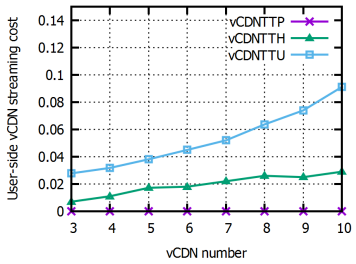
2. In network-side scenario, vCDN components are orchestrated toward the core level represented by the core devices (PoP and data centers).

In Fig. 6-11a and Fig. 6-11b, we depict respectively the average vCDN caching and streaming cost factors. In vCDNTTP use case, where the NFV targets only the PoP server in the core network, vCDN caching and streaming in the user-side is null. However, this metric increases in vCDNTTH and vCDNTTU use cases. It is reduced in vCDNTTU since the algorithm tries to minimize the orchestration processes and instances the vCDN in the access level.

In Fig. 6-12a and Fig. 6-12b, we depict respectively the average vCDN caching and streaming cost factors. Results show that theses metrics are increasing with vCDN number and vCDNTTU flavor has more significant impact on the caching than on the streaming which is quasi-null.

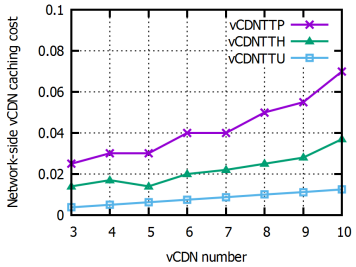


(a) User-side vCDN caching

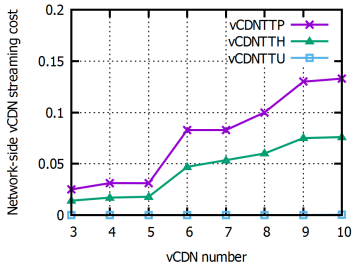


(b) User-side vCDN streaming

Figure 6-11 – vCDN network optimization costs: user side



(a) Network-side vCDN caching



(b) Network-side vCDN streaming

Figure 6-12 – vCDN network optimization costs: network side

6.6 Conclusion

This chapter presents a novel optimization technique for vCDN components orchestration using linear programming technique. We take into consideration novel constraints related to virtualization. Still, all the previous optimization techniques for placement, migration, and orchestration are mono objectives. In NFV delivery context, different actors (content provider, network operator, and client) may cooperate and compete at the same time. Therefore, in the next chapter, we are going to model the vCDN migration process using a novel multi-criteria decision making technique where different deployment solutions are proposed.

Chapter 7

CDNaaS Framework: TOPSIS as multi-criteria decision making for vCDN migration

7.1 Introduction

According to multiplicities of criteria for migration, a vCDN migration process is complex. Further, there are some metrics needed to be maximized such as the QoE of the end-user and other factors and criteria that needed to be minimized as we proved in the previous chapters (i.e., in our vCDN placement, migration, and orchestration algorithms). What we propose here is to use a Multi-Criteria Decision Making (MCDM) method to assure a trade-off between all these metrics in an optimal way.

Moreover, for optimal software resources utilization, vCDN migration needs an intelligent decision-making process for selecting the suitable point of operation where such vCDN components should be migrated. To do this, a multi-criteria decision analysis method (TOPSIS) is proposed in this context as a promising function that enables a flexible selection of the suitable area (core layer, distribution layer or access one) of instantiation of vCDN according to different criteria re-

lated to each vCDN actors (i.e., content provider, network operator, and end-user). Then, a full CDNaaS framework (i.e, enhanced by the multi-criteria decision module) for vCDN management and orchestration is presented.

The rest of this chapter is organized as follows. Section 7.2 describes the adaptation of TOPSIS technique in vCDN migration context. Section 7.3 details the performance evaluation. Section 7.4 introduces our vCDN workflow and the conclusions of the work are highlighted in Section 7.5.

7.2 TOPSIS-based method for vCDN migration

Selecting the optimal place for migration a vCDN component is an important optimization task giving the large network scale and the complex parameters to be considered. In DVD2C project [129], we considered different layers of vCDN request deployment representing the extended position of vCDN: i) vCDN-to-the PoP layer which is representing the operator core data center, ii) vCDN-to-the home network which is representing by the home gateways or Set-Top-Boxes (STB) and iii) vCDN-to-the user devices layer which is represented by the end-user devices. Precisely, our proposal is strongly based on the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method [83]. This technique transforms the context of the end-user (QoE), network operator (cost of migration) into two parts: i) alternatives or options and ii) criteria or attributes.

Algorithm 4 summarizes the pseudo code of TOPSIS based vCDN migration. It is based mainly on two steps:

- TOPSIS-algorithm: In this step, the network parameters are identified in order to carefully choose the decision criteria and the main alternatives.

- Layer selection& Migration-process: In this step, the network administrator selects the suitable layer of deployment. The vCDN nodes dynamically migrate to the optimal NFV servers according to the previous output of the TOPSIS technique. This reduces the number of parameters and constraints

Algorithm 4 TOPSIS algorithm for multi-criteria vCDN migration

1: **Input:** j: criteria, i: alternatives, $x_{i,j}$
2: **Output:**
3: $TOPSIS(criteria, alternatives)$;
4: $Layer-selection()$;

used by the optimization algorithm. Therefore, running time and algorithm complexity could be reduced.

For the sake of clarity, hereafter, we describe these main stages as follows:

7.2.1 TOPSIS formulation

TOPSIS is a multi-criteria decision analysis method. In other words it is a decision-making approach that could be used in different domains. Generally, it is based on two main features: options and criteria. Options are the list of alternatives that can be taken as a decision. Criteria are the metrics that can be considered to make an optimal decision. In vCDN migration context, TOPSIS can be used as follows:

- The potential options that may be proposed are vCDN-to-the-POP (vCDNTTP), vCDN-to-the-home (vCDNTTH) and vCDN-to-the-User device (vCDNTTD) and in case of no migration will be vCDN-at-the-Origin server (vCDNATTO).
- They characterize the content provider, the network operator, and the customer requirements in terms of cost, quality and gain.

To define and enable the TOPSIS method, an option-criteria matrix is defined as follows: $M = (d_{i,j})_{i,j}, i \in \{1, m\}, j \in \{1, n\}$. Where m is the number of options, and n is the number of alternatives or criteria. In the context of vCDN migration, we have ($m = 4$ alternatives) as: (vCDNTTP, vCDNTTH, vCDNTTD, vCDNATTO) and ($n = 8$ criteria) as explained hereafter:

- Content provider has three criteria: They are the cost of pushing content within the network operator, the internal configuration required, and the av-

erage QoE measured and required according to the signed and valid SLA (i.e., content provider (cp) cost, (cp) config, and (cp) quality).

- Network operator has three criteria: They are the cost of migration and optimization, the QoS, and the gain. (i.e., network operator (op) cost, (op) quality, and (op) gain).

- Customer has two criteria: the cost of accessing to the network operator and benefiting of such vCDN service and the satisfaction's index (QoE) in terms of its profile (i.e., end-user (u) cost and (u) quality).

The pseudo-code of the TOPSIS decision-making strategy is summarized in Algorithm 5. Hereafter we explain its main stages for better calcification. After defining the main criteria and alternatives related to the vCDN service migration, TOPSIS formulates the decision making matrix $(d_{ij})_{m*n}$ which should be entered as an input ($Algo.5\ line\ 1$). Note that the network operator should enter dynamically the matrix values related to the importance of the criteria and according to its objective (e.g., minimizing the cost, maximizing the QoE, etc.). Further, giving the decision matrix, we calculate the entropy weight of each criteria (w_j) where; $(j \in \{1, m\})$ according to the Shannon theory that tells the maximum benefit weight of each criteria ($Algo.5\ lines\ 5-7$). This value is needed then to calculate the weighted and normalized decision matrix $(wr_{ij})_{m*n}$ $Algo.5\ lines\ 8-11$). Furthermore, we calculate the Positive Ideal Solution ($PIS = Ideal+$) and the Negative Ideal Solution ($NIS = Ideal-$) in order to choose the optimal criteria related to the alternatives (Algo.5 lines 12 and 13). Basically, the best alternative should be the closest to (PIS) and the farthest from (NIS). This optimal criterion is calculated according to the $L2-distance$. Then, we calculate the ($p-$) and ($p+$) that represent the L2-norm distances from the vCDNTTX position to the worst and best conditions respectively ($Algo.5\ lines\ 14-15$). Finally, for each alternative a similarity function to the worst condition ($Algo.2\ line\ 16$) is calculated and sorted in a list representing the ranked alternatives ($Algo.5\ lines\ 16-17$). Then, the list is given

Algorithm 5 TOPSIS decision list formulation algorithm

Input: A Decision Making Matrix $D = (d_{ij})_{m*n}$
Output: A list $L = (L1, j); j \in \{1, m\}$
$Q = \{m\}$
while $Q \neq \{\}$ **do**
  $\forall s \in \{1, n\}; b_{i,s} \leftarrow \frac{d_{i,s}}{\sum_{j=1}^{j=m} dj,s}$
  $\forall s \in \{1, n\}; k_s \leftarrow \sum_{i=1}^{i=m} b_{i,s} \times \ln b_{i,s}$
  $\forall s \in \{1, n\}; w_s = \frac{k_s}{\sum_{j=1}^{j=n} k_j}$
  $R = (r_{is})_{m \times n}$
  $\forall i \in \{1, m\}; \forall s \in \{1, n\}; r_{is} = \frac{d_{i,s}}{\sqrt{\sum_{j=1}^{j=m} d_{j,s}^2}}$
  $WR = (wr_{is})_{m \times n}$;
  $\forall i \in \{1, m\}; \forall s \in \{1, n\}; (wr_{is})_{m \times n} = r_{m \times n} \times w_{n \times n}$
  $Ideal+ \leftarrow max\ wr_{is} = (wr)^{*+}$
  $Ideal- \leftarrow min\ wr_{is} = (wr)^{*-}$
  $\forall j \in \{1, m\}; p_j^+ \leftarrow dist(wr_{i,s}, (wr)_s^{*+})$
  $\forall j \in \{1, m\}; p_j^- \leftarrow dist(wr_{i,s}, (wr)_s^{*-})$ //the Euclidean distance
  $\forall j \in \{1, m\}; sim_{1,j} \leftarrow p_j^-/(p_j^- + p_j^+)$ //
  $\forall j \in \{1, m\}; L_{1,j} \leftarrow sort(sim)$
end while

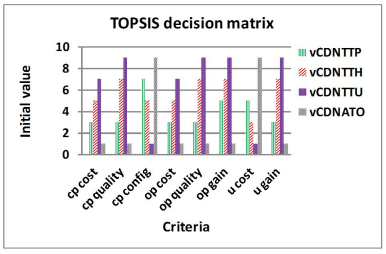
to the network operator as an output in order to select the appropriate layer to where a vCDN should be positioned.

7.2.2 Layer selection

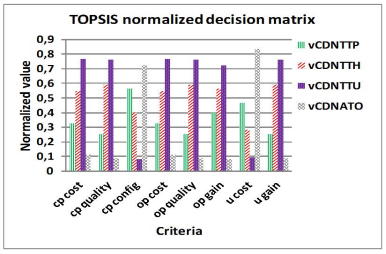
The output of our algorithm is an ordered list of alternatives. The layer selection method outputs a ranked structure list giving a rank to each alternative. Through the proposed context (e.g., vCDN migration), TOPSIS selects the best layer (vCDNTTP, vCDNTTH, vCDNTTU or vCDNATO) in which we can run the optimization algorithm for migrating vCDN nodes. Note here that the run-time needed for selecting the optimal layer is in terms of a few seconds (30 sec as measured in next subsection) which valid the feasibility of the proposed method.

7.2.3 Layer evaluation

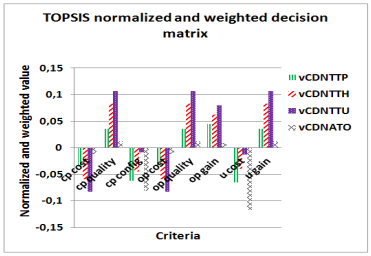
To evaluate our context with TOPSIS, we used the computing environment MATLAB. Fig. 7-1a depicts the decision making matrix. Fig. 7-1b and Fig. 7-



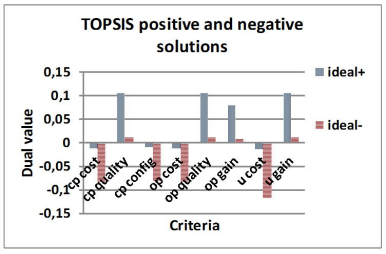
(a) Decision matrix



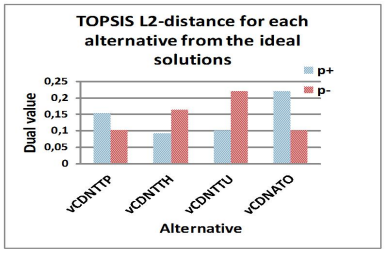
(b) Normalized decision matrix



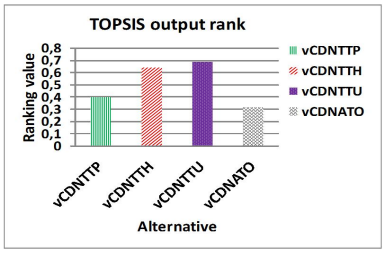
(c) Weighted normalized matrix



(d) Positive/Negative solutions



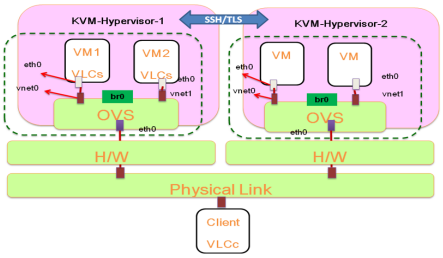
(e) L2-distance for vCDNTTX



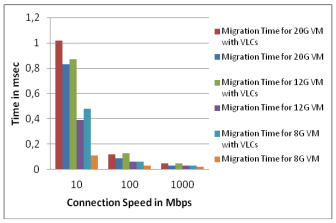
(f) Network layer selection

Figure 7-1 – The TOPSIS sequence for vCDNTTX decision

1c depict the calculation of the normalized and the weighted-normalized decision matrix respectively. Moreover, Fig. 7-1d shows the TOPSIS positive and negative solutions related to vCDN migration criteria and our heterogeneous criteria. Then, Fig. 7-1e plots the distance of each alternative from the ideal solutions (positive and negative). Finally, Fig. 7-1f ranks our proposed alternatives. Experiments show the need for migration (vCDNATO is minimal) and suggest PoP and User Device area to be optimal points of placement and migration of vCDN components.



(a) Testbed architecture



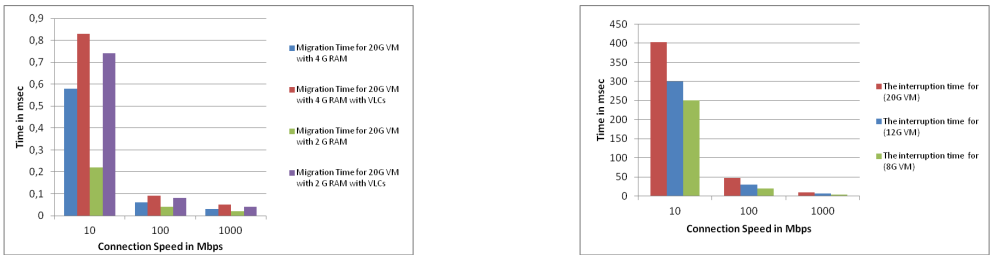
(b) Migration Time for different VM sizes and network connection speeds

Figure 7-2 – Testbed and VM migration times

7.3 Testbed-based performance evaluation

In this part, a performance evaluation for vCDN use case is conducted. Different scenarios of Virtual Machines (VM) representing the vCDN are customized and validated in terms of vCPU, vRAM and vStorage disk. Moreover, different network conditions for interconnections are considered in order to measure the delay time plus the services interruption time in case of live migration for real time services. The platform used for this test consisted of open source virtualization emulator QEMU with Kernel Virtual Machine (KVM). The actual virtualized reference architecture used in our test is shown in Fig. 7-2a. The testbed is conducted using three machines with Ubuntu 14; the first machine represented the server parts of VLC streaming through VM1 and VM2 controlled by the first KVM hypervisor-1. The second hypervisor-2 will represent the new location/PoP to receive the migrated server. The third machine used as VLC client attached to the server during the real time migration to test the connectivity and service interruption times.

In this live migration scenario, we can summarize the following tests as firstly shown in Fig. 7-2b. As long as we have increased the size of the virtual machine, we will need more time to finish the migration. Through these tests, we concluded that the time required for VM migration is different if the size of the machine has changed and if the communication speed is also changed. As a result, the time will increase in the case of live session migration with video streaming although the diffused video has the same size or length.



(a) Migration time for same VM size (20G) with RAM resizing

(b) Service interruption time in case of 4G RAM and different VM sizes and connection speeds

Figure 7-3 – VM migration and interruption times

Moreover, we change the virtual machine RAM to get better performance with minimum migration time. So, the previous test was repeated for same VM size of 20*G* and different RAM sizes (2*G* and 4*G*). For this we obtained the enhanced reduction in migration time as shown in Fig. 7-3a compared to Fig. 7-2b. Finally, the service interruption time (SIT) is measured as shown in Fig. 7-3b. It is cleared that, the network parameters like connection speed has a positive impact on the live migration process as the SIT can be neglected with higher speeds as cleared in Fig. 7-3b with Giga Ethernet connection between the two machines.

7.4 CDNaaS workflow

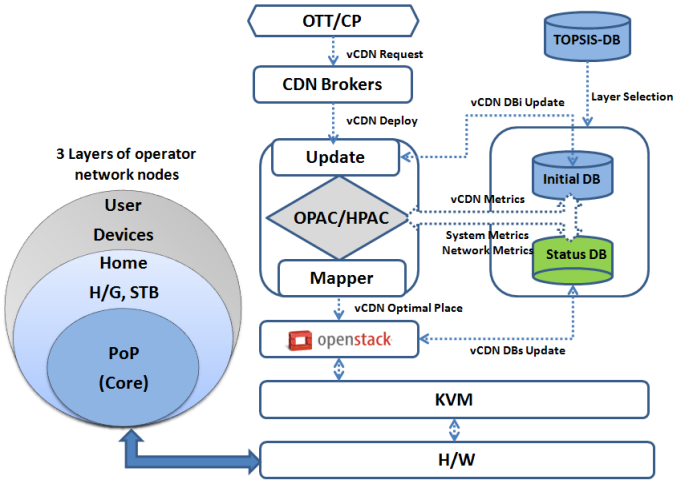


Figure 7-4 – Optimization of CDNaaS workflow for DVD2C project

We present in Fig. 7-4 the workflow that integrates the TOPSIS technique beside optimal and near optimal vCDN migration algorithms [87]. Integrating the proposed methods can follow this diagram with the main stakeholders as:

- Content provider (OTT/CP): The owner of the origin video content and want to create a vCDN network.
- CDN Brokers: It is service orchestrator that interacts with the network Operator of the infrastructure, triggering the deployment of vCDN.
- TOPSIS-DB: a database that includes the ranking values (output of the TOPSIS technique) representing the Layer selection work. Ranks are then entered as an input in the optimization model through inserting the rank values in the initial DB.
- OPAC and HPAC algorithms: They are respectively the implementation of the optimization algorithms in small and large scale network that optimize the vCDN placement and migration based on exact (optimal) and heuristic (near optimal) optimization approaches. The vCDNTTX deployment solutions assisted by TOPSIS method are responsible for deciding whether to extend the migration of vCDN nodes till the Home/User devices or not.
- Databases: mainly two DBs: i) retrieving the vCDN system and network parameters (InitialDB) and then ii) updating a (statusDB) that maps the current location of vCDN (e.g., on which PoP it is running).
- OpenStack: The NFV platform, KVM: The hypervisor and the accelerator of the CPU in the kernel space.
- H/W: The physical infrastructure that supports the virtualization tasks such as cloning, migration, etc.

For the sake of clarifying the sequence vCDNaaS workflow, we propose that firstly different content providers may request for the creation of vCDN services, then

after the deployment and the orchestration of the vCDN nodes, the network operator may execute the optimize command according to the client group demands, therefore, the system fetches the operator databases to get the required information in order to launch in turn the optimize command. Then, the optimization algorithms decide where to place/migrate (OPAC/HPAC) the vCDNs and provide the system/operator the result. Finally, the system executes the migration process according to our results and releases the dedicated resources in case of Service Level Agreement (SLA) expiration/vCDN-delete-request.

To enumerate the main advantages of the proposed virtual CDN, we compare the proposal of (vCDN) to physical CDN (pCDN). Thus, we quote:

- pCDN relied on a pre-fetching process. This process locates video caches in caches nodes before that end user made a request. The process is based on statistics and on prediction modules. It is an off-line optimization. In vCDN, caching process is used but in proactive manner so that it uses mobility and Bandwidth prediction modules. Further, using OPAC placement module in the orchestration space, the optimization is dynamic and on-line. Therefore, vCDN is more flexible and can deal with live events, unexpected high traffic while pCDN is still dependent to the static configuration based on peak hours which is not reliable.

- pCDN uses algorithm with no migration support while in vCDN, benefiting from virtualization, migration is enabled. For instance, server, content, and session migration are three categories for migration.

- pCDN is a hardware based solution while vCDN is virtual/software based solution. Different acceleration tools are used either in kernel space or in user space. For instance, DPKD, KVM, and virtual switches are software packages to do this.

- vCDN can imply multiple paths between OVS switches and nodes. Therefore, it offloads as max as possible the load on the network infrastructure. On

the other hand, pCDN using single path TCP, increases network overhead, doesn't resist failure and not implies any fault tolerance module.

7.5 Conclusion

This chapter proposed a multi-criteria decision making technique called TOPSIS for selecting the optimal deployment layer of vCDN migration problem. Then, the proposed technique is integrated in a CDNaaS workflow for vCDN orchestration with respect to the previous chapters. Still, TOPSIS is an heuristic solution that can deal with large scale systems. Therefore, a multi-objective optimization problem formulation may be proposed in the future work to recommend optimal solutions and assess the efficiency of TOPSIS.

In the next chapter, we are going to study complex active networks and see if traditional optimization constraints or multi-criteria methods are still valid.

Chapter 8

Service placement in complex active networks

8.1 Introduction

This chapter deals with the optimal planning of complex active networks. Complex networks [58] include many data networks used in real life. Social networks and Internet video distribution networks are good examples for complex networks. Active networks [176] are networks that interact with the data content. Originally proposed in a distributed computing context, through Aglets, this paradigm has evolved and got more mature in the Internet Research Task Force. They are envisioned to have a large deployment in the near future through Information-Centric Networking (ICN) [137] to serve video content distribution (caching) [33] [108] and IoT [187]. Complex active networks inherit the challenges of both of these networks which that make their evaluation or simulation/optimization problematic. Applying exact optimization techniques to the planning of these networks is not efficient due to their large sizes and the introduced by dynamic components. Still, we need to have some abstract methods and tools for network optimization and control of such important applications.

Network optimization can be easily modeled with classical tools or heuristic

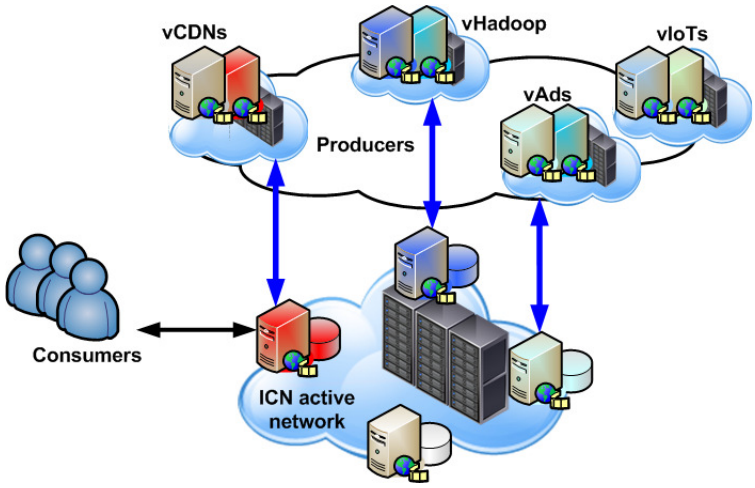


Figure 8-1 – ICN based active network scenarios

methods when the graph becomes too large. Introducing active nodes by means of ICN concept introduces a modification in the graph equivalent to a *negative resistance* in electronic circuits (where active nodes supply an extra capacity/power if the input capacity measured is negative. Indeed, the ICN functions provide some kind of *work* inside the node. This transforms the two dimensional graph to a three dimensional one where the third dimension comes from the interaction between vertical layers within a node. Therefore, the conventional methods that can be used to optimize the graph are not anymore valid and we need to look for a more realistic optimization method that can take the third dimension into account.

Fig. 8-1 depicts distributed ICN active nodes deployed by the network operator with respect to producers (for security delegation issues). The network is acting as a middle-ware of transactions between producers and consumers. Transactions are represented by combinations like questions, responses, and feedbacks. Feedback replaces the network channel in the legacy non-active network. It means receiver-driven model. ICN active network is universal in the sense that it enables different scenarios like virtual IoT (vIOT), virtual Hadoop (vHadoop), virtual CDN (vCDNs), and virtual advertiser (vAds) to exist over ICN. Currently, despite the existence of different scenarios and use cases, optimization tasks are missing from the literature. We therefore, try in this chapter to contribute with modeling the

novel context of introducing active nodes within the distribution/processing network, proposing optimization algorithms for the planning and operation of active networks and explain how they would be intelligent complex virtual LAN.

The proposed model and optimization algorithms in this chapter target general scenarios over ICN complex active network where active nodes add a third dimension to the traditional network topology representing local *work* (e.g., augmented reality, data mining, adaptation, publicity insertion, IoT data timing, etc.). Once the network flow of the ICN active network is defined, optimization algorithms aim to maximize the dynamic data rates under a total ICN budget.

The rest of this chapter is organized as follows: Section 8.2 highlights the network flow algorithms in the graph theory context through the related work. Then, Section 8.3 states the background in this field. Section 8.4 introduces the problem statement and the main contributions. Section 8.5 proposes the optimal placement algorithm (OPPA). Section 8.6 proposes the near optimal solution. Section 8.7 evaluates the behavior of the proposed algorithms. The chapter is concluded in 8.8.

8.2 Related work

The main related work on cut-tree (Gomory-Hu[1]) algorithms are quoted in this section.

Ford Fulkerson [62] solves the problem of the maximum-flow between specific s and t nodes in the graph G and introduced (and proofed) the theorem of maximum-flow minimum-cut that indicates that the (s,t) maximum-flow (the maximum amount of flow that can be transmitted from s to t) is equal to the (s,t) minimum-cut (the minimum total weight of the edges that can separate (disconnect) s from t in G.

An Extension to the Ford Fulkerson method that solves the problem of the maximum-flow between a specific (s,t) vertices in the graph is the multi-terminal

[1] G-H in the rest of the chapter

maximum flow problem that focuses on computing the maximum flow between all network nodes. It is originally studied by Gomory and Hu [80] who propose a cut tree that concisely expresses all minimum cuts (maximum flows) of a given graph. The authors demonstrate that there are only (n-1) non crossing edges representing this computation instead of $\frac{n\times(n-1)}{2}$ where n represents the set of network nodes. Hence, the algorithm complexity is (n-1) min-cut computations. The baseline of the G-H idea is the concatenation process used for avoiding crossing and compute the $s-t$ min-cut values.

Dan Gusfield [76] proposes a simplification to the G-H algorithm by adding at least a three lines of code. The algorithm does not need to ensure (maintain) $(n-1)$ non-crossing edges for constructing the cut tree and hence the algorithm complexity is reduced. Despite that, network target of the aforementioned algorithm is a communication network [80] where main links load is uncertain and unknown in such cases, edge capacity is fixed and entered as an input beside the original graph in all the above methods and algorithms. Then, network modification and especially in the link capacity should be taken into account in the network flow algorithms.

Authors in [72] introduce an experimental study of the main cut tree algorithms described above (G-H and Gusfield). Then, in the spirit of Gomory Hu theorem, they proposed a fast implementation of the G-H tree comparing to the Gusfield method where the original Gomory Hu algorithm complexity is reduced. This fast implementation is due to the combination between the original G-H with some heuristics. In fact, authors through their experimental study try to incorporate some heuristics in the traditional implementation of G-H (not in the simplified G-H algorithm of Gusfield). Heuristic methods are implemented for picking the next (s,t) source-sink pairs in the original G-H algorithm where these vertices are chosen randomly.

T. Hartmann et al. [81] highlight cut-tree algorithms among the main network reports that help for network analysis and detection. They implement a fast and

simple algorithm for dynamic construction of a G-H tree in response to network modifications such as vertex removal, vertex insertion, link cost increase, and link cost decrease. They claim that in the case of edge capacity change, the algorithm saves min-cut computations (there is no need to re-compute the min-cut of a specific sub-graph). Nevertheless, it is still not clear how the algorithm avoids the reconstruction of the G-H tree. Is a G-H needed for the remaining sub-graph ? if yes, what is the benefit of saving min-cut computation. Otherwise, how to compute the min-cuts of the remaining (s,t) nodes ? Further, from an algorithmic perspective, authors do not provide the algorithm complexity which is needed to assess the algorithm feasibility. Furthermore, we believe that the conditions defined for the re-usability of min-cuts in response to network modification can not help in a distribution network where any intermediate node is equipped by a budget regulator for assuring the balance between demands, services, performance, and budget.

In [9], authors deal with the problem of all parametric min-cuts analysis in a communication network. Indeed, they consider the problem of finding the minimum cut of all the node pairs in a capacitated undirected network. They take into account the possibility of edge modification and propose an efficient algorithm for this parametric min-cut problem in a polynomial time. Exact formulation of the maximum flow problem is missing in their work. Moreover, the communication network is supposed to be a lossy network only where intermediate nodes (between the source and sink) can not enhance the throughout traffic.

In [80], instead of using the traditional algorithms for construction the cut tree, authors use a Steiner edge connectivity through a tree packing algorithm. Authors present an alternative implementation of the G-H algorithm based on concatenation baseline to compute minimum Steiner-cut. Despite that, the algorithm capacity is reduced as much as possible and reaches $O(m \times n)$, when edge capacities are equal to 1 which restricts the utilization of the algorithms in communication or distribution networks where network capacity is important and determine its performance.

T. Akiba et al [5] raise another problem related to the construction of cut trees and propose an efficient G-H construction algorithm for today's large-scale graphs. However, authors do not consider network load modification (remaining arc capacity) in their novel method which represents a main limitation of the work.

8.3 Background

Several algorithms have been proposed to optimize the planning and operation of complex networks in polynomial time. We proposed to use G-H algorithm to efficiently find the minimum cut in large video distribution networks that correspond to carrier grade service infrastructures. The proposed algorithm (HPAC in the chapter) is used in two steps. First, we find the G-H tree transformation of the initial graph and then we explore the unique shortest path from an access point to the server containing the vCDN. Thus, when the demands cannot be anymore served because of the bandwidth restrictions in the resulting tree, we migrate the vCDN server to the first node in the tree that does not create any conflict on the tree nodes and links.

Given the importance of the G-H transformation in network analysis, and believing that the network has to consider different intrinsic parameters that follow its behavior (network of distribution, processing, on demand services, etc.) this G-H has to take into account all these parameters to build an efficient network optimization algorithm. In the next two sections, novel network models are highlighted and detailed through a network's use case.

8.3.1 Complex active networks

Complex Active Networks (CANs) are novel concepts to networking architecture that allow intermediate nodes to process (enhance/downgrade) on the fly the real-time traffic flowing through them using external bandwidth amplifiers or power indicator tools.

CANs can be modeled through three main classes: i) Erdos-Renyi-based random models that target geometric graph and follow a linear degree distribution ($P(k) = \frac{k}{n}$), ii) Watts-Strogatz-based small world models that target large network size and follow a binomial (or a Poisson) degree distribution and ii) Barbasi-Albert-based scale-free models that target also large network size (suitable for distribution network) and follows a power degree distribution ($p(k) = k^{-\lambda}, 2 < \lambda < 3$).

To date, these models still represent the baseline tools for network graph generation that target complex active networks. Moreover, CANs pose different problems and introduce serious challenges such as: link capacity prediction, network partitioning, and so on.

Novel use cases of CANs have been highlighted. We quote Information-Centric Networking (ICN) [94], Hadoop, and Internet of Things (IoT) [187] as examples. Hence, a new look at the networking model must follow these novel networking architectures for such potential matching between network architecture, model, and optimization.

ICN, acting as a network compiler/interpreter (of interests and data), adds work to each intermediate node between the producer and the consumer. It can add or consume bandwidth. Therefore, we next give an overview of ICN (as a use case) and the features it brings to the networking layer.

8.3.2 Information-Centric Network

ICN names the content rather than the host in the networking level. Different ICN architectures are proposed such as: network of information (NetInf) [126], content centric networking (CCN) [32], and data oriented network (DONA). Most of these information-centric network architectures are implemented on top of TCP/UDP/IP/P2P layer. All of them are inspired from by the Google talk of van Jacobson [93] who introduced the baselines and the fundamental features of the CCN architecture (node model, naming, routing, transport, caching, etc.) and the strategy layer for the adaptive forwarding.

Hence, CCN represents the baseline ICN architecture as a new look at networking for reshaping messages based on their names (name-based routing) pioneered by Van Jacobson in PARC [94]. In CCN, names are hierarchical and similar to URLs. Name resolution system (NRS) and data routing procedures are either integrated or coupled. The exchanged messages between consumers and producers are in the form of interest (question) and data (response). Indeed, end-users express only *what* they want (content name), and they let the CCN network respond to the *where* and *how* the content will be retrieved implicitly. Therefore, an appropriate CCN topology planning and dimensioning is an optimization per se. In fact, CCN network consists of consumers that request the content, producers that publish this content, and ICN routers that cache and ask for the content on behalf of the consumers. The content router (CR) of ICN has three main data structures as follows: i) Forwarding Information Base (FIB) table: it binds the content name to next hop as in IP layer that binds the IP prefix to the destination, ii) Pending Interest Table (PIT): it binds the content name of the unsatisfied requests to the requesting face, and iii) Content Store (CS) table: it binds the content name to the data per se.

Caching in CCN implies the on-path caching. CSs use by default the LRU replacement policy. Off-path is also supported by redirecting user interest to a CDN (as an example) and not to the source/publisher of the content. Mobility is handled by $Kite$ model as mentioned in UCLA work [189]. Security is assured through binding name to the content by crypto signature field in a CCN data packet.

CCN brings many advantages. Indeed it assures an efficient content distribution. Further it implies cache consistency using unique named data object (NDO) and resolving by this way the problem of content moving (the content is independent from the storage and the location). It breaks the End-to-End security model and secures the content rather than the host through data integrity using hash function and Origin verification.It facilitates mobility, multi-homing (multi network access) by just re-issuing requests for NDOs. Finally, it leverages hop-by-

hop transport model as in Delay Tolerant Network (DTN) scenarios.

8.4 Problem statement and contributions

Negative resistance situation: We face this situation when the consumer demands going through the resulting G-H tree cannot be satisfied anymore. We demonstrated through a virtual Infrastructure Optimization Simulator (vIOS)[2], implemented for deploying different vCDN/ICN functions, that a negative situation is occurred and there is a need to migrate a group of delivery functions to optimal data center locations. More precisely, given the high migration cost in CANs, our research problem is how to model the network infrastructure after adding active nodes in a complex graph and how to optimize the network planning and service placement in presence of these active nodes.

Our modification to G-H considers modeling the network infrastructure in the presence of active nodes that act as if they provide additional (or consuming) bandwidth in the transport network (e.g., by modifying the throughout traffic on the fly, adding advertisement, executing binary codes for computation, annotated video, augmented reality, insertion on the fly, etc.). Users requesting a critical service (e.g., public safety), or a real time service should be satisfied in an optimal time according to the application using the active networking facilities. The research objective is to leverage ICN caching and processing capabilities to propose scalable, optimal, and adaptive service placement algorithms of ICN nodes in CANs.

8.4.1 Active node budget

The dynamic nature of intermediary nodes, that can provide extra work during a session has to appear in the optimization process. We propose a budget in each node (less or greater than the unit). It can be passed onto the node links as an extra capacity. The introduced budget γ represents a flow multiplicity as follows:

[2]https://github.com/TelecomSudparis-RST/vIOS.

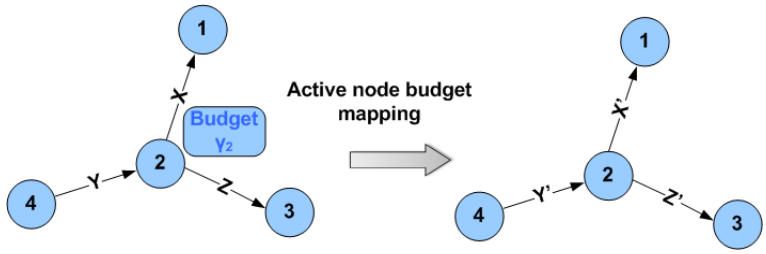


Figure 8-2 – Active node budget mapping

$\forall a \in A, j \in \Gamma^+(a) : F'(a,j) = \gamma \times F(a,j)$ Where A is the set of active nodes, F and F' are the total flow on an edge (a,j), and $\Gamma^+(a)$ is the out-degree of the active node a.

In Fig. 8-2, we give an example of active node budget mapping where a budget γ_2 of the server 2 is passed onto its outgoing links. The outgoing link weights of the server (X and Z) representing the residual flow are amplified by γ_2.

Let us Know generalize the budget formulation. Suppose that F^+ and F^- are the outgoing and incoming traffic from an active node $a \in A$. The traffic can be modeled as follows: $\forall a \in A, l \in \Gamma_a^+ : F_a^+ = \sum_l F(a,l)$ and $\forall a \in A, l \in \Gamma_a^-; F_a^- = \sum_l F(l,a)$ where Γ^+ and Γ^- are the set of the in-degree and the out-degree of the active node a respectively.

Due to the work and effort that an ICN may introduce in the network there is no more flow balance at an active node. Hence, the adjusted Kirchhoff (flow balance) equation at each active node will be: $\forall a \in A : F_a^+ = \gamma \times F_a^-$Where γ is a real number ((> 1) for flow augmentation and (< 1) for flow diminution) that represents the dynamic work inside an active node. This equation will also introduce a non conservation of all the network traffic (i.e., the sum of all outgoing traffic will be the sum of all incoming traffic multiplied by the budgetγ).

8.5 OPPA: Optimal Practical Placement Algorithm for ICN

We propose OPPA, an exact optimization algorithm that takes as an input the topology of the underlying network. It aims then to optimally upgrade some nodes by deploying ICN software (placement problem). The network flow problem dif-

fers from the classical theory in order to model complex active networks where there is no more flow conservation at each node. The flow model is then used to solve a placement problem constrained by topology related parameters and budget parameters that determines the network operator capacity. The adjusted network flow model between each source-destination (s,t) (from s to t) node pairs is:

$$\sum_{j\in\Gamma^+(i)} F_{i,j} - \sum_{j\in\Gamma^-(i)} \lambda_i F_{j,i} = \begin{cases} 0 & \text{if } i \neq s, i \neq t \\ A_s & \text{if } i = s \\ -B_t & \text{if } i = t \end{cases}$$

where A_s and B_t represent the quantity of flow streamed by the source s and received by the destination t. Non flow conservation implies that ($A_s \geq B_t$) for flow amplification and ($A_s \geq B_t$) for flow degradation.

8.5.1 ICN budget model

We formulate an Integer Linear Programming (ILP) to optimally model the ICN active network, and is shown below:

$$\max \sum_{s\in S, t\in T} F_{s,t} \tag{8.1}$$

Subject to:

$$\sum_{j\in\Gamma^+(i)} F_{i,j} - \sum_{j\in\Gamma^-(i)} \lambda_i F_{j,i} = 0 \tag{8.2}$$

$$\sum_{j\in\Gamma^+(s)} F_{s,j} - \sum_{j\in\Gamma^-(s)} \lambda_s F_{j,s} = F_{s,t} \tag{8.3}$$

$$\forall i, j \in S : F_{i,j} \leq \lambda_i \times C_{i,j} \tag{8.4}$$

$$\forall i \in S : \lambda_i \leq B_i \tag{8.5}$$

$$\sum_{i \in S} \lambda_i \leq B \tag{8.6}$$

$$\forall i \in S : \lambda_i \geq 1 \tag{8.7}$$

Description of the model: The model considers the scenario of ICN active network where intermediate nodes participate in the content distribution process. The formulation maximizes the maximum flow between each pair of source-destination nodes ($F_{s,t}$) (Eq. (8.1)) under Kirchhoff law constraint on intermediate nodes and at the source node respectively (Eq. (8.2) and (8.3)). Then, an amended bandwidth constraint as defined in Eq. (8.4) where network flow is amplified by λ. The inserted amplification is positive and is constrained by the maximum node budget (Eq. (8.5) and Eq. (8.7)). Network operator also can not invest more than its total budge (Eq. (8.6)). It ensures the bandwidth amplification, capacity constraint, budget constraint, non-negativity budget, etc.

8.5.2 OPPA

We propose to enable only ICN nodes to modify the throughout traffic between all (s,t) node pairs. To do this we use the following flow balance constraint that determines an optimal placement of ICN software that enhances the streamed traffic between the producer p and the consumer c on the fly:

The OPPA based on the ICN's budget model is listed below:

$$\min \sum_{s \in S} \sum_{f \in F} x_f^s \times t_f^s \tag{8.8}$$

subject to:

The equations (8.5), (8.6), and (8.7)

$$\sum_{j \in \Gamma^+(i)} z_{i,j}^{v,f} - \sum_{j \in \Gamma^-(i)} \lambda_i z_{j,i}^{v,f} = \begin{cases} x_f^i & \text{if } i \neq v \\ -1 & \text{if } i = v \end{cases} \tag{8.9}$$

Table 8.1 – Mathematical Notation

Parameters	***Definition***
V	The set of client group nodes
S	The set of server nodes
F	The set of virtual nodes (e.g., vCDN nodes)
D^s	Maximum throughput of the streaming server $s \in S$
C^s	Maximum memory capacity of the server s
f_{size}	virtual node's size $(vRAM, vCPU, vDISK)$ $(f \in F)$
$L_{i,j}$	Link capacity between two nodes i and j (from i to j)
d_v^f	The set of consumer demands in terms of interests
λ_s	A budget parameter that indicates the in-network processing in node s
B	The total network budget for ICN node upgrade in terms of money
B_i	The budget of node i in terms of money
t_f^s	The treatment cost of the functionality f to s
Decision variables	***Definition***
x_f^s	binary variable which indicates that node s should be upgraded by ICN software to store and treat data.
$y_{v,f}^s$	Binary variable which indicates the video hit from node v of f in server s
$z_{i,j}^{v,f}$	Binary variable indicating whether the link (i,j) is used to stream f to v

$$\forall s \in S : y_{v,f}^s \leq x_f^s \tag{8.10}$$

$$\forall v \in V \mid d_v^f \neq 0 : \sum_{s \in S} y_{v,f}^s = 1 \tag{8.11}$$

$$\forall s \in S : \sum_{v \in V} \sum_{f \in F} y_{v,f}^s \times d_v^f \leq D^s \tag{8.12}$$

$$\forall s \in S : \sum_{f \in F} x_f^s \times f_{size} \leq C^s \tag{8.13}$$

$$\forall i, j \in V \cup S : \sum_{v \in V} \sum_{f \in F} z_{i,j}^{v,f} \times d_v^f \leq L_{i,j} \tag{8.14}$$

$$\forall i, j \in V \cup S : z_{i,j}^{v,f} \leq \lambda_i \times L_{i,j} \tag{8.15}$$

The above formulation of the network problem is optimal. However, it is an NP-hard problem. Therefore, an extended cut tree algorithm is formulated (EGHT) and used as input in our proposed HPPA algorithm (heuristic) that solves the *negative resistance* problem defined above.

8.6 HPPA: Heuristic and Practical Placement Algorithm for ICN scenario

Our proposal is based on the extended G-H of the initial network (represented by access, aggregate and core nodes). In other words, HPPA transforms the input network into a G-H tree. Then, upgrading some nodes by ICN software is performed thanks to our extended G-H tree allowing to efficiently reduce the number of edges to be considered when upgrading nodes, placing contents (in-network caching feature of ICN), and end users redirection to the optimal ICN (on-path caching).

Algorithm 6 summarizes the pseudo code of HPPA. Hereafter, we describe these main stages.

HPPA is the algorithm used for *negative resistance* networks (e.g., face to unexpected increase in consumer demands) and hereafter we explain how it works.

Algorithm 6 HPPA: Heuristic Practical Placement Algorithm for ICN

Initialization()
$EGHT \leftarrow$ Extended Gomory-Hu tree algorithm ()
Consume-demands ()
Construct-consumed-EGHT(capacity, budget)
Ranking-node()
if $W_{s,t}^{min} < 0$ **then**
    $\lambda_s \leftarrow W_{s,t}^{min}$
    Enhance-Amplify: $b_s = W_{s,t}^{min}$
    $W_{s,t}^{max} \leftarrow W_{s,t}^{max} - W_{s,t}^{min}$
end if
if $W_{s,t}^{min} > 0$ **then**
    $\lambda_s \leftarrow -W_{s,t}^{min}$
    Downgrade: $b_s = -W_{s,t}^{min}$
    $W_{s,t}^{max} \leftarrow W_{s,t}^{max} + W_{s,t}^{min}$
end if
Return-ICN-Maximum-Flows()
$\forall d \in d_{client}^{service} do$
Enable-ICN-in-network caching()
Update-service-placement()

Algorithm 7 EGHT: Extended Gomory-Hu Tree Algorithm

Input: A connected graph $G = (V(G), E(G), capacity, budget)$
Output: An Extended Gomory Hu Tree $EGHT = (V(GHT), E(GHT))$
$V(GHT) = V(G)$, $E(GHT) = \emptyset$
Initialize the tree: every edge points to node 0
For each source vertex except vertex zero, Find its neighbor
Find the minimum cut between s and t with the installed capacity
Find the minimum cut between s and t with the amplified capacity.
Update the tree
Construct the tree

8.6.1 EGHT: extended Gomory-Hu tree algorithm

Our extension to the traditional Gomory Hu tree is to construct two cut-trees given the initial topology graph. Our implementation of EGHT is inspired from the Gusfield simplification of G-H tree where there is no need to maintain non-crossing edges in computing all the (s, t) min-cuts.

This algorithm is optimal in the sens that is provides the maximum flow computation for all node pairs. It is entered as an input to any transportation problem related to CANs to resolve placement issue, network load reduction, and e-e QoS/QoE and so on. Hereafter, we propose an utilization of this EGHT to resolve

three main issues related to the ICN-based CANs:

- How to model the network infrastructure after (before it is already exist through exact/heuristic approaches) adding the ICN active nodes in a complex graph where Kirchhoff's law must be adjusted and other constraints should be updated.
- How to optimize the network dimensioning in presence of ICN nodes ?
- where to locate the streaming headend ?
- How to route the streams to the users ?
- Which budget should be placed on which nodes ?
- How to deal with dynamic operation: For instance, in case the client group membership changes, e.g., users join or leave ?, 1) upgrade the ICN node, 2) redirect the users to ICN node instead of requesting the source, 3) determine the ICN gain.

The EGHT's algorithm complexity is $2 \times (N - 1)$ minimum-cut computations (i.e., maximum flow computations). Hereafter, we describe our solution over an ICN scenario.

8.6.2 HPPA downgrading/enhancement on the fly

Firstly, HPPA requires an initialization step. In this step, a topology graph is built from the information about the network nodes (their physical locations) and the network links representing the relation ship between these nodes. Then streaming sources (e.g., ICN/IoT containers, hadoop data nodes, vCDN nodes, etc.) representing the *producers* and end-users representing the *consumers* are initially placed on this graph.

Secondly, EGHT is executed. This step outputs two cut trees from the topology graph used for network analysis and then helps to determine the potential upgrade of ICN/CCN nodes. For each edge in EGHT, two maximum flows are computed

representing the maximum flow value ($W_{s,t}^{min}$) and its potential amplification $W_{s,t}^{max}$ receptively.

Thirdly, consume demands method is invoked. In this step, all the consumer demands (e.g., ICN interests) are consumed by the two trees and consume the network capacity (of course system capacity is consumed also).

Then, we construct a consumed EGHT. At this step, a consumed EGHT is constructed and represented by two consumed G-H trees that has the remaining network capacity after that all the consumer demands have been satisfied. Further, a ranking-edges is performed. In this step, we rank the edge weights from increasing.

For each consumer demand, a shortest-path is found from the *consumer*'s location to the *provider* on the original graph. The network links in this shortest-path have their capacity consumed to satisfy the demand bandwidth. After that all the demands have been analyzed, some of the network links will have no remaining link capacity (i.e.; ($W_{s,t}^{min} = 0$), some other links will have negative remaining link capacity (i.e.; ($W_{s,t}^{min} < 0$), and other links will still have. Otherwise, if all links have remaining capacity, it means that the network can easily serve all the consumer demands and there is no need to downgrade and upgrade some nodes with CCN software.

Furthermore, *enhance-Amplify* process is triggered. This step is occurred When negative link is detected. It upgrades some server nodes with ICN software (e.g., CCNx) that amplify the maximum outgoing flows (from s to t). Sometimes, a *downgrade* process is needed. This step is occurred When positive link is detected. It means consume a set of the incoming flows to save network bandwidth.

The Return-ICN-Maximum-Flows method returns the maximum flow between all the node pairs in presence of ICN active nodes. Then, we enable the in-network feature of ICN which means the capability of a router to cache a requested content. Next end users are redirected to this ICN active node.

In-network caching: The decision to amplify the bandwidth and cache the ICN

data takes the following logic: First of all, every consumer demand is re-analyzed to verify if it traverses a saturated (s,t) link (negative link capacity). If it does, then a bandwidth amplification is needed. If there is an intermediate node in the demand path that can hold the requested ICN data (using the needed capacity values set for the ICN data and the current available resources and if the intermediate node has the network/system capacity to hold the streaming source (by calculating the difference between the installed bandwidth and the currently used bandwidth), then an amplification is occurred and its cost (hosting cost) is calculated.

However, if the ICN caching would take place between the *producer* and the *consumer*, all the other consumer demands requesting the same ICN data from the original *producer* would have to be satisfied (served) from the new ICN node (in-network caching feature of ICN) in the case of off-path caching, otherwise, the original producer will serve the requested data.

Update-service-placement: The algorithm is dynamic in the sens that it updates the database of each ICN node (i.e., its content store). Novel demands benefit from the in network caching and increase the bandwidth gain. Indeed, ICN nodes become the *producer* of similar interests.

We propose the following example (see Fig. 8-3) to illustrate ICN algorithm for treating judiciously ICN streamed data. It depicts a scenario of ICN insertion. In this scenario, consumers representing aggregated interests for a media data stored in media source repositories with a specific throughput (i.e., content quality) equal to 40 $Mbps$. The proposed ICN method searches to adapt the bandwidth capacity along the shortest path from consumers to publishers. Active ICN software are inserted in nodes 1 and 2 to enhance network capacity.

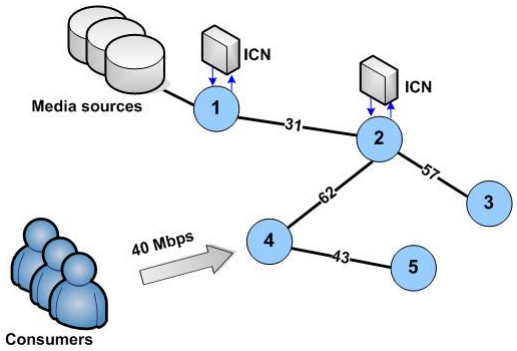


Figure 8-3 – G-H based ICN insertion

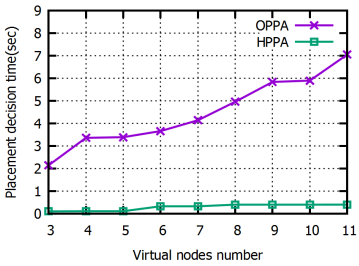


Figure 8-4 – OPPA-HPPA decision time

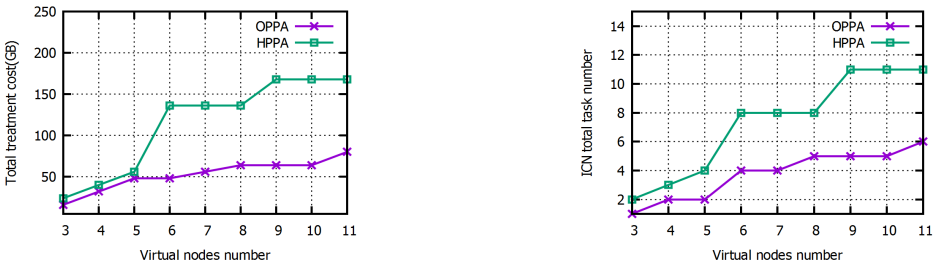
8.7 OPPA vs HPPA: performance evaluation

8.7.1 Small network scale

In order to assess OPPA and HPPA algorithms in a small scale scenario, we used runtime, treatment cost and parallel task number as our performance key metrics. Fig. 8-4 depicts the runtime which is the execution time of the algorithms. Fig. 8-5a shows the treatment cost of the ICN data in terms of resource size (storage). The parallel task number is the total number of the treatment task (how many times we need to treat the requested data). It is plotted in Fig 8-5b.

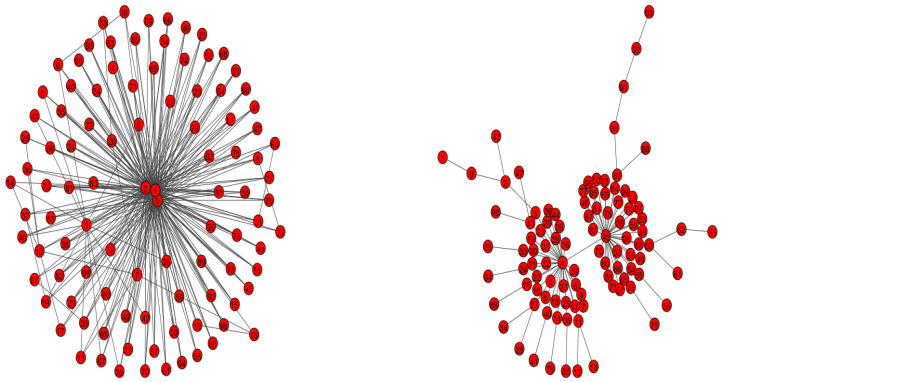
8.7.2 Large scale scenario: a Barabási–Albert based network operator

In large scale network, we evaluated our HPPA algorithm using scale free topology. According to the IETF, Barbasi-Albert-based scale-free network may be a suitable solution for evaluation ICN in large scale [138]. The scenario relies on the



(a) ICN total treatment cost (b) ICN total parallel task

Figure 8-5 – OPPA-HPPA comparison in the small network scale scenario



(a) Scale free network (b) G-H-based transformation

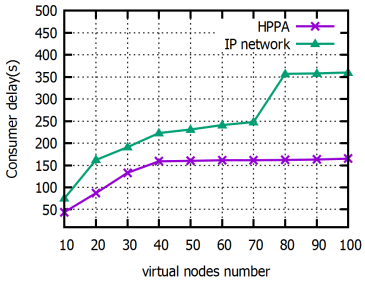
Figure 8-6 – Network topology used for large scale

well known Barabási–Albert undirected and weighted graph. The graph has 100 vertices (nodes) and 200 edges as shown in Fig. 8-6a. Its G-H-based transformation is shown in Fig. 8-6b which has only 99 edges (49.5%). The OPPA algorithm could not be used here with reasonable resources.

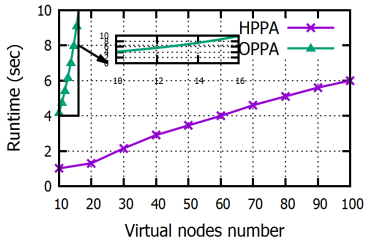
Fig. 8-7 depicts the results of our algorithm in large network scale. Experimentations suggest that the optimization algorithm is feasible (Fig. 8-7b), reduce consumer response time (Fig. 8-7a) with bounded network costs (Figures 8-7c and 8-7d).

8.7.3 OPPA vs HPPA comparison

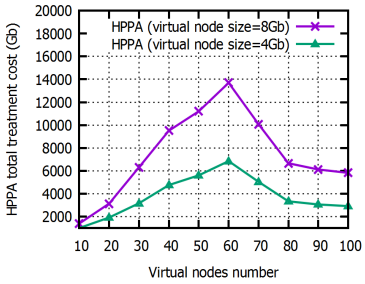
A brief comparison between the two approaches is given in the Table 8.2 .



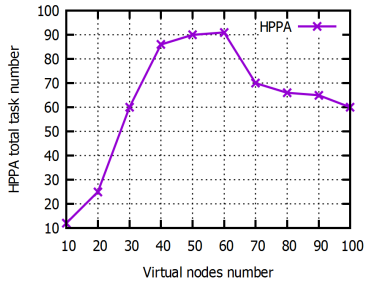
(a) Total consumer delay



(b) Runtime



(c) ICN total treatment cost



(d) ICN total parallel task

Figure 8-7 – HPPA in large network scale scenario

Table 8.2 – Efficiency comparison between OPPA and HPPA

Metric	Small scale		Large scale	
	OPPA	HPPA	OPPA	HPPA
Run-time(sec)	< 7	< 1	unfeasible	< 7
total treatment cost(Gb)	*low*	*high*	unfeasible	bounded(< 14)
total task number	*stric*	*loose*	unfeasible	bounded(< 90)

8.8 Conclusion

This chapter presents a novel use case of complex active network based on ICN approach. Intermediate nodes are active and treat the data on the fly. Network optimization is modeled and quantified. Results are encouraging and proof that consumer delay is reduced and the network performance is enhanced under our defined key performance metrics. To provide a real networking tool, the decision of downgrading and enhancing the outgoing flows should be fully distributed. Therefore, in the next chapter, we provide an enhanced version of network flow

algorithm called maximum concurrent flow. Then a distributed version is implemented, tested and evaluated.

Chapter 9

Conclusions and Perspectives

9.1 Conclusions

In this thesis, we have studied virtual content distribution networks and answered to several problems that have arisen in the state of the art such as how, what, where, and when virtualize and optimize. Since this network helps to improve (optimize) end-user satisfaction, virtualization has been combined with optimization and several algorithms have been provided that can be applied to different networks.

Moreover, we contribute by a virtual infrastructure optimization simulator (vIOS): a high level application that optimizes vCDN services planning in both simple and complex active networks.

9.2 Perspectives

In this thesis we used exact and heuristic methods for software resources management. In next network generations, we have a lot of data, a great capacity of computation which favors the use of artificial intelligence techniques instead of building complex mathematical models.

Appendix A

Real cache overview

Real caching is the process that assures the caching service assisted with hardware H/W (CDNs equipment, video servers). We can enumerate three important sub-categories:

A.1 Intermediary caching

AKAMAI is a major company specialized in the provision of cache servers. Its server dissemination in the core network leads to serve quickly end users. In the literature, several algorithms exist to optimize the distance between the consumer and the content provider under the constraints of QoS. The question to answer was how to build a strong algorithm/architecture that intelligently places a lot of caches (centralized, decentralized or hybrid approaches). The relevant work is classified as follows:

1. Single cache: Shim et al. [160] proposed a single cache algorithm using LNCRW3U (least Normalized Cost Replacement for the Web with Updates) as a management mechanism based on the local greedy algorithm (described below). This approach is inefficient because it does not consider the popularity of video. Algorithm Local greedy: Select a node I and an item n. If item n is currently not stored at node I, and it has higher utility than some item

m (e.g. with minimum utility) that is currently stored at node I, then replace item m by item n)

2. Multiple caches: Fan et al. [59] described another alternative to the previous approach also called summary caching. In this protocol, every caching disposes of a table of summaries related to each proxy server. Cooperation of several caches can reduce the total traffic but it suffers from service availability (miss content scenario). This happens when neighbor cache servers do not have the content. Further, this approach requires extra bandwidth when a cache server serves the cache's misses of the others.

3. Hierarchical caching: This caching technique is based on placing nodes of caches in different levels of network structure (level1 core network, level 2 edge network level 3 access network). The first hierarchical proposal is probably the one in [111], where authors describe their time-based protocol aiming to solve hierarchical placing problems (HPP). This protocol may enhance the hit rate of a document. However, this hierarchical view increases the total latency because each level of the hierarchical caching structure introduced its latency. Further, storing documents in different level overloads the network and then decreases the quality of experience perceived by the user.

4. Distributed caching: Distributed caching algorithms place caching nodes only in the edge of the network and those nodes are the only ones that can cooperate in the distributed process. This algorithm is better than hierarchical caching as it is shown in [148] because it reduces the total latency, decreases the load and the required bandwidth. All this is through the use of a single level of cache. The author also introduced another approach that mixes the two previous algorithms called: "Hybrid Caching System". We have to highlight here, that in the distributed caching; one can face some problems such as difficulty in updating caches because of the use of a single level cache (long network distance) as it shown in [182] and [109].

5. Hybrid caching: In the Hybrid caching approach [148], authors propose to place a minimum number of nodes at the edge of the network. This approach reduces latency, load and bandwidth but suffers from the lack of protocols for managing caches. They also require complex topologies that cannot fit with other scenarios.

6. En-route caching: En-route caching [166] is an approach that intercepts the requests from the user and when the content is in the local cache of the interceptor, it answers the requester by sending the content. It leaves the request otherwise unchanged. This operation has some advantages [166]. The most important one is that it decreases extra overhead. Authors do not say how the cache is filled originally. Therefore, this technique is not sufficient because they do not consider video popularity that should be only filtered by those cache nodes. It suffers also from the fact that many CDN operators encrypt the original request so as to prevent such operations and to keep the hand on the process (Youtube).

7. Cache cluster: Borst et al. [26] introduce an algorithm for caching services to minimize bandwidth cost instead of latency in [92] and [148]. They use the hybrid caching structure. The replication strategy uses cooperation and has two different scenarios, intra-level cache cooperation and inter-level cache cooperation. The first is based on Local greedy algorithm described above while the second is a more simplified version of the greedy algorithm. Borst shows that the adopted algorithm improves cache performance but it does not take into consideration delay, QoE, network load which are important parameters/metrics for service video delivery optimization. Furthermore, they focused on a specific topology (they are not taking into account a general network topology) which made their solution fit only with such scenarios.

A.2 Direct caching

In direct caching category, caches are hosted in the core network. We can enumerate under this category the following techniques:

1. Caching proxy (CP): Meeyoung et al. proposed in [34] a single and large cache with only popular videos so as to push them near the end user. This basic technique maximizes the QoE perceived by clients and enhances cache hit ratio to be close to 51%. However, the main limitation of this approach is that it maintains only popular videos requiring frequent cache updates.

2. Weighted-Rank Cache Replacement Policy: In [152], a caching algorithm based on weighted cost (WC) calculation of any object (video) is presented. Objects with weak ranks are replaced from the cache. This technique assures better management of cache space and improves network bandwidth. However, it suffers from high delay required from WC calculation.

3. Multiple Descriptions Coding (MDC): Authors in [143] presented a caching mechanism based on video segmentation assisted with Least Recently Used (LRU) algorithm. This technique reduces server load and overcomes churn problems related to user dynamism (join/leave) in P2P networks. However, this technique can not deal with high user demands.

4. Popularity Aware Limited Caching: Authors in [164] introduced PALC to enhance cache capacity. Their technique showed 95% of cache efficiency as it was explained in the work. However, maintaining only popular video presented the most important limitation to this approach. Moreover, video online streaming requires an efficient cache update algorithm which is missing in that proposed work. Table A.1 summarizes the important features of direct caching.

Table A.1 – Direct caching techniques

Technique	Advantage	Improved metric	Limitations
CP [34]	Simple implementation, QoE enhancement	Video popularity	Single point of failure, increase server's load
WCRP [152]	Cache management	Network bandwidth	Increases the delay
MDC [143]	Reduces server's load	load	No scalability
PALC [164]	Cache efficiency	capacity	Maintain only popular video, no cache update

A.3 Indirect caching

In the indirect caching, caches are hosted by another operator. We can enumerate under this category the following techniques:

1. Collaborative cache mechanism based on resources auctions: Core networks in cellular systems manage their local subscribers and have their profiles. Jie et al. [43] introduced a new caching mechanism for collaboration between multiple Wireless Service Providers (WSPs). This collaboration is based on resource (Bandwidth) auctions. They propose that cache servers be deployed at Mobile Management Entities (MME) or Mobile Switching Center (MSC) so as to decrease the distance between contents and end users. This approach can enhance the QoE and maximize the response time for end users but it is inapplicable because of dynamicity of subscribers.

2. Content aware caching: Meghana et al. [7], start from the idea that cache sizes and link capacity are limited. Thus, she gives four techniques to properly manage the caches so as to minimize the response time for users:

   - Technique 1: Periodic Max-Weight algorithm with random eviction.
   - Technique 2: Iterative Max-Weight scheduling with Min-Weight Eviction Policy.

- Technique 3: Iterative Periodic Max-Weight algorithm.
- Technique 4: Iterative Periodic Max-Weight with Min-Weight algorithm.

3. Peer Aware Content Caching (PACC): Bjorkqvist et al. [24] introduce this approach to be dedicated to Content Distribution Networks (e.g. AKAMAI). The advantage of this approach is that it has not been oriented to a specific topology. It divides the network to three layers: two vertical layers and one horizontal layer. They also propose two policies PACC-AR and PACC-CL to execute the caching mechanism between peer nodes. This approach has some disadvantages as it does not take into consideration video popularity for cache management and cache update processes.

4. Cooperative Hybrid caching strategy for P2P Mobile Network: Mo Zhou et al. [190] designed a caching system to optimize network bandwidth and guarantee high response time to mobile users in order to increase the QoE and the overall QoS. The introduced technique is based on two strategies: single greedy caching strategy and cooperative hybrid caching strategies. It is shown that the second strategy outperforms the first one in some cases. The structure used in this approach is not hierarchical or tree topology based but P2P. As authors focus on optimizing the traffic over the network, the caching strategy takes into account {video popularity, distance, video size} as metrics for the optimization. Cache replacement (eviction) procedures are managed through the single greedy caching. Cooperative hybrid caching is based on popularity of video content. This technique requires high computing and does not take into account user satisfaction. Moreover, authors do not give a clearly popularity function. Furthermore, they do not account for extra bandwidth needed to seek the closest desired video.

Appendix B

Virtual cache overview

The virtual caching is the process that assures the caching services and hardware independence. Precisely, we can enumerate three important sub-categories:

B.1 Content moving fetching

We will enumerate the few algorithms proposed for virtual caching techniques in video delivery content distributed networks:

1. Prediction Based caching (PBC): Bogdan et al. [31] introduced a cache replacement algorithm that answers the question: which videos to evict from the cache? Authors describe then content moving algorithm based on decision making after a cache miss scenario. They describe an algorithm that works as follows:

   - If (the requested video V is in the cache) then stream it.
   - Else forward_request_to_other_site(V);

   The decision was based on penalty in order to decide either keep a copy of the item and transfer the video or stream it and not cache. This technique enhanced CDN scalability but it still suffers from achieving high quality of service (including delay, bandwidth) due to the given penalty function which

Table B.1 – Content moving techniques

Technique	Advantage	Improved metric	Limitation
PBC [31]	Scalability	Popularity function	Complex
CaaS [70]	Network utilization	Throughput	Missed architecture

requires high computational overhead and extra bandwidth to calculate the instantaneous value of the popularity.

2. Cache as a Service (CaaS): Cache as a Service [70] is a new caching technique for fetching requested items and moving them to end users. The results showed that this approach improve network utilization and user QoS. However, it requires a predefined architecture in order to deal with NFV and SDN technologies. The summary is in Table B.1.

B.2 Server moving replication

We will present in this subcategory the pertinent replication techniques in the literature.

1. Replication Algorithm with Load Balancing (RALB): In [191], authors proposed the replication algorithm RALB to enhance video adaptation to the available bandwidth. This approach is based on content replication in P2P systems for video on demand scenarios. Peers store some movies in order to increase network bandwidth and reduce server's load. Authors use random replication assisted with load balancing. Firstly, they introduced a centralized approach where video replication is static. Then, they add a reactive and distributed algorithm in which peers decide, based on video popularity, to either store or discard the movie. Hence, the replication is amended after any peer views video. Therefore, this mechanism adapts to the dynamicity

of nodes and peer churn. However, the authors did not specify neither the video popularity function used nor the cache update mechanism which are most important in video caching.

2. Optimal Replication Strategy ORA: Weijie Wu et al. [179] introduced ORA strategy which requires knowledge of video popularity. This approach addresses two issues. The first addressed the optimal replication ratio while the second addressed how to do this replication? Results showed that the algorithm increases optimality when we replicate more popular video and remains sub-optimal like proportional replication strategy.

3. Home Box assisted CDN (HB-CDN): Soraya et al. [35] describe an algorithm introducing a new equipment or gateway as a middle-ware between the CDN and end users. This equipment is for content caching and acts as a content provider for the end users. It reduces server load, and minimizes time delays. Some DSL operators start to have an equivalent caching system in the box especially useful for replay and short delay replay (almost real-time broadcast).

4. Incentive Caching For P2P-VOD Systems (RBS): Weijie et al. [180] proposed a reward-based scheme (RBS) in large scale p2p on demand streaming. This algorithm offers two pricing schemes where the content provider proposes strategies in order to serve the desired content to the end users while taking into account the operational cost of the content provider. The first strategy is used to meet the high demands of users. It takes into account the relationship between users demand and the number of replications of content. The second strategy is used to reduce the reward of items with high operational cost. Authors showed that their technique reduces the cost and enhances the access latency.

5. Two-Level Result Caching (TLC): Erica et al. [150] introduced the problem of load balancing which can be defined as: where should we place the replica? Therefore, they gave TLC technique in order to solve this problem: assur-

ing load balancing between cache servers and achieve high throughput. This technique uses the Lest Recently Used (LRU) cache management algorithm in order to maintain a LRU result cache LRU-RC combined with a DHT to know the position of peers holding the replica of content requested in the recent past. The main disadvantage of this techniques is that increases jitter (thus increases delay) which is an important metric for video streaming applications. Moreover, authors do not take into consideration the popularity of the video.

6. Proportional Replication PRA: Saurabh et al. [168] described the proportional replication algorithm (PRA) in peer to peer networks. The algorithm is based on a proportional replication strategy that maintains a number of replicas of each content proportional to the request rate of it. LRU, FIFO, or LFU techniques are used as a cache replacement algorithm.

7. Lazy Replica: Bin at al. [36] introduced the Lazy Replica strategy for content replication in peer to peer networks. This replication approach introduces two assumptions that cannot be known in advance: the time of departure of peers and the content popularity. Assuming these conditions, this technique has to do the replication of the most popular content in peers having minimal probability of departure. It decreases server load by 15%, improves throughput and latency.

8. Intelligent Replica Placement/Management Algorithm (QIRMA): Ayyasami et al. [14] described an intelligent replica technique. Authors showed that their approach achieved good results. It enhanced throughput, jitter, network traffic, latency and fault tolerance. However, it leads to some bottleneck problems because of excessive redundancy which may lead to network overload), low availability, congestion. Finally, Table B.2 summarizes all the previous techniques in terms of advantages, improved metrics, and disadvantages.

Table B.2 – Replication techniques

Technique	Advantage	Improved metric	Limitation
RLB/ARLB [191]	Video adaptation	Video popularity	Increases computational load
ORA [179]	Gives popular videos	Replication threshold	Not fit with proportional replication
HB-CDN [35]	Reduce server's load, load balancing	Load	Extra time delay
RBS [180]	Increase satisfied user as possible	Reduce the cost	Miss QoS, high price
TLC [150]	Manages the cache	Cache position	High delay required
PRA [168]	Reduces server's load	Network bandwidth	Video popularity is not considered
LAZY REPLICA [36]	QoS parameters, reduce server load	Throughput	High operational cost
QIRMA [14]	QoS parameters	Throughput, jitter, delay	Bottleneck and Congestion problems

B.3 Session moving

Session moving is the third category in virtual caching techniques. It can be realized using software defined network solutions or Mobile IP/IPv6. The realization can be done either using classical cloud brokers or new terminologies like NFV or SDN as it will be detailed in the next section.

Appendix C

OMAC: Optimal Migration Algorithm for virtual CDN

OMAC algorithm is a reformulation of OPAC algorithm that considers vCDN migration problem inside network operator.

C.1 OMAC: scenarios

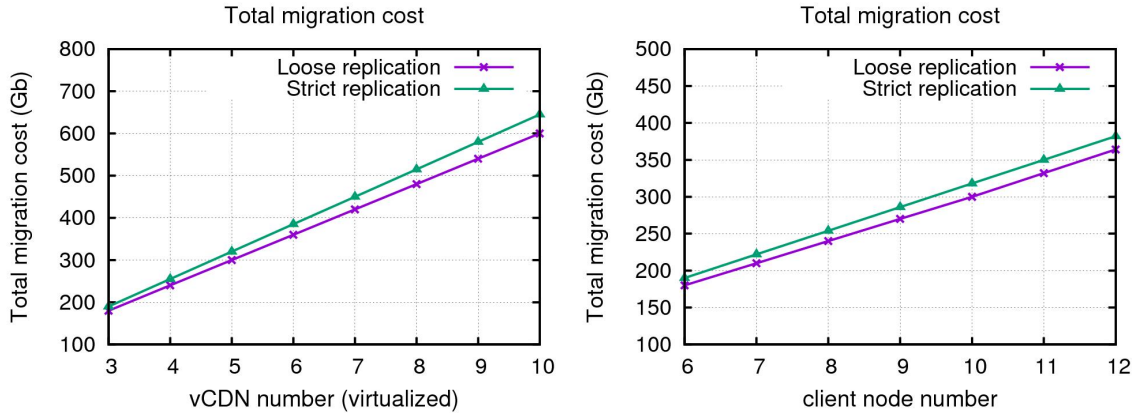
In order to assess and quantify OMAC behavior, we proposed two scenarios: 1) strict replication, and 2) loose replication. The two approaches are defined as follows:

1. In strict replication, the total number of replica in the network is bounded and each vCDN node is replicated at most one time. The vCDN replication process obeys the following equation:

$$\forall f \in F : \sum_{s \in S} x_f^s \leq 1 \tag{C.1}$$

2. In loose replication, the total number of replica in the network is dynamic and the vCDN nodes are replicated freely.

Since we are focused on the total migration cost as the objective function, we



(a) vCDN number impact (b) Client node number impact

Figure C-1 – Total migration cost

have measured this cost in the two scenarios as shown in Fig. C-1.

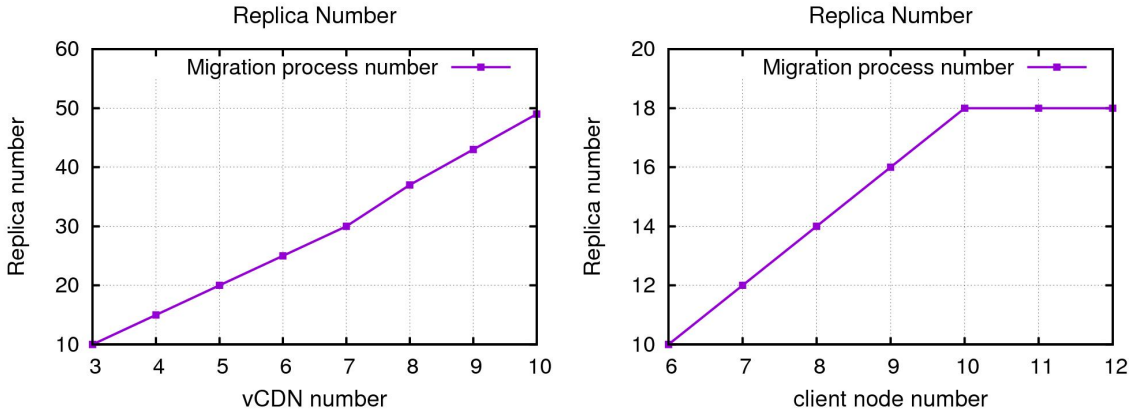
The result shows that in small scale, the loose replication approach is slightly efficient in term of migration cost comparing to the static replication one. This is because in the case of one replication, the minimum distance between s_f and s is great and therefore, the migration cost will be significant. In the other hand, in loose replication, we replicate more than one time, but there is no need to reach the access layer for example. Recall that we have focused on minimizing the migration cost and not the content retrieval delay or the response time.

Moreover, the total number of replica is measured as follows:

$$Replica\ number = \sum_{s \in S \setminus \{s_f\}} \sum_{f \in F} x_f^s \tag{C.2}$$

In strict replication, the replica number is less or equal to $|F|$ while in loose replication scenario, which is the most efficient in terms of migration cost and network resource saving, the replica number is variable in response to the change in client group demand matrix defined above $\left(d_v^f\right)$.

Fig. C-2 shows that the number of replica in "loose scenario" increases with $|F|$ and $|V|$ while it reaches a static value at $|V| = 10$. Therefore, in terms of migration cost, we have an open question that we do not answer: what will happen if we add



(a) vCDN number impact (b) Client node number impact

Figure C-2 – Replica number impact

an extra-constraint for limiting the replica number ? and is it efficient or not ? A first answer to this question could use an additional constraint as follows:

$$Replica\ number = \sum_{s \in S \setminus \{s_f\}} \sum_{f \in F} x_f^s \leq R^f \tag{C.3}$$

Where R^f will be the maximum replica number of vCDN ($f \in F$).

Therefore, such an open question may try to find the appropriate R^f that can be fixed in the mathematical model ?

C.2 Conclusion

Two approaches for OMAC are proposed and results show that loose cache replication/migration algorithm is slightly more efficient.

Appendix D

Distributed Maximum Concurrent Flow Algorithm

D.1 Introduction

In ICN protocol, data consumers send an *Interest* packet with a name or name prefix and retrieve, from the near LAN, a *Data* packet named under that prefix and it can be publicly verified.

To deploy this facility in network operators, optimization algorithms are necessary to select the appropriate placement of ICN functions. They supposed to take into consideration system, network and quality parameters. However, in complex active network where intermediate nodes not only intercept consumer interests and respond back but only treat the information and modify the original content (for example adding building plan in augmented reality use case, advertisement in video streaming scenario or even patient organs in medical treatment) each source of data (server) satisfies the request a group of consumers concurrently.

Streamed data are flowed from a single source to multiple destinations simultaneously. Target nodes correspond to consumer groups interested in source's data. However each consumer group requests a specific quality of experience corresponding to a traffic class. It can be easily translated to a quality of service fol-

lowing the exponential relationship as in IQX hypothesis.

The above problem is a *pure QoS routing problem* at origin that solves the following question:

How to route the quality of service from source s to all destinations ?

Currently, the maximum flow minimum cut theorem affirms that the maximum amount of data that can be shipped from a source to a destination is equal to the minimum cut capacity that separates the source from the destination. This theorem is important however it is not of practical interest in ICN network for two main reason: *i*) it is a centralized (sequential) approach which needs a global view of the topology beside the dynamic related parameters such as residual link bandwidth. *ii*) it is a point-to-point approach which does not fit with single source multi destinations scenario such as in this ICN use case. Giving these reasons, in this chapter we try to extend the above theorem (i.e., maximum flow minimum cut) to design a distributed algorithm fitting with the ICN context. Further, the (multi) point to multi-point [1] communication model is adopted from the origin graph. We try therefore to propose efficient QoS routing algorithm that dynamically detects the bottleneck between the source and the set of destinations, place ICN functions to treat/process (amplify or downgrade) the streamed data in edge/optimal locations and stream the maximum quality (max-min fairness and throughput) to consumer groups.

In networking theory, minimizing the overall congestion is often referred to as maximum concurrent flow problem (MCFP). Several solutions are proposed to solve this problem in sequential or distributed manner. Each solution represents a computing model of the MCF problem:

1. Sequential approaches for solving this network problem are based on specific frameworks. Although each framework considers its specific algorithm and logic, the common idea is to route the concurrent demands from sources to destinations without violating the capacity constraint on each edge. Network

[1] multi-source multi-destination is trivial step of single-source multi-destination

is a centralized, decentralized, or distributed reservoir of networking and processing for concurrent flows. The objective is to maximize the throughput of traffic. Based on the underlying networking, the heuristic relaxation solution that iteratively pushes small/large amount of flow within the network is the main approach in MCF problem.

2. Distributed approaches follow the same logic to solve the problem but in a cooperative manner between processes installed on different node locations. Still, the objective these approaches is to maximize the minimum and common fraction that satisfied the demands of each commodity.

Up to now, exact and heuristic algorithms proposed in the previous chapters are centralized by SDN. A global view of the network is needed to calculate an optimal topology of routing requests. The underlying infrastructure is dumpy and does not take any decision of traffic migration or optimization. Now, and following the introduction of active nodes such as ICN that decide an increase or degradation of flows, one needs a distributed version of the maximum flow algorithm used in the previous chapters. So we propose in this chapter, a distributed single source maximum concurrent flow algorithm called SS-DMCF.

The rest of this chapter is organized as follows: Section D.2 highlights the state of the art of the MCF algorithms. Section D.3 introduces the potential MCF computing models. Then, Section D.4 gives execution methods of MCF. Section D.5 studies the combinatorial MCF optimization models. Section D.6 gives an idea about the potential applications the may rely on MCF. Section D.7 highlights our proposed distributed MCF algorithm. Section D.8 compares the MCF optimization techniques and the work is concluded in Section D.9. Different pseudocodes of the used algorithms are given in annexes.

D.2 MCF state of the art

This section highlights the relevant Maximum Concurrent Flow (MCF) algorithms in data-centric context [137].

Maximum concurrent flow problem is a generalization of sub-problems related to the multi-commodity network flow which make it harder when solving large scale deployment problem.

In 1930, Russian Railway System studies the maximum flow problem (MFP). The objective was how to find the maximum amount of flow that can be pushed from a supply node (a source) to a demanding node (a sink or a destination). Then, Fold Fulkerson gives the method of an augmenting path (called Ford Fulkerson method later) that introduces the notion of cancellation flow, residual capacity/graph, and augmenting flows. Different approaches are then proposed to calculate the maximum flow between a given pair of node in the network. Later, Fold Fulkerson introduces the maximum flow minimum cut theorem in this case of network, solving the single-commodity flow problem. Authors remark, later, that the proposed theorem and and its corollaries for MFP cannot be used in multi-commodity network flow. They affirm that the max-flow min-cut theorem cannot be extended to more than two commodities.

Giving that multiple commodities share network edges, concurrent commodity demands should not violate edge capacity while maximizing their throughput. MCF solutions are then proposed. Exact solution algorithms are proposed in the literature to solve this problem using linear programming mode, simplex method, etc. Different running time in worst case scenarios are shown in the literature of exact MCF approaches. The approximation solutions for MCF focus on minimizing as much as possible (in ε-approximate far from the optimal) the overall congestion on edges, α. It represents the sum of flows over all commodities on an edge *f(u,v)* divided by the capacity of the edge *c(u,v)*.

$$\alpha = \frac{\sum_{i=1}^{k} f_i(u,v)}{c(u,v)}$$

In approximation algorithm, a feasible concurrent flow vector is pushed initially from sources to destinations. Then, the approximate value of MCF is:

$$\alpha \leq (1+\varepsilon) \times \alpha^*$$

Where α^* is the exact MCF solution, $\varepsilon$is the approximate parameter.

Among the main MCF approximation algorithms, we quote the primal-dual approach proposed by Danzig. Different works are also proposed in this field. Indeed, Shahrokhi et al. [157] propose the maximum concurrent problem with uniform capacity and exponential length metric. The proposed algorithm is initiated by choosing a length function ($l(e)$) and finding an initial feasible flow vector (generally a small amount of flow that can be routed concurrently). Then, a shortest path in terms of the proposed length function is found for each entering commodity. The algorithm then does a rerouting to the flow. It terminates in O ($\varepsilon * n * m^7$). Klein et al. [104] then proposed a faster algorithm with a different length function. Using the amended length function, a larger amount of flow is sent by iteration. Leighton et al. [112] generalizes the MCF solution considering arbitrary edge capacities and demands.

We highlight also the work of Garg and koeneman [69] that leverages the primal-dual method, the method of Karakostas [101] that aggregates a set of commodities by the source, and the flow deviation method which is introduced by Fratta [65]. The approximation algorithms for the MCF problem use in general either incremental or flow deviation based approaches. In the incremental based approach, length function is $l(e) = e^{\alpha * f(e)}$ while in the flow deviation based approach, the length function is $\frac{c(e)}{(c(e)-f(e))^2}$. Note that the approximated solution not only depends on the ε parameter but also on the size of the system inputs.

D.3 MCF computing models

MCF is a network flow problem which can use different computing models. In this section, we detail these frameworks.

D.3.1 Centralized MCF models

The idea in MCF centralized method is to have a super node acting as a traffic engineering server that routes the demand from production locations to appropriate destinations. Super node may use different heuristics to find approximate solutions for the MCF problem. It can iteratively add small amount of flow (incremental method) or it can add large amount of flow, then it redistribute (reroute) the flow from the most congested (loaded) edge/path to the less congested one.

In centralized computing, super node machine executes only one MCF instruction at a time. Giving this requirement, single CPU and RAM are allocated to execute the sequence of MCF instructions sequentially. Parallel computing for solving the MCF problem is beneficial in the way that multiple processors (CPUs) and RAMs collaborate to execute the set of the MCF instructions in parallel. Note that parallel computing use also one machine to execute the MCF problem.

D.3.2 Distributed MCF models

In the state of the art, we found a few relevant algorithms that aim to solve the distributed MCF problem (DMCF) (i.e., MCF in a distributed manner). In the classical DMCF, such an algorithm inheres the same baseline mathematical formulations of the multi-commodity flow problem. Then, the decision of forwarding the flows of the commodities is taken by local nodes (computing units) as proposed initially by Awerbuch et al. [12]. Then, they improve the classical approach and present a novel decision model for the DMCF based on the multi-agent system or the billboard model [13]. Computing units representing the agents that are responsible for commodities. In the billboard, each agent can read the total edge flow within the network in an algorithm round r, which is the sum of the

edge flows of each commodity. Each agent then has information about the total edge flows and writes the next edge flow (to be routed) of its commodity in the next round $r+1$. The complexity of this method is linear with the maximum path length size.

In the aforementioned distributed computing models, multiple machines (CPU, RAM) collaborate to solve the DMCF problem (DMCFP). Two types of communication between these machines are presented in this computing model: 1) communication using messages and 2) communication using a shared middleware. In each communication model, two main execution models may be presented: synchronous or asynchronous models.

Hereafter, we detail the main distributed algorithms which deal with the MCFP.

Local decision based DMCF

In this approach, the decision of routing network flows on an edge is taken locally by intermediate nodes. Network nodes balance the flow on its corresponding edges. The process ends in determined number of phases as proved by Awerbush et al. [12]. The main algorithm works as follows: In the first phase, each source pushes $\frac{(1+\varepsilon)*d_i}{\delta_i}$ amount of flows on each of the δ_i edges incident to each source S_i. Then, some instructions should be executed at each node which is very costive. Therefore, the following approach for the decision is introduced, the global decision.

Global decision based DMCF

In a Global decision based DMCF, a decision is related to each commodity. A multi-agent framework is used to solve this distributed problem rather than local decision procedure as defined above. Each agent is responsible for routing the flow of its commodity taking into account the current congestion (load) on each edge.

Two distributed solution are proposed by Awerbush et al. called: 1) a greedy distributed optimization, and 2) an approximate steepest descendant framework.

In the approximate steepest descendant framework, authors propose the following algorithm that solves the DMCF. It is executed in T phases and T_p steps:

1. ${f^e}_i \overset{flow}{\longleftarrow} \varepsilon * c_e/Ks$
2. $T \overset{phases}{\longleftarrow} O\ (log(m)/\ \varepsilon^2)$
3. *For (T phases) do*
   (a) $T_P \overset{steps}{\longleftarrow} O\ (L^* log^2(m)^*\ log(\ k/\varepsilon))$
   (b) *For ($T_p$steps) do*
      i. ${f^e}_i \overset{flow}{\longleftarrow}$ *current flow value*
      ii. *for (each commodity i, such that d_i/T flow between ($s_i$and t_i) not yet routed) do in parallel*
         A. ${C^e}_i = \frac{\varepsilon^2}{\log(m)} {f^e}_i$
         B. *Compute {blocking flow} under capacities ${C^e}_i$from s_t to t_i that routes flows along (1+ ε)-approximate shortest paths (in terms of congestion (l(e))*
         C. *Route this flow / $d_i/T \geq$ this flow*

Authors raise the potential use of MCF for QoS routing including the pure routing and flow control issues. They claim that MCF problem models exactly the pure routing problem in networks. The main idea is to route a small amount of flow (of all commodities) on edges. This amount need to be placed initially which determines a feasible flow. Then, this amount is increased in a multiplicative way. Authors fix an edge length function (the metric) as follows:

$$l\,(e) = \frac{m^{\frac{\alpha(e)}{\varepsilon}})}{c(e)}$$

The blocking flow technique used here is computed by sending flow from the source to the destination along the shortest path in terms of *the previous length function*..

In the greedy distributed framework, authors use the same framework. Indeed, multiple agents operate in cooperative manner but uncoordinated way. Congestion and bandwidth management are raised as the main application of their work. They propose a stateless greedy distributed algorithm for the concurrent multi-commodity flow problem with poly-logarithmic convergence. The approximation is done around 1+ ε distance from the exact solution. The running time is $O(H * log^{O(1)} * m * \frac{1}{\varepsilon}^{O(1)})$ **where *H* is the maximum allowed path length.**

As differentiation to all the previous flow control algorithms established for the TCP/IP stack (additive increase in the windows size representing the maximum of the allowed packets that the sender can send to the receiver), authors use here a multiplicative increase process which is more aggressive than the decrease. Their objective was "resource allocation in decentralized network architecture". Authors affirm that different compete applications over a shared network resources can be solved in greedy manner. Indeed, they enhance the current flow control (TCP) and the routing (IP) mechanisms based on shortest path calculation in terms of hops (TTL) because it does not guarantee the optimization of resources in a scenario of concurrent flow and demands. To deal then with this hard problem (MCF), authors propose a novel routing metric (link cost) that depends on the congestion that runs on an edge (i.e., the load). This routing metric may identify the network bottleneck (sparsest cut problem) and then ease the maximization of the concurrent throughput (concurrent flow). Authors introduce then problem of concurrent multi-commodity flow problem by including the following congestion indicator:

$$l(e) = \frac{f(e)}{c(e)} = \frac{\sum_{i=0}^{i=k} f_i(e)}{c(e)}$$

The objective function of their work is then as follows. Minimize (congestion) $\stackrel{\text{def}}{=} Min(Max(\ l(e)\ ) \stackrel{\text{def}}{=} Min(|\text{f}\ |)$

They introduce then the billboard framework. It is a computing model that is distributed, based on multiple agents that operate in coordinate manner but uncoordinated way.

Each commodity is handled by an agent (computing or a processing unit (CPU)). There is a global clock access by all the commodities to satisfy the computation requirement. Nevertheless, the approach is still distributed. Each edge is a billboard in which the agent records the current congestion of its commodity on that edge. Multiple agents can read/write in the billboards. They can read current congestion (total amount of flow) at each edge. Based on that framework, each agent can decide how to reroute next flow and write the values on the corresponding billboards.

To recapitulate, authors propose a very interesting distributed algorithm for the MCF problem. They introduce routing metric, flow control rules, and a greedy re-routing procedure. The routing metric is function of the current congestion representing the edge cost and how the cost of this edge increases with congestion. This metric is then used to encourage agents to reroute their traffic (flows) in the less congested paths (shortest paths in terms of congestion metric). As a summary, each agent i (responsible for the i^{th} commodity) read the total flow on an edge, calculate the routing metric, and finally do the flow control and the re-routing of the traffic.

Previous DMCF work

Leighton et al. [112] define the max-flow of a multi-commodity flow problem as the maximum throughput f such that f^*d_i units of commodity i is simultaneously routed for each i without violating the capacity constraint. Then,

$$f \leq \frac{C(u,v)}{D(u,v)}$$

where f is the maximum concurrent throughput, v is the complement of u, C is the capacity of the minimum sparsest cut, and D is the demand tagged on the corresponding cut.

Jonathan et al. [103] propose an algorithm for calculating the ε-approximated solution of a MCF problem in an almost linear time. The proposed approach is

complex and uses oblivious routing as a basic tool but also requires many concepts (flow sparsification, electrical flow, solving Laplacian systems, etc.) but it seems very promising for the practical solving of routing problems in networks.

H. Räcke et al. [145] introduced the concept of oblivious on-line routing in which each origin-destination request is routed without any knowledge of the state of network congestion. The surprising result is that there is such a routing mechanism for which the network congestion is not more than 0(log n) of the optimal congestion. This work is interesting in more than one way: on one hand, because on-line routing gives a benchmark on the quality of the dynamic and distributed routing, and on the other hand, because this benchmark is relatively robust and good. Unfortunately, the method proposed in [15] to calculate such routing requires solving NP-hard problems. This algorithmic complexity is reduced in [15] to a super-linear polynomial complexity, which remains crippling for large networks. The notion of oblivious routing can also be analyzed in the context of robust routing, where one seeks to calculate an optimal routing when the demands are uncertain [10].

D.4 How to execute DMCF

In DMCF, each processor has its local memory and it is responsible for executing a part of the DMCF algorithm. Basically, the algorithm is executed after a terminating a set of phases. In each phase, we have the notion of round or steps. The execution of the algorithm instructions may be fully or partially synchronized or asynchronous.

In synchronous execution, each processor has its local memory and it is responsible for executing a part of the DMCF instructions. In asynchronous execution, each round corresponding to a commodity (agent) is started without a clock indicating the beginning of the round. This execution is used by Awerbush et al. in their work (distributed MCF) in which he gives an upper bound of the number of phases (rounds). Hence, there is a determined round number sufficient to estimate

the MCF value. The execution model used is the hybrid model: authors synchronize the beginning of each round. Nevertheless, the execution of the instructions in each round (steps) is asynchronous.

D.5 MCF combinatorial optimization models

In combinatorial optimization (e.g., network flow, etc.), we quote exact and approximation algorithms. Most of the combinatorial optimization problems are Not Polynomial (NP) time (NP=? P). This means that those problems are hard to solve (in polynomial time) but easy to check the solution once it is given (yes/no answer). Therefore, approximation algorithms are also used to solve such problems given large combinatorial optimization problem instances. Approximation algorithms are analyzed mainly through the running time in different use cases (empirical, average, and worst cases).

The main formulations of the exact MCF are edge-path formulation and node-edge formulation. The objective of these formulations is to minimize the overall congestion which is equivalent to maximize the throughput or the concurrent flow. The constraints of the MCF problem are demand satisfaction, edge capacity, and node balance (Kirchhoff law). Two extra formulations are presented in the literature which are tree formulation and compact formulation. In tree formulation, trees are generated for each commodity that has multiple destinations. In compact formulation, linear programming is still used. However, the system size of constraints and variables are reduced. The main approximation algorithms for MCF are: Incremental or rerouting.

D.5.1 Exact models

MCF is solved through the well known node-edge or edge-path model formulations. In edge-path formulation, the throughput of commodities is maximized representing the upper bound relative congestion on edges.

Edge-path, node-edge and tree formulations

We denote the set of all paths between end nodes i and j for demand pair $\{i,j\} \in D$ by P_{ij}. The union of P_{ij} over all demand pairs is denoted by P, and the set of edges in path p is denoted by E_p. According to these notations, the equivalent MCF formulations (edge-path model is modeled below) is given as follows using linear programming technique:

$$\max(\gamma) \tag{D.1}$$

$$s.t.$$

$$\sum_{p \in P_{i,j}} f_p = \gamma * d_{i,j} \forall \{i,j\} \in D \tag{D.2}$$

$$\sum_{p \in P : \{i,j\} \in E_p} f_p \leq c_{i,j} \forall \{i,j\} \in E \tag{D.3}$$

$$f_p \geq 0 \; p \in P \tag{D.4}$$

where γ is the throughput and f_p is the amount of flow on path p. Constraint set (D.2) ensures that the same proportion of demand is met for all demand pairs, constraint set (D.3) enforces the capacity limits on the edges, and (D.4) are non-negativity constraints on the flow variables. We refer to a solution $\{f, \gamma\}$ to the edge-path formulation as an edge-path solution.

Triples formulation

The solution of the above formulations gives an exact value (throughput) of the MCF problem. However, the system size is high which causes significant running time. Matula et al. [53] proposes in a recent work a compact linear programming formulation of the MCF problem. Authors introduce the triples formulation to solve the problem. The idea is to include a novel variable such as $x^k_{i,j}$ indicating that the edge *(i, j)* diverts this amount of flow through on an intermediate node

k. Authors prove that their solution is faster in terms of running time than the previous formulations (edge-path and node-edge). They didn't compare their work to the tree based formulation. In their proposed work, they achieve three main contributions: 1) a triple based formulation for MCFP 2) a proof of equivalence, and 3) an efficient (linear programming) size computation is presented comparing to the well known exact formulations.

Hereafter we describe their work. Authors define an MCFP instance as the triplet: *<G, c, d> where G is the undirected graph, c is the capacity function, and d is demand function.*. Then, the source and the destination are defined as the pair {i,j} for $i<j$.

Originally, the idea is derived from the node-triples formulation of Matula that gives a proof of the dual between Distance Elongation Problem (DEP) and the MCFP. The exact formulation (triples) is envisioned from the introduced "direct flow solution" (see heuristic model). In this formulation, authors present a linear programming (LP) system using an introduced variable indicating the total flow diverted off the edge {i, j} through the node k defined as $x^k_{i,j}$. It is introduced to satisfy the demand while keeping the edge capacity constraint not violated for all the edges which is not guaranteed in their heuristic relaxation procedure.

Formally, the total flow on an edge {i, j} equals to the initial direct flow **plus** [the amount of flow diverted **onto** {i, j} (through the nodes i or j)] **minus [**the amount of flow diverted off {i, j} (from the nodes i or j]**.** This amount of flow should not exceed the edge capacity $C_{i,j}$. This introduces the triples formulation which is formulated as follows:

$\max(Z)$ *s.t.*

$Z * d_{i,j} + (\sum_{i=1}^{k} x^i{}_{k,j} + x^j{}_{k,i}) - \sum_{i=1}^{k} x^j{}_{k,i} \leq - C_{i,j}$

Then, authors compare the LP sizes of the exact MCF formulations to show the efficiency of their triples formulation in terms of running time. Moreover, they propose a deriving procedure to move across formulations using basic mathematical models.

D.5.2 MCF approximation models

The approximation model is based on two main methods:

Maximum flow based approach

This approach is based on calculating the maximum flow and then saturating at least one edge for each commodity by sending the flow of this demand. Then, given the flow concurrency behavior on the edge bandwidth, a rerouting process for deviating some flow from the loaded edge to the less congested ones is done in an intelligent way. This solution called also the re-routing procedure for MCFP. Firstly, it is introduced by Shahroukhi et al. [157] by formulating an exponential length function $l(e) = e^{\frac{\alpha * |A|^2 f(e)}{\varepsilon}}$ where |A| represents the cardinal of the edges, α is a constant and $f(e)$ is the total amount of flow circulating on an edge. Authors use this framework to reroute flow from highly congested (utilized) edges to less congested ones. The improvement of this framework is started by Klein et al. [104] and Leighton et al. [112] who propose different edge length functions.

The main step of the maximum-flow based approach is as follows. First, an exponential length function is initiated as below:

$$l(e) = e^{\frac{\alpha * |A|^2 f(e)}{\varepsilon}}$$

Then, for each commodity defined by the triplet (s_i,t_i,d_i) the flow is sent from s_i *to* t_i along the path of minimum edges. Then, a comparison between $max_e f(e)$ and $\frac{\sum_{i=1}^{i=k} dis(si,ti)}{\sum_e l(e)}$ is verified. If the first term is greater than the second term then we get the ε-approximate value to the maximum concurrent flow. Otherwise, a shortest path for each commodity is calculated, and we choose the commodity with maximum $l(longest\ path\ except\ the\ shortest\ path)$ where longest path is the path which larger hops between the source and the destination of the commodities. Then, we re-route a small fraction of flow from the most congested (high $l(e)$) longest path to the shortest path. The termination condition of their algorithm is

the satisfaction criteria in the comparison between the previous terms.

Following the same idea, Klein et al. [104] modified the length function to include maximum commodity flows aver all the edges $|f| = max_e f(e)$. Later, Leighton et al. [112], and Radzik et al. [146] generalized the length function and consider an arbitrary capacity function (instead of uniform capacity in the work of Shahroukhi and Matula [156] [157]).

To summarize the maximum flow based approach in calculating the maximum concurrent flow, we list below the main steps. First, we calculate the max-flow for each commodity. Then, we reroute independently, the max-flow value of each commodity. Finally, we re-route a small fraction of the flow from the loaded edge to the minimum-cost flow path.

Shortest path based approach

In this approach, new technique is introduced which is based mainly on finding a shortest path in terms of length function (less congested path) instead of hops. In each iteration, the algorithm pushes a small amount of flow along the calculated shortest path. For more clarification, we push a unit of flow (for example) along the shortest path and then we update the flows.

Garg et al. [69] solve the problem in a polynomial bound of phases. In each MCF phases, there is k iterations. For an iteration related to a commodity, authors push the minimum capacity (C_{min}) of the shortest path. Later, Madry et al. [115] improve this approach to execute Dijkstra for finding all pair shortest paths. The length function used in this technique is exponential and depends on the current flow plus the old routed flow.

D.5.3 Heuristic model

Recently, Matula et al. propose an extension of the MCF problem to identify the hierarchical community structure and hubs in networks. They leverage the duality between maximum concurrent flow and the minimum sparsest cut. Moreover,

they give another heuristic as explained below:

Heuristic relaxation procedure (direct flow solution)

Matula et al. [53] inspired in their recent work a new solution for the MCF problem. The proposed heuristic is as follows:

1. Initially, the algorithm proposes to route $z*d_{i,j}$ units of flow between all pairs {i, j} to find a feasible solution.

2. Then, it tries to satisfy (i.e., according to edge-path formulation) the equation that assure the data consumer demands satisfaction:

$\sum_p \ f_p = z*d_{i,j}$

The proposed heuristic (direct flow solution) may not necessary satisfy the capacity constraint of edges defined as ($\sum_p \ f_p \leq C_{i,j}$)

1. Then, at a throughput level (a fixed z value), we verify if $z*d_{i,j} \leq C_{i,j}$ *or not*

2. If not, the heuristic selects a pair {i, j} that has too much flow (larger than the capacity).

3. Diverts an amount of flow from {i,j} to {i, k}.

4. Repeats the "divert procedure" until the edge capacity constraint is verified for all the network edges.

D.6 MCF applications

Among the main application in MCF, we quote the sparsest cut problem (SCP). This problem finds the critical edges (bottleneck) in the distribution network. Finding SCP is NP-hard problem. Therefore, for estimating the sparsest cut, we measure the sparsity ratio of the cuts. The cut with minimum sparsity ratio is the sparsest cut.

In a distributed network, the transportation of commodities is based on bandwidth allocation mechanisms. MCF is used to solve this problem while minimizing the operation cost.

In circuit electric, multi-commodity flow problem is the sparsest cut used to route current between transistors. Further, MCF is used for the routing (re-routing) of commodities by proposing novel routing metrics. Further, Generating independent key for independent users simultaneously is problem that can be easy solved by MCF exact approaches. MCF is used also to control the traffic between endpoint in order to avoid congestion caused by concurrent end-to-end flow control mechanism MCF is then a potential hop by hop congestion control for ICN.

D.7 Proposed DMCF model

We propose a novel DCMF algorithm for routing data consumer demands and recommending caching points. It is an exact model, distributed using local decision computing model, and asynchronous in terms of execution. Hereafter we describe our algorithm. We leverage the theorem of decomposition to aggregate the set of commodities entering in our framework using the well known origin-specific problem (OSP) formulation in the state of the art. In this formulation,

Algorithm 8 Sequential Maximum Concurrent Flow

1: **Input:**
A directed graph $G = (V, A)$ with capacities $(c_a)_{a \in A}$
A set of commodities (demands) $(s, t_k, d_k)_{k \in K}$
2: **Output:**
The value γ^* of the maximum concurrent flow between s and $T(s)$.
3: **Initialization**
$\gamma^* \leftarrow 0$
4: **While** there exists a shortest-path tree (in terms of hops) SPT spanning $T(s)$ in R_G:

- Find the maximum flow γ such that $\gamma \times d_k$ can be concurrently routed in SPT between s and t_k
- Augment the flow and update R_G
- $\gamma^* \leftarrow \gamma^* + \gamma$

end while

Algorithm 9 Distributed Maximum Concurrent Flow

1: **Input**

- Set flows in all edges to 0
- Set residual graph R_G equal to initial graph

2: **Output:**
The value γ^* of the maximum concurrent flow between s and $T(s)$.
3: **Initialization**
$\gamma^* \leftarrow 0$
4: **While** there is a flow augmenting path tree from s to $T(s)$ in R_G:

- Find the shortest augmenting path tree SPT between s and $T(s)$ in R_G
- Find the maximum concurrent flow γ along SPT
- Broadcast SPT
- Update flows
- Update R_G using SPT and γ
- $\gamma^* \leftarrow \gamma^* + \gamma$

end while

we have multiple sources called S representing data producers and a set of destinations related to each source called $T(S)$ representing their data consumers. A tree routed on S and spanning $T(S)$ called Spanning tree represents the relation between the entities.

In Alg. 8, we present a sequential MCF algorithm proposed in [21]. It represents an extension of the Edmonds-Karp [62] algorithm that finds in each iteration a shortest path tree instead of a shortest path. Then, an optimal concurrent flow fraction γ^* is calculated and updated in each iteration.

In, Alg. 9, we propose a distributed version of the sequential MCF algorithm. Hereafter, we describe its main steps.

D.7.1 Finding single-source shortest path tree (SPT)

Definition: Single-source shortest path tree: is the problem of finding shortest path between a single source vertex and every other vertex in the graph.

Algorithm: It is based on Pregel, a Google library for asynchronous distributed

optimization. Each vertex stores a value denoting the distance from source vertex to this vertex. Value at each vertex is initialized to INF. In each superstep: i) receives messages from its neighbors with updated potential minimum distances from source vertex, ii) if minimum of these updated values is less than the current minimum distance of the vertex, value is updated and potential updates are sent to the neighbors (current value + outgoing edge weight (unit weight in our case)), iii) in first superstep, only source vertex will update its value to zero and send update messages to its neighbors, iv) algorithm terminates when no more updates.

D.7.2 Finding the concurrent flow (maximum of gamma)

Once the shortest path tree (SPT) has been determined, the concurrent **throughput** pushed through SPT is set to be the minimum over all arcs of residual capacity to cumulative demands (of edges in SPT) given by the following equation: $\gamma^{\star} = \min_{\mathrm{a}} \frac{c_a}{D_a}$ (i.e., the maximum concurrent throughput is smallest ratio which is equivalent to the concurrent throughput you can push along SPT to satisfy the cumulative demand without violating the capacity constraint).

D.7.3 Update the flows (augment/increment the flows)

Once we find the exact concurrent throughput, we update the flows by broadcasting the tree SPT and incrementing flows for every edge in the SPT in the classical way similar to the Ford-Fulkerson algorithm).

D.7.4 Update the residual graph

The residual graph is updated also in the classical way but in a distributed manner using the MapReduce facility (distributed computing of the residual capacity on each arc in the broad-casted SPT). The sub algorithm of building the residual graph is as follows:

1. If *SPT* contains edge (i,j) in R_G:

(a) Emit *((i, j): c - γ^*. D)*

(b) Emit *((i, j): γ^*. D)*

2. Else: *emit ((i,j):c)*

Then, a computation of the shortest path of the next iteration is done until there is no shortest path tree. Once the DMCF is terminated, γ^* is the exact maximum concurrent throughput we can push.

D.8 Comparison

In Table D.1, a brief comparison between our algorithm (SS-DMCF) and the relevant work in the state of the art is presented. The proposed solution outperforms the above work in three main axes:

1. Optimal and approximate solution.
2. Polynomial time solution: Since a shortest path spanning tree algorithm is used to calculate the generalized maximum flow between the single source and all the destinations, the solution is solved in polynomial time.
3. Simple and not complex: It requires only QoS metrics that indicate the flow utilization of all commodities.

D.9 Conclusion

In this chapter we have surveyed the relevant framework of the sequential and distributed maximum concurrent flow problem (MCFP). Then, a single source distributed MCF (SS-DMCF) algorithm is introduced in the context of traffic engineering, flow (demands) routing, and QoS routing in data-driven networks such as ICN/NDN. In the future work, we plan to use machine learning techniques instead of building mathematical models.

Table D.1 – Comparison between SS-DMCF and state-of-the-art

Work	*Approach*	*Metrics*	*Limitations*
Jonathan et al. [103]	Approximate MCF	Oblivious routing	Missed optimal solution
Matula et al. [53]	Exact	LP	Exponential complexity
Racke et al. [15]	TE	Congestion	Lacks flow utilization metric
SS-DMCF algorithms	Optimal	QoS metrics	No limitations

D.10 Annex A: shortest path spanning tree computation

In this step, we compute a Shortest Path Spanning Tree in synchronous network.

The requirements for this computing are:

- Connected graph $G = (V, E)$ where V is the set of vertices and E is the set of directed edges.
- $weight_{u,v}$ for edge $\{u, v\}$
- Distinguished root vertex (the single source) v_0
- Processes have no knowledge about the graph
- i_0 is the UID process of the root vertex v_0
- Processes know UIDs of their neighbors, and know which ports (interface/face) are connected to each neighbor.
- Processes must produce a Shortest Path Tree (BFS tree in terms of hop count) routed at vertex v_0
- Branches are directed paths from v_0

Algorithm 10 Find the SPT, algorithm for the process i

1: **Input:**

- ***dist:*** a nonnegative real or ∞, representing the shortest path known distance from v_0
- ***parent:*** a UID or undefined, initially undefined

2: **At each round:**

- Send a ***distance(dist)*** message to all neighbors
- Receive messages from neighbors; let $d_j be the distance received from neighbor j$.
- Perform a relaxation step:

$$dist = \min(dist, \min_j(d_j + weight_{i,j})) \tag{D.5}$$

- If dist decreases then set $parent \leftarrow j$, where j is any neighbor that produced a new dist.

  - Spanning
  - Shortest Path: the total weight of the tree branch to each node is the minimum total weight for any path from v_0 in G.

- ***Output:*** Each process $i \neq i_0$ should output ***parent(j)***, ***distance(d)***, meaning that:

  - j's vertex is the parent of i's vertex on a shortest from v_0
  - d is the total weight (distance) of a shortest path from $v_0$to j.

SPT time complexity:

- Number of rounds until all the variables stabilize to their final values.
- n-1 round, where n is the number of nodes in the network.

SPT message complexity:

- Number of messages sent by all the processes during the entire execution.
- $O(n \times |E|)$

D.11 Annex B: destination node program

Destination node sends the flow to its parents.

D.12 Annex C: intermediate node program

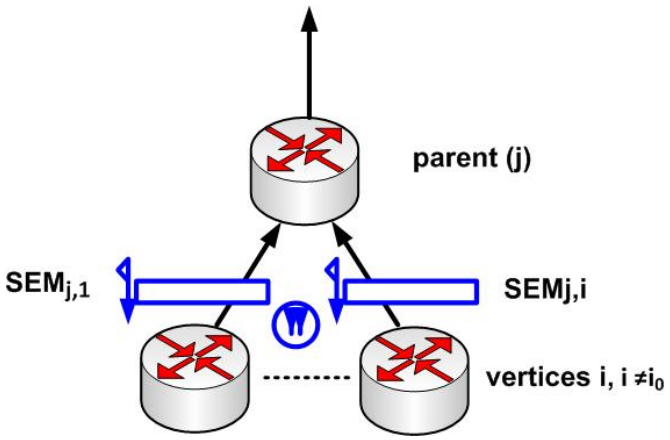


Figure D-1 – Intermediate node program

As shown in Fig. D-1, an intermediate node does the following pseudo code in order to calculate MCF in a distributed fashion:

Algorithm 11 Calculate the cumulative demand

1: void (Indegree)
2: **while** SEM $\neq Indegree$ **do**
3: $G[parent(j)][parent][D]+ \leftarrow arc[i, parent(j)]$
4: $SEM \leftarrow SEM - 1$
5: **end while**
6: $arc[parent(j)][parent][D] \leftarrow G[parent(j)][parent][D]$

Algorithm 12 Calculate the MCF γ^*

```
void (Indegree)
while SEM ≠ Indegree do
    G[parent(j)][parent][γ]+ ← C[i,parent(j)] / D[i,parent(j)]
    if minigamma ≥ G[parent(j)][parent][γ] then
        minigamma = G[parent(j)][parent][γ]
    end if
    SEM ← SEM − 1
end while
arc[parent(j)][parent][γ] ← G[parent(j)][parent][γ]
```

Algorithm 13 Calculate the residual capacities

```
void (Indegree)
while SEM ≠ Indegree do
    G[parent(j)][parent][C]− ← γ* × G[parent(j)][parent][D]
    G[parent][parent(j)][C]+ ← γ* × G[parent(j)][parent][D]
    SEM ← SEM − 1
end while
if G[parent(j)][parent][C] ≤ 0 then
    Mark link
else if ¡condition¿ then
    ¡text¿
end if
```

Appendix E

Optimal Hadoop over ICN Placement Algorithm for Networking and Distributed Computing

E.1 Introduction

Large scale distributed computing is the predictable baseline process in our big data era. The Apache Hadoop [42], a popular open-source framework for distributed storage and processing of large data sets has become the promising technology for processing multiple distributed applications such as traffic analysis, cache network update, network anomalies detection, data mining, machine learning, and bioinformatics research. The diversity of applications requires Hadoop to be more flexible, adaptive to different contexts while providing high performance, resulting in complicated networking protocol design, complex pieces of software that require non-trivial configuration and tuning for better performance [161]. A candidate architecture, ICN [94] [184] shifts the networking focus from host-centric communication to multi-source content retrieval, a better fit to big data networking/computing that TCP/IP, the Internet communication protocol. ICN can potentially provide enormous benefits in easing the deployment, improv-

ing robustness and overall performance for hadoop systems [151] [89]. Additionally, ICN offers flexible support to several applications, and ICN in-network caching/processing at intermediate nodes allows content retrieval with high responsiveness [86] [85] [61] [158].

ICN Research Group (ICNRG) within the Internet Research Task Force (IRTF) highlighted the distributed computing among the major ICN scenarios [137]. ICNRG overall requirements for networking and distributed computing (e.g., Hadoop systems) to leverage ICN are addressed in [161] and [108]. However, integrating ICN with Hadoop needs studies for Hadoop over ICN architecture.

Despite the above IRTF effort, there are not clear architecture for ICN support of Hadoop. Consequently, in this chapter we present a Hadoop over ICN architecture, considering not only architectural principles but also novel layering responsibilities of of HoICN architecture.

Still, introducing ICN into large-scale networking and distributed computing such as Hadoop will have profound impacts on the design of large data centers, distributed computing infrastructures, and hence HoICN needs studies for optimal placement of ICN nodes. Consequently, we contribute by an optimal placement algorithm for ICN/Hadoop nodes. As modern network optimization deals with complex graph topologies, we also present a Gomory-Hu algorithm that detects network bottlenecks in terms of networking and computing resources through the maximum-flow minimum-cut theorem [73] and to find different tree levels for potential upgrade in ICN/Hadoop nodes and gateways. G-H is an optimal cut-tree algorithm that compacts the larger network graph structure using cuts to retain only feasible candidate topology and consequently lead to a smaller scale ICN/Hadoop non-deterministic polynomial (NP) placement problem.

The rest of this chapter is organized as follows: Section E.2 highlights the ICN related work and its operation in the field of computation. Section E.3 designs the merging architecture between hadoop and ICN. Section E.4 formulates two optimization algorithms. Section E.5 evaluates the global optimization method.

Section E.6 concludes the work.

E.2 Related work

R. Mijumbi et al. [120] study the relation between ICN and the promising softwarization techniques (i.e., NFV and SDN). Authors highlight ICN as a candidate NFV use case that may be included as a helper within the SDN/NFV reference architecture in order to select the optimal position of placing virtual network functions (VNFs). The architecture representing this relation is interesting but too general. ICN cannot be deployed as a general VNF. It needs strong personalization and adaptation to the proposed application. In our approach ICN is considered as an intelligent data plane that assists hadoop system and it is deployed through global optimization algorithm.

Mangili et al. [117] design a novel model that takes as input a given network topology, a list of demands arriving from consumers, and a list of producers, each one of them publishes a content (content might be published by several publishers), the optimization has to choose which router should be upgraded with ICN and in each ICN router which object should be cached (among a list of available objects). Authors hence introduce a relevant work that optimizes first the upgrade (deployment) cost and second the object placement within ICN nodes. They propose an IP model that minimizes the cost of request routing, then the model is extended through additional parameters such as interest propagation cost, router upgrade cost, and the installed caching cost on the upgraded ICN router.

C. Tschudin et al. [161] investigate the relation between networking and computation and propose Named Function Networking (NFN) as an ICN extension. In their approach, consumers send named expression interests and the network is responsible for interpreting these interests, executing the related functions and caching the computed results. The approach is very interesting and it is adopted in the IETF standard [137].

Routing protocols responsible for computing routes and handling network

changes are still needed in ICN/NDN approaches for link recovery and bootstrap adaptive forwarding as mentioned in [181]. V. Lehmen et al. [174] address the routing scalability problem through proposing an hyperbolic routing (HR) strategy with a smart forwarding plane in NDN. Their strategy scales well upon update in network topology, however destination selection is sub optimal. Author hence designed a novel strategy called adaptive smoothed RTT-based forwarding (ASF) to mitigate this problem and benefit from the intelligent forwarding plane through NDN approach. In our approach we combine traffic routing and deployment through an optimization algorithm.

A. Lindgren et al. [114] recommend design choices for applying ICN to IoT with small changes to the ICN concepts. Still, supporting efficient and scalable IoT applications over existing ICN architectures is challenging given the massive number of IoT devices. In our work, we introduce Hadoop as a high level application to process massive IoT.

H. Zhang et al. [185] propose NDNFit, a novel prototype for mobile health ecosystem. It is built over NDN communication. Their proposed architecture has different components for storing, processing, and visualizing/interacting with IoT data.

K. Schneider et al. [155] propose a practical congestion control for ICN which uses RTT and packet loss as transmission control metrics. Their solution based on measuring congestion rate at each NDN router in order to keep the NDN links non-congested. This distributed solution may introduce high delays within the network nodes (jitter) which may make it unfeasible for real-time scenarios, etc.. The approach adopts a hop-by-hop congestion scheme where intermediate routers have to detect congestion and avoid it through marking some interests and sending a short feedback back to consumers and involved routers in the conversation. Consumers hence reduce sending rate of interests and routers drop interests or divert them to different paths.

E.3 Hadoop over ICN (HoICN) design

E.3.1 Information-Centric Networking

ICN is a data-centric approach introduced as a novel networking paradigm. It is based on two main messages, *interest* and *data* [94]. In the former message, the name of the data is defined. The latter message binds the name to the raw data. It is a receiver-driven approach that focuses on information instead of nodes and where data transmission is triggered after a consumer interest.

The main key functionalities of ICN are request routing, caching and processing, mobility, and security. Indeed, ICN generates an optimal path toward the content through lookup and longest-prefix matching algorithms. Moreover, popular data are cached and processed on the fly in the intermediate routers in order to reduce the end-to-end consumer delay. Through ICN paradigm, caching solutions are on-path caching and off-path caching [184]. ICN supports also different mobility management solutions [153]. Indeed, subscriber mobility is resolved by re-issuing interest of the same content, publisher mobility which is more difficult is resolved through interest tracing and chasing [189]. Security is also inherent in ICN architecture and assures data integrity and verification by binding name to content [183].

E.3.2 Principle Hadoop components

Hadoop is a big data framework that eases data processing, analysis, and mining in different disciplines such as information fetching, neural science and deep learning. It consists of two main components: Hadoop Distributed File System (HDFS) and MapReduce. The former is responsible for data storage and presentation (i.e., a storage layer) while the latter is responsible for parallel processing and distributed computation (i.e., a compute layer).

From a design perspective, Hadoop is a decentralized (hybrid) architecture where each component (HDFS or MapReduce) is centralized and controls dis-

tributed nodes. For instance, HDFS layer is controlled by name nodes and is distributed through different data nodes (caches). On the other hand, MapReduce layer is controlled by Job Tracker and is distributed through different Task trackers (processors). The central/distributed (hybrid) architecture nature of hadoop may benefit from ICN networking architecture [108] that provides a distributed cache network able to process and cache on the fly massive data.

HDFS and MapReduce layers cooperate together to allow storage and processing functionalities within the same system. MapReduce component processes and analyzes data traffic from different locations including network nodes, ICN cache nodes and buffers (memory). It is also adapted for quantifying/analyzing network behavior and detecting anomalies. MapReduce provides then the capability to run distributed computations and generate data sets needed for traffic analysis through analytic tools. For the sake of simplicity, in the proposed architecture, MapReduce aggregates data in the distributed ICN-enabled data nodes. Then, global data sets are generated and used by different tools to extract the requested aggregated information such as medical reports in a e-health context or environmental indicators in weather forecast one, etc.

E.3.3 HoICN node architecture

In general, HoICN will be useful when the big data request involves information available in the network (traffic analysis, IoT information, Video...). This network data is directly treated and mapped onto the data mining format (Hadoop in our case). HoICN approach is based on two ICN packet types: hadoop interest packet corresponding to a consumer group request for a computation result (instead of data in the legacy ICN paradigm). Hadoop interest is a simple mathematical or statistical named function (e.g., max, min, average, etc...). The second HoICN packet type is a hadoop data packet corresponding to the computation result of executing this function on local data. Hadoop related functions are moved toward the data instead of data moving in order to save network bandwidth and

storage. ICN is used to compute, and to cache the computation result within the network. Hadoop is data agnostic and it can interact with heterogeneous data collectors (e.g., IoT data, call data record (CDR), etc...). For instance, it can process massive IoT data on the fly (in-network processing) instead of migrating data toward the centralized/consolidated cloud controllers (NFV/SDN). ICN node support of hadoop interest/data exchanged messages includes an extra data structure beside the legacy tables already proposed in the IRTF standard such as FIB, CS, and PIT. It is called Hadoop Information Table (HIT).

E.3.4 HoICN layer responsibilities

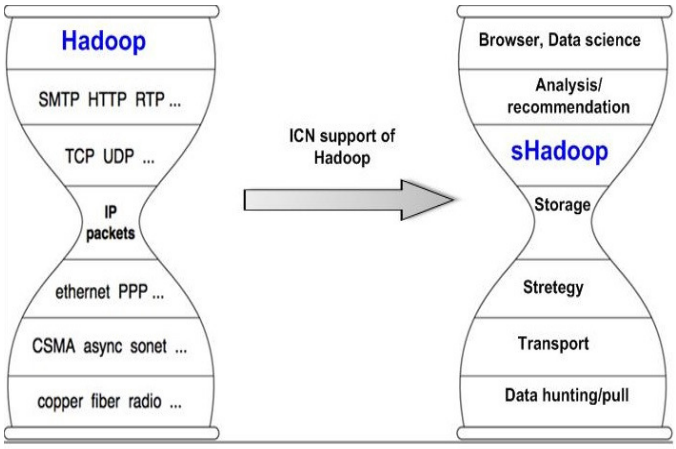


Figure E-1 – HoICN layer responsibilities

Following ICN standards, we believe that interpreting hadoop packets in the networking level instead of the application level requires some modifications to the legacy ICN paradigm. Indeed, as shown in Fig. E-1, Hadoop over IP (left side of the figure) runs on the application level. The amended layering is shown in the right side of the figure. It should achieve the following principle responsibilities:

1. Data hunting/pull layer: HoICN is a pull-based approach where the consumer sends interest and the the network interprets it and fulfills its request through ICN data packet. ICN-enabled routers capture Hadoop interests

(named functions) from consumer groups. The network then executes the function and sends back the result.

2. Transport layer: ICN is used between named data nodes for bulk data transfer using hadoop interest and data packets. This layer is then will responsible for the flow control between interests and data.

3. Strategy layer: The forwarding strategy of hadoop interest and data is as following: i) ICN router does a longest-prefix match lookup on the HIT table indexes, ii) If there is a computation result information (CRI) matches the hadoop interest name, then the named function is already executed and the result will be forwarded back to the consumer. Otherwise, the router binds the CRI name to the requesting face in PIT data structure. Then, the ICN router forwards the hadoop interest to where data reside.

4. Storage: The network is in charge of caching the processed data results in different ICN nodes.

5. Secure Hadoop (sHadoop): using ICN as hadoop distributed storage and processing will benefit from data integrity inherent is ICN.

6. Analysis and recommendation: This layer is responsible for data analysis (through hadoop tools). It proposes some recommendations to end users.

7. Browser and data science: This is the consumer application that injects hadoop interest within the ICN network. Extracting knowledge (data science) from data may enhance response feedback through recommendations given from the network.

E.3.5 Massive IoT data as a use case

Processing massive IoT data constitutes a big data network. Therefore, it is a potential use case for Hadoop. In Fig. E-2, we depict a merging network topol-

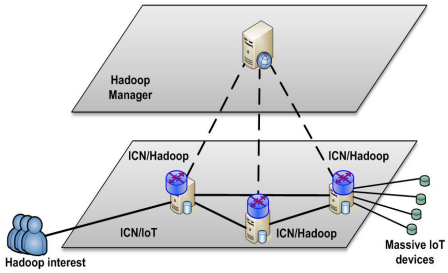


Figure E-2 – HoICN topology: Massive IoT data

ogy between hadoop and IoT and where to deploy ICN nodes acting as cache or processors nodes. Hadoop data node are distributed and fledged to the ICN.

E.4 HoICN: optimization algorithm

Although ICN may be used for caching and processing IoT data through a simple linkage to Hadoop, ICN standards do not specify how these nodes can be deployed in a large scale network. Implementing ICN-Hadoop nodes everywhere in the underlying topology induces high cost and is not efficient (only some intermediate nodes could host a treatment service). We believe that Hadoop will be deployed as a virtual service on behalf of content providers in the operators network. In this context, we try to propose a HoICN placement algorithm to solve this issue. Given a large network infrastructure, HOPA will solve this deployment problem and will give ICN instantiation graph for new ICN-Hadoop positions. Given an initial network topology, algorithm inputs are: massive IoT nodes, IoT gateways corresponding to all ICN-Hadoop nodes, and consumers requesting computation results related to IoT (e.g., average temperature in a year).

The proposed algorithm selects where to cache the Hadoop computation results based on: system parameters (intermediate node capacity of CPU and memory), network parameters (the network bandwidth), and quality of experience parameter (end user required quality). Hereafter, we present optimal and heuristic optimization approaches.

Table E.1 – HoICN Mathematical Notation

Parameters	***Definition***
V	The set of consumers
S	The set of server nodes
D^s	Maximum network capacity of the server $s \in S$
F	The set of ICN-Hadoop data nodes
G	The set of named functions
$g(f)_{size}$	The size of the computation result in terms of memory ($f \in F$, $g \in G$)
C^s	The maximum memory capacity of the server s
$L_{i,j}$	The direct link capacity from node i to node j
$d_v^{g(f)}$	The set of consumer group hadoop interests
$p^s_{g(f)}$	The placement cost of $g(f)$ on s
Decision variables	***Definition***
$x^s_{g(f)}$	Placement binary variable which indicates that the ICN-Hadoop computation result ($g(f)$) should be placed on the (optimal) server $s \in S$
$y^s_{v,g(f)}$	Mapping binary variable which indicates that consumer group ($v \in V$) needs an ICN-Hadoop computation result which should be placed on the optimal server $s \in S$
$z^{v,g(f)}_{i,j}$	Flow balance binary variable which indicates whether the link (i, j) is used for satisfying consumer interest. The consumer is interesting in g(f)

E.4.1 HOPA: HoICN Optimal Placement Algorithm

In Table E.1, we show the proposed parameters (system, network and quality) and decision variables (placement, routing, and flow balance) of our optimization (exact and heuristic) algorithms.

The description of our decision variables are as follows:

1. The binary variable x indicates the placement of ICN node, and the migration of the computation result from one server to optimal location s. It is defined as:

$$x^s_{g(f)} = \begin{cases} 1, & \text{if } g(f) \text{ migrates to } s \\ 0, & \text{Otherwise} \end{cases} \tag{E.1}$$

2. The binary variable y indicates that the consumer needs a computation result, and the server s caches it. It is defined as :

$$y^{s}_{v,g(f)} = \begin{cases} 1 & \text{if } v \text{ needs } g(f) \text{ and } s \text{ caches } g(f) \\ 0 & \text{Otherwise} \end{cases} \tag{E.2}$$

3. The binary variable $z^{v,g(f)}_{i,j}$ indicates whether a link (i, j) is used (from i to j) to route g(f) from a replica server (the one for which $y^{s}_{v,g(f)} = 1$) to consumer v. It is defined as:

$$z^{v,g(f)}_{i,j} = \begin{cases} 1, & \text{if there is a path } P_{g(f),v} \text{ using link } (i, j) \\ 0, & \text{Otherwise} \end{cases} \tag{E.3}$$

The optimization model of the optimal approach is quoted as follows:

1. The HoICN objective function is to minimize the total placement (deployment) cost. It is modeled as:

$$\forall g \in G : F = \sum_{s \in S} \sum_{f \in F} x^{s}_{g(f)} \times p^{s}_{g(f)} \tag{E.4}$$

2. Consumer group interesting in a hadoop result where $d^{g(f)}_{v}$ not null retrieves the result from one optimal server:

$$\forall v \in V \mid d^{g(f)}_{v} \neq 0 : \sum_{s \in S} y^{s}_{v,g(f)} = 1 \tag{E.5}$$

3. Satisfying the interests in cached computation results concurrently is limited by the network capacity of the optimal server as follows:

$$\forall s \in S : \sum_{v \in V} \sum_{f \in F} y^{s}_{v,g(f)} \times d^{g(f)}_{v} \leq D^{s} \tag{E.6}$$

4. The system constraint that assures that cached computation results is limited by the server system capacity is:

$$\forall g \in G, \forall s \in S : \sum_{f \in F} x^{s}_{g(f)} \times g(f)_{size} \leq C^{s} \tag{E.7}$$

5. The flow balance constraint guarantees that incoming flow (some computation results) equals to the out-coming flow except the source and the destination. In the source, there is only out-coming flows. In the sink, there is only incoming flows:

$$\forall g \in G : \sum_{j} z^{v,g(f)}_{i,j} - \sum_{j} z^{v,g(f)}_{j,i} = \begin{cases} 0 & \text{if } i \notin \{v, s\} \\ y^{s}_{v,g(f)} & \text{if } i = s \\ -1 & \text{if } i = v \end{cases} \tag{E.8}$$

6. The capacity constraint assuring that each network link is bounded by its capacity is modeled as follows:

$$\forall g \in G, \forall i, j \in V \cup S : \sum_{v \in V} \sum_{f \in F} z^{v,g(f)}_{i,j} \times d^{g(f)}_{v} \leq L_{i,j} \tag{E.9}$$

E.4.2 HHPA: HoICN Heuristic Placement Algorithm

HHPA selects near optimal nodes to act as intermediate ICN servers. Those nodes interpret future user interests for the same computation. To do this, we transform the initial graph (representing the larger network) to a flow-equivalent tree where each edge weight represents the maximum amount of flow. Gomory-Hu [73] as a powerful multi-terminal maximum-flow algorithm is scalable and can be used for that purpose. Then, supposing that Hadoop nodes are executed on the IoT gateways (in order to collect and process the massive IoT data), we migrate the computation results (instead of the raw data) from the original processing node towards consumers in the forms of ICN data packets. We verify in this process

(a) Small-world based network topology (b) Graph transformation

Figure E-3 – Network topology used for large scale

that system, network, and quality constraints are satisfied.

E.5 HoICN: performance evaluation

In this section we evaluate our optimization algorithm. The proposed network topology follows Watts-Strogatz[1] that is highlighted in the IETF/IRTF standard [138]. We believe that it may fit with Hadoop clusterization and distribution nature where all the nodes are close to each other in terms of network distance. In Fig. E-3a, we show the initial topology used for the evaluation. It is undirected and weighted graph that has 100 vertices representing network nodes and 200 edges. The transformed graph based on Gomory-Hu method is shown in Fig. E-3b which has only 99 edges (49.5%). These figures depict the topology used for evaluating HoCCN placement algorithm in a large scale scenario.

To assess HoICN optimization algorithm, we used the runtime and the consumer delay metric. It represents the response time when using ICN facility to cache the computation result instead of the legacy Hadoop process where intermediate network nodes do not interpret consumer interests. $sg(f)$ represents the server that initially executed the code on the local data and then it cached the computation result. s_v represents the access point toward the consumer group. This

[1] Watts-Strogatz small world models that target large network size and follows a binomial (or a Poisson) degree distribution.

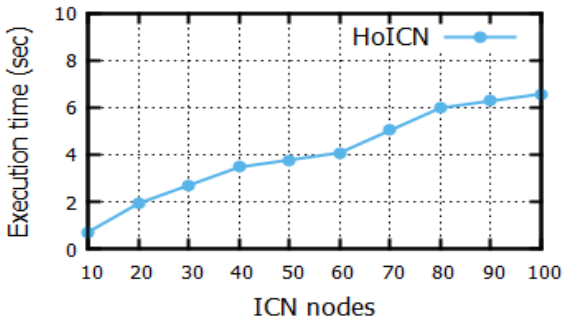


Figure E-4 – HHPA: execution runtime

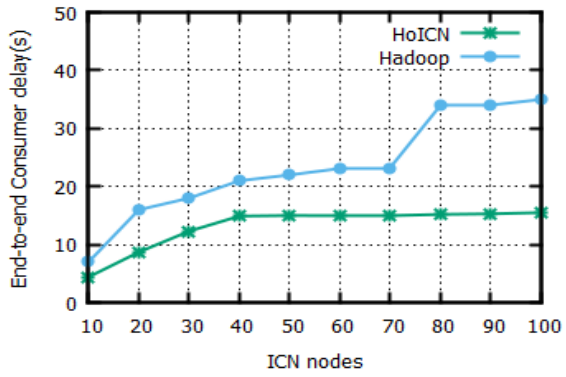


Figure E-5 – HHPA: end-to-end consumer delay

metric is modeled as follows:

$$HoICN\ delay = \sum_{c \in C, f \in F} \frac{d(s_v, s_{g(f)}) \times g(f)_{size}}{\max_{(i,j) \in P_{s_{g(f)}, s_v}} (BW_{i,j})} \tag{E.10}$$

Fig. E-5 depicts the execution time of HoICN approach. Result demonstrates the feasibility of the optimization approach in large network scale since that the run-time is in terms of a few seconds (7 sec) for large ICN instances.

Fig. E-4 shows the end-to-end consumer delay needed for retrieval a computation result. Result shows that HoICN method reduce the total delay. This is due to ICN deployment facility placed between the producer and the consumer. The legacy Hadoop network which does not imply this feature suffers from significant consumer delay comparing to our HoICN.

E.6 Conclusion

We proposed in this chapter to use ICN for Hadoop-based systems and we map Hadoop functions onto ICN routing nodes. In this context, we presented two optimization algorithms (HOPA and HHPA) to deal with small and large network scales respectively. Experiments demonstrate that the integration of Hadoop with ICN benefits from the distributed nature of both concepts. Results show that the ICN support of Hadoop is feasible for large network scale and the end-to-end consumer delay is reduced. We are working on getting this architecture and optimization running on our virtual placement platform and outsourced in [165] [2].

As presented in this chapter, hadoop can benefit from ICN distributed file layer to analyze and mine the global knowledge and behavior of the IoT network. In the next chapter, we detail the IoT use case and propose a network optimization tool.

[2] Vios platform enables the placement and migration of VNF functions such as CDNs and ICN

Appendix F

Optimal Placement Algorithm (OPA) for IoT over ICN

F.1 Introduction

ICN is a candidate architecture that eases the deployment of massive IoT. IoT over ICN can provide a multi-point to multi-point communication model that helps achieving multi-source data retrieval with less overhead. Additionally, ICN naming provides flexible support to several services over the same IoT network, and ICN in-network caching at intermediate nodes allows IoT content retrieval with high responsiveness.

ICN Research Group (ICNRG) within the Internet Research Task Force (IRTF) highlighted IoT among the major ICN scenarios [137]. ICNRG overall requirements for IoT to leverage ICN are addressed in [108] and [188]. However, introducing ICN in IoT needs studies for optimal placement of ICN nodes while vouching the security of these intermediate ICN/IoT containers.

Despite the above mentioned IRTF efforts, there are not clear contributions on IoT over ICN infrastructure optimizing ICN nodes placement. Consequently, in this chapter we present a maximum flow min cut algorithm (the Optimal Placement Algorithm) for ICN/IoT nodes, considering heterogeneous performance pa-

rameters not only covering network issues but also limited system resources.

As network optimization deals with very large graph topologies, classical exact optimization techniques do not fit our problem. Instead, we use the well known Gomory-Hu (G-H) method [73] to detect the network bottlenecks through the maximum-flow minimum-cut theorem [62] and to find different tree levels for potential upgrade in ICN/IoT nodes and gateways. G-H is an optimal network flow algorithm that compacts the network graph structure using cuts to retain only feasible candidate topology and consequently lead to a smaller scale ICN/IoT placement problem. Although each ICN/IoT node is secured since its production, we present some security considerations to emphasize the reduction in network security cost while enabling ICN for IoT with different security levels.

The rest of this chapter is organized as follows: Section F.2 highlights ICN and its operation in IoT and the related work. Section F.3 presents our Optimal Placement Algorithm, (OPA) for ICN/IoT network architecture. Section F.4 discusses some security considerations and Section F.5 evaluates the algorithm performance over large graph topology. In Section F.6, we compare OPA in ICN/IoT with a traditional IoT network deployment. We conclude the chapter and present our future work in Section F.7.

F.2 ICN in IoT and related work

F.2.1 ICN principles

ICN names the content rather than the host in the networking level. Different ICN architectures are proposed such as: network of information (NetInf) [126], named data networking (NDN) [125], content centric networking (CCN) [32], data oriented network (DONA), and publish and subscribe information protocol (PSIRP) [142]. Most of these information-centric network architectures are implemented on top of TCP/UDP/IP/P2P layer. All of them, are originally inspired from the early work of van Jacobson [93] who introduced the baselines

and the fundamental features of the ICN architecture (e.g., node model, naming, routing, transport, caching, etc.) and the strategy layer for the adaptive forwarding [94]. In ICN, names are hierarchical and similar to URLs. Name Resolution Service (NRS) and data routing procedures are in general integrated or coupled. The exchanged messages between consumers "data requestors" and producers "data original providers" are respectively in the form of *interest* packets and *data* packets. Indeed, end-users express only *what* they want (content name), and they let the ICN network respond to the *where* and *how* the content will be retrieved. Therefore, an appropriate ICN topology is an optimization per se. ICN network consists of consumers that request the content, producers that originally provide/publish these content, and ICN routers that cache/treat the content. The content router (CR) in ICN has three main data structures as follows: i) Forwarding Information Base (FIB) table: it binds the content name to the next hop as in IP layer that binds the IP prefix to the destination, ii) Pending Interest Table (PIT): it binds the content name of the unsatisfied requests to the requesting face, and iii) Content Store (CS) table: it binds the content name to the data per se.

Caching in ICN usually implies the on-path caching and CSs use by default the Least Recently Used (LRU) replacement policy. Off-path is also supported by redirecting user interest to a Content Delivery Network (CDN) (as an example) and not to the source/publisher of the content. ICN needs new mobility management solutions different from host-centric approaches. The $Kite$ model [189] proposed by UCLA presents a novel solution for data producer mobility, leveraging the state of PIT table on each ICN router to reach Mobile Nodes (MNs). $Kite$ supports different mobile application scenarios such as push, pull, share and upload. Firstly, a correspondent node contacts an anchor node, and each time it requests published data, interests follow the PIT table to contact the anchor node, and then follow traced interests to reach the mobile node. The approach is relevant and requires interest packets targeting a mobile producer to always pass by anchor (which can present a single point of failure). Moreover, authors design how to support the data pro-

ducer mobility (publishers) in ICN network, through letting the consumers fetch the produced data easily after producer mobility. A survey of different solutions for producer mobility is also found in [186].

F.2.2 Why ICN for IoT

In this chapter we recommend leveraging ICN as a forwarding plane for IoT. ICN features provide several advantages in IoT deployment as listed below:

- Content naming: most of IoT applications do not require establishing an end-to-end communication between two devices, however rely on retrieving content (regardless its location). ICN content naming simplifies content retrieval by different IoT service providers representing multi-tenancy.
- Data-centricity: IoT consumers requesting (temperature measurement, etc..) are not looking for the end host device but look for the data itself.
- In-network caching: ICN routers provide data from content store even regardless of the reachability to the data source, which reduces the latency and signaling to reach the original data source. Furthermore, by caching massive IoT data, composed services and data treatment can be extensively improved (efficient data analytics at the network edge).
- Data Authenticity: ICN provides object-based encryption, and names are used to verify content integrity and authenticity through binding name to data and its publisher or origin.

Although the advantages mentioned above, there are still some open issues for ICN over IoT as follows:

- Naming: Device Naming in some IoT applications is needed in different cases (e.g., actuation command, switch on/off a device, etc.). ICN caching may have a counter effect, and action may be delayed. Moreover, in video data ”coming from IoT cameras” scenario, it is still unclear how ICN naming will work. Handling delay sensitive application is also another scenario.

- Data size: Size of data in IoT is generally very small (a few bytes such as 1-on, 0-off), where the size of the ”data name” may exceed the size of the ”data” (i.e., memory issue). Naming overhead also appears in the case of hierarchical names that can be too long. Also, dynamic data generation of IoT is a challenge for hash-based content names.

- Data caching: Caching IoT data needs an adequate caching policy. Moreover, decoupling publisher/consumer is an advantage for content retrieval but it needs full name deduction from consumer.

We address some of the above mentioned issues in our solution, especially what concern the overall network performance (IoT data size, caching, etc). Still, IoT over ICN needs some requirements to be enough beneficial. Having an optimal placement for ICN/IoT nodes is a very important problem that is not yet solved. Additionally, the following design factors need consideration in IoT over ICN architectures (note that our proposed solution considers these design factors):

- Network transport model: a combination of PULL (receiver-driven) and PUSH (sender-driven) models can be supported. Nevertheless, the base model design with reference to ICN standard should be PULL. Unlike TCP/IP, a design decision of IoT over ICN is to follow the barlund's transactional model of communication [123], which models ICN active network where IoT producers, ICN nodes and consumers interact with data through feedbacks.

- Sequence numbers: Immutable data objects with sequence numbers are needed to support large scale distribution and valid data in ICN domains (as in dynamic databases).

- From the hardware perspective: ICN nodes and IoT device are not fully fledged in order to let IoT devices (highly mobile, low duty cycle) delegate the responsibility of responding to requests to more stable ICN node connections. These delegated nodes can be IoT Gateways (and not tiny IoT de-

vices) instead of having them as separate nodes that are not part of the IoT world (they can still be IoT nodes but functioning as IoT Gateways. Moreover, resource constrained devices should not serve as data producers or ICN gateways (capability advertisements).

- Actuation: handling actuators in ICN network should be controlled through the ICN communication model without the ICN caching property. Enabling caching for actuation scenarios make data consumers unsure that the request reached the actuator and can lead to delay in the actuation process.

- Node architecture: ICN node should be extended with explicit support for time and data freshness. IoT nodes may periodically update their information/state.

- Security: it should be object-based rather than session-based. Instead of securing the communication pipe (the channel) such as the case in TLS, SSH, SSL, securing the content state in the network using advanced cryptographic tools is the adequate approach. However the big challenge that is not yet resolved is how to share and distribute the right cryptographic keys to the right nodes.

F.2.3 ICN in IoT: related Work

In-network caching has been widely investigated in ICN/NDN wired networks, while few contribution addressed NDN caching for wireless and IoT networks. The work in [184] presents a survey on the current approaches, which are usually considering fixed and wired connected content routers and also designed for multimedia content distribution. [141] and [167] discuss probabilistic caching strategy with the key idea that each NDN node randomly caches incoming Data with a certain probability p, where $0 < p < 1$. The (CE^2) strategy is a special case of the probabilistic scheme, where ($p = 1$). Decreasing p reduces the in-network caching probability and maximizes the diversity of Data cached in the network. How-

ever, [141] considers fixed networks with wired connected routers and [167] adds large amount of overhead and complexity to achieve performance, which is unsuitable for resource constrained devices in IoT. Only few work consider caching in IoT systems ([16], [144], [175]). The work in [16], evaluates the performance of content retrieval from different consumers with standard NDN in-network caching, however, the cache size of resource-constrained nodes (used in the experiments) is 1 Kbyte and information is ephemeral (short-lived, transient). The work in [144] analyzes the impact of IoT information freshness over NDN caching through using a consumer driven freshness approach besides the freshness parameter included in Data packets (establishing how many seconds the content can be valid in the CS). This improves the accuracy of the data received by consumers. The challenge here is how this caching approach sustains in the presence of big number of consumers with different freshness requirements. The work in [175] presents a first study on caching IoT content in Internet wired content routers (electrically powered static routers), proposing a distributed probabilistic caching algorithm where routers dynamically update their caching probability by considering their hop distances to the source and the consumers and the data freshness (i.e. the closer the caching location is to the source, the fresher the retrieved data packet is). This approach may not be always valid and depends on the type of data, sensor/IoT type emission rate. Unlike [175], wireless NDN-IoT multi-hop network composed of (mobile) resource constrained nodes is considered in [77]. A probabilistic caching strategy is proposed that considers the data freshness and the potential constrained capabilities of devices (mainly energy level and storage capacity).

The work in [91] considers traffic optimization through caching less popular content on content routers near to the origin site and caching more popular content on content routers near to the users aiming to minimize the sum of cache allocation power and traffic transmission power. This approach may not be so suitable if IoT content are consumed near the origin site. The work in [90] designs content locations in order of descending popularity to minimize total power

consumption. The proposed algorithm derives the optimal cache locations of a content item based on a 0-1 ILP (Integer Linear Programming). Any solver for large-scale ILP problems at high speed can be used to solve the 0-1 ILP model. The proposed algorithm starts by (1) getting content popularity and (2) selecting a target content in the order of popularity then (3) a route candidate is extracted to deliver the target content from a cache location to a requesting user on the shortest path tree rooted at the origin site of the target content and defining the route candidates as the design variables then (4) design the optimal cache locations for the target content based on the route candidate in consideration of the pre-designed routes for more popular content having the same origin site as that of the target content then (5) go to step (2) and execute the design of the next target content.

The work in [37] investigate the minimum energy consumption that CCN can achieve with optimal cache locations by considering different caching hardware technologies, number of download per hours and content popularity. The authors firstly set up an energy consumption model for CCN and then formulate linear and non-linear programming problems to minimize total energy consumption of CCN. Also a heuristic approach through a Genetic Algorithm (GA) is proposed to find energy efficient cache locations. Using reported energy efficiency of computational hardware and network equipment, the chapter shows that CCN yields greater energy savings for very popular content and small sized catalog compared to conventional Content Delivery Network (CDN). The results also indicate that two aspects of the memory technology, energy-proportional caching and sufficient memory capacity are critical to the overall energy gain of NDN. This chapter focuses only on energy efficiency regarding data delivery and storage and do not consider energy consumption caused by other computations (e.g. name lookup and security). In a similar context, the work in [75] examines energy consumption of content delivery architectures comparing the benefits of CCN and optical bypass. The chapter compares Content Delivery Network (CDN) versus CCN in content delivery efficiency. The authors show that by optimizing the content placement

according to its popularity, CCN achieve good scalability in energy consumption in delivering popular content (i.e. the per-bit energy decreases as the download rate increases). On the other hand, dynamic optical bypass is more efficient in serving less popular content. In addition, equipment energy efficiency and network topology are two other factors impacting energy consumption in CCN. The results show that CCN consumes less energy in delivering a small sized catalog while conventional based CDNs consumes less energy in delivering a large sized catalog. Consequently, it is suggested that a synergy of CCN, server-based CDN and dynamic optical bypass architectures may improve the energy efficiency in serving content with heterogeneous popularity.

The work in [29] presents energy efficiency issues in ICN showing how in network caching in ICN raises energy consumption concerns and showing the importance of quantifying the energy consumption of transmission links plus storage to best design the cache replication strategy in ICN. What's important to consider also are the frequency of content request and the content popularity. The chapter gives an overview on the energy consumption of different memory technologies, sizes, and access techniques. The chapter also discusses the impact of the layer of caching in the network on the consumption and the tradeoffs between energy-efficiency and performance.

F.3 OPA: Optimal Placement Algorithm for ICN/IoT nodes

ICN does not specify how nodes can be deployed in a large scale network. One could simply say that it will be implemented everywhere in the underlying topology, but this is not efficient and can induce high cost (not all the network nodes could host an efficient caching or intermediate treatment service). Our algorithm, (Optimal Placement Algorithm, OPA), given a network infrastructure, will solve this deployment problem and will give ICN instantiation graph for new ICN positions. It takes as input, the global network topology consisting of :

- IoT group producer nodes "sensors, cameras, etc."
- IoT gateways "aggregation hubs, routers" corresponding to all network elements including larger Internet.
- and consumers "applications, servers in data centers"

The proposed algorithm finds the optimal deployment strategy for ICN/IoT nodes functionalities based on the following parameters:

- The required consumer end to end response time.
- The node system performance. It is the system overhead (memory and CPU resource) after deploying ICN functionality in the candidate nodes.
- migration cost (network optimization), which represents the total cost of moving ICN/IoT functionality in terms of network bandwidth.

Hereafter, we first state the system hypothesies and then present the placement models based on exact Integer Linear Programming (ILP) and heuristic graph theory optimization.

F.3.1 The placement algorithm

We consider two main hypothesis in our approach:

- Consumer groups (clients) that have no direct connectivity to the IoT devices (they connect to the IoT devices through a gateway).
- Refreshing periodicity between IoT-gateways and devices (to collect IoT data) and that is smaller than the OPA evaluation periodicity.

Table F.1 defines the main system/network parameters and decision variables.

Exact ILP Solution The general formulation of the exact algorithm is as follows:

$$\min \sum_{s \in S} \sum_{f \in F} x_f^s \times p_f^s \tag{F.1}$$

Table F.1 – Mathematical Notation

Parameters	***Definition***
V	Data consumers (e.g., IoT-enabled vehicle)
S	Data producers (e.g., mobile edge cloud)
D^s	Maximum network capacity of the server $s \in S$
F	The set of ICN/IoT nodes or containers
f_{size}	ICN/IoT container's size in terms of memory ($f \in F$)
C^s	Maximum memory capacity of the server s
$L_{i,j}$	Link capacity between two nodes i and j (from i to j)
d_v^f	The set of consumer group's interests
p_f^s	The placement cost of f on s
Decision variables	***Definition***
x_f^s	Placement binary variable which indicates that the ICN/IoT ($f \in F$) should be placed on the (optimal) server $s \in S$
$y_{v,f}^s$	Mapping binary variable which indicates that consumer group ($v \in V$) needs an ICN/IoT container ($f \in F$) and ICN/IoT is placed on the server $s \in S$
$z_{i,j}^{v,f}$	Flow balance binary variable which indicates whether the link (i, j) is used for sending IoT data f to v

Subject to

$$\forall s \in S : y_{v,f}^s \leq x_f^s \tag{F.2}$$

$$\forall v \in V \mid d_v^f \neq 0 : \sum_{s \in S} y_{v,f}^s = 1 \tag{F.3}$$

$$\forall s \in S : \sum_{v \in V} \sum_{f \in F} y_{v,f}^s \times d_v^f \leq D^s \tag{F.4}$$

$$\forall s \in S : \sum_{f \in F} x_f^s \times f_{size} \leq C^s \tag{F.5}$$

$$\sum_j z_{i,j}^{v,f} - \sum_j z_{j,i}^{v,f} = \begin{cases} 0 & \text{if } i \neq v, i \neq s \\ y_{v,f}^s & \text{if } i = s \\ -1 & \text{if } i = v \end{cases} \tag{F.6}$$

$$\forall i, j \in V \cup S : \sum_{v \in V} \sum_{f \in F} z_{i,j}^{v,f} \times d_v^f \leq L_{i,j} \tag{F.7}$$

The ICN instantiation graph results from the optimization process. It is applied on an input network graph (it can be considered as the larger Internet). After optimization, a set of nodes will host the ICN function. They are identified by a binary variable x_f^s (equals 1 if the node can be upgrade with ICN function and 0 otherwise).

When consumer v sends an interest message for a given ICN data, the request variable $y_{v,f}^s$ is equal to 1 when data is available in the node s and 0 otherwise.

Finally, if a link (i, j) is used in the instantiation graph, the binary variable $z_{i,j}^{v,f}$ will be equal to equal to 0 otherwise.

In eq. (F.1), we formulate the objective function that minimizes the total placement cost of ICN nodes in the IoT network.

In eq. (F.2), we ensure that the binary variable y is less than or equal x. In fact, y equals to 1, if and only if v needs f, and f is located on server s (we should not place an ICN function on node s if there is no interest for it).

Eq. (F.3) states that the optimal server s can serve the consumer nodes interested in the ICN data f. The sum prevents consumers from having to chose between different servers hosting the same ICN data.

Eq. (F.4) enforces network constraints. We cannot exceed the maximum downloading capacity.

Eq. (F.5), is relative to node system performance. It enforces the system caching

feature of ICN nodes that should not exceed a maximum size.

Eq. (F.6) represents the network flow conservation between the intermediate ICN nodes and the consumers. In particular, if a node is upgraded to ICN, we ensure that flow balance equals to 1, meaning that it directly serves the incoming consumer interests. Otherwise, if the node is not upgraded, the flow balance is null. At the consumer side, there is no outgoing traffic (left sum is null). Hence the flow balance is negative.

In eq. (F.7), we ensure that the link capacity between network nodes should not exceed the available network bandwidth.

The above problem is NP-hard due to our combinatorial complex system and difficult to scale up. It can however be easily run on the *CPLEX* environment. As we target very large ICN infrastructures, a graph based optimization algorithm has to be designed. We present hereafter, another scalable placement algorithm.

OPA Heuristic Algorithm OPA is based on the well know Gomory-Hu scalable algorithm and it aims at placing ICN/IoT software with the same above mentioned strategy and parameters. Gomory-Hu is an off-line optimization step that compacts the the larger network topology to construct a tree with maximum-flow between all pairs of nodes. Algorithm 14 summarizes the pseudo code of OPA. Hereafter, we describe these main stages.

OPA is based on a Gomory-Hu transformation of the initial graph $G = (V(G), E(G))$ where V are the set of vertices and E are the set of edges. Vertices represent the network servers and edges represent the relation between vertices.

Algorithm 14 OPA: Optimal Placement Algorithm for ICN/IoT

1: **Input:** V, S, D^s, F, f_{size}, C^s, $L_{i,j}$, d_v^f, p_f^s, $G = (V(G), E(G))$,
2: s_v, s_f
3: **Output:** x_f^s, total ICN placement cost
4: $CTC \leftarrow$ Cut-Tree-Construction $\left(G, L_{i,j}\right)$
5: Upgrading-ICN/IoT-software$\left(\ \right)$
6: ICN/IoT-Caching$\left(\ \right)$

The initial network topology is supposed to be a scale-free network (the degree distribution follows power law). The output of this transformation is a cut-tree construction (Gomory-Hu tree, CTC) that represents the maximum-flow between all network server pairs. The cut-tree is used for bottleneck detection.

The model relies on the same information and parameters as in the exact ILP solution (number of servers, system capacity, network capacity, consumer interests). Then, an initial graph that holds the full parameters is created. Further, a Cut-Tree is constructed based on the topology capacity without including the consumer interests. The set of consumer interests is then passed on the tree. During this step, OPA algorithm explores the cut tree creating a path from the gateway to the server hosting the ICN/IoT. If the flow cannot reach the destination (i.e. the original server), caused by shortage in bandwidth on this path, we simply place the new ICN function before the rupture node of the cut tree. A test on system capacity is also performed to ensure that the target ICN node can host this new service. Although a path may not be obtained from the first trial, the problem is still polynomial compared to the NP-hard complexity of the exact solution. Finally, we would like to highlight that OPA was integrated in our virtual migration platform and outsourced in [165]. The Vios platform enables the placement and migration on NFV functions such as CDNs and ICN functions.

F.4 Security considerations

OPA places ICN/IoT nodes closer to consumers. Hence, the security of the IoT will be improved. Indeed, thanks to the ICN features, IoT data integrity is ensured through embedded encryption. We also minimize the network distance between the origin data producers (IoT devices) and consumers (that typically uses a LORA like protocol). If we assume that $Pr(attack) \propto N$ (N is the number of nodes from consumers to producers), in ICN, the $Pr(attack) < N$ A network security cost is

proposed as the following:

$$Network\ Security\ Cost = \alpha \times d(Pr, C) \tag{F.8}$$

Where α characterizes the node stability, Pr is the ICN/IoT gateway (or an intermediate ICN/IoT container after using OPA), C is the consumer group, and finally d is the network distance between Pr and C.

Finally, the security level may be introduced in our initial objective function (F.1) as an additional constraint.

OPA improves three security issues:

- ICN Caching: OPA acts as a cache relay benefiting from ICN security. Although Intermediate caching nodes may be untrusted, still the ICN infrastructure guarantees the trust for IoT data.

- Processing: OPA enables data analytics, treatment and processing of the cached IoT data. Cached data is treated by intermediate ICN nodes. A security SLA has to be valid between those entities.

- Energy Efficiency: Since OPA is designed for ICN/IoT, it eliminates the need to establish a secure connection between the resource-constrained devices acting as data producers and all the data consumers.

F.5 OPA: performance evaluation

IoT network uses the Ultra Narrow Band (UNB) for the Machine to Machine (M2M) communication. This network poses different problems such as increasing the end to end delay. In general, such network interconnects more than 7 million of devices and uses a point-to-point communication. In this section, the network is assisted with our intelligent algorithm that introduces ICN nodes in different levels. We show that network update upon upgrading optimal nodes by ICN/IoT

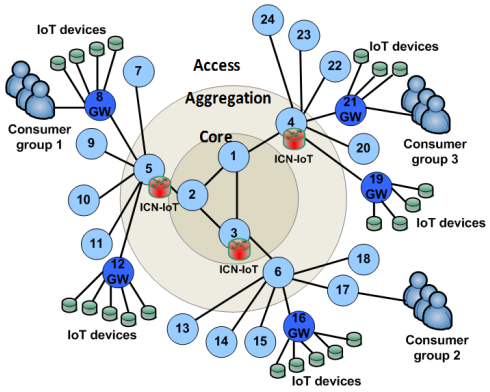


Figure F-1 – ICN-based IoT distribution network. Given the network topology, consumers requests, and objects served by content providers, OPA model chooses which server should be upgraded with ICN/IoT software.

software reduces the delay and dynamically (through on-line optimization) proposes potential points of operation (placement/upgrade).

For the sake of assessing OPA, we propose the following scale-free based topology that represents one of the major complex graphs as shown in Fig. F-1. It depicts different IoT gateways that collect IoT data from tiny IoT devices. These gateways assisted with ICN software (ICN/IoT) act as data producers on behalf of IoT devices. OPA algorithm aims to place ICN/IoT nodes to serve data consumer interests.

F.5.1 Scale free networks: a Barabási–Albert model-based network operator

We evaluate our scenario through the well known Barabási–Albert model [18] [1] undirected and weighted graph. Vertex connectivities follow a scale-free power law distribution $P(k)$. It represents the probability that a vertex interacts to k other vertices is: $P(k) \sim k^{-\gamma}$. The initial graph has 100 vertices (IoT gateway nodes) with a degree distribution that follows the power γ of 2.5, and obeys to the scale-free implementation of *psumtree*. Its Cut-tree-based transformation (Gusfield transformation of the Gomory-Hu algorithm is used here [76]) has only 99 edges (49.5%).

[1] Barabási–Albert graphs are not random topologies. Instead, they follow a power degree distribution (nonlinear model) so that can be used to assess network performance and as well as interpret the security benefits.

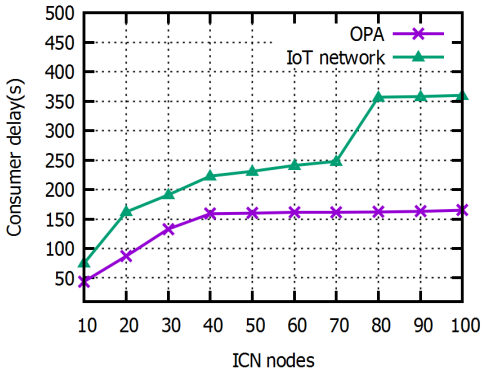


Figure F-2 – Data consumer delay

To assess OPA, we introduce the following metrics:

$$Consumer\ delay = \sum_{v \in V} \sum_{f \in F} d(s_v, s_f) \times f_{size} \times \max_{(i,j) \in P_{s_f,s}} \frac{1}{L_{i,j}} \tag{F.9}$$

$$OPA\ placement\ cost = \sum_{s \in S} \sum_{f \in F} x_f^s \times p_f^s \tag{F.10}$$

$$OPA\ placement\ delay = \sum_{s \in S} \sum_{f \in F} x_f^s \times f_{size} \times \max_{(i,j) \in P_{s_f,s}} \frac{1}{L_{i,j}} \tag{F.11}$$

$$Caching\ Strategy = \sum_{s \in S \setminus \{s_f\}} \sum_{f \in F} x_f^s \tag{F.12}$$

Eq. (F.9) defines the consumer delay metric that represents the response time while using OPA instead of the legacy IoT networking. sf and sv represent the data producer (aggregator or ICN gateway) and the data consumer point of attachments respectively. Equations (F.10) and (F.11) define the placement cost in terms of memory cost and placement delay (total delay to perform the placement along the shortest path from the IoT gateway to the optimal server node). Eq. (F.12) represents the average number of instantiated ICN nodes.

Fig. F-2 shows the impact of ICN nodes on the data consumer delay. Results show that OPA reduces the total delay.

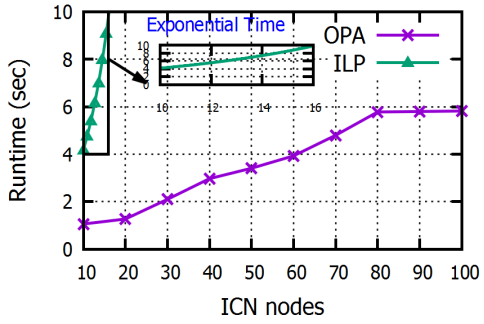


Figure F-3 – OPA run-time in scale-free network

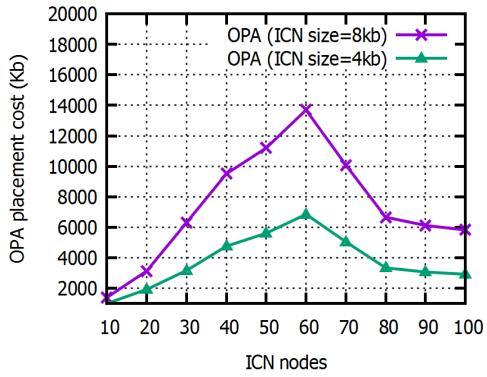
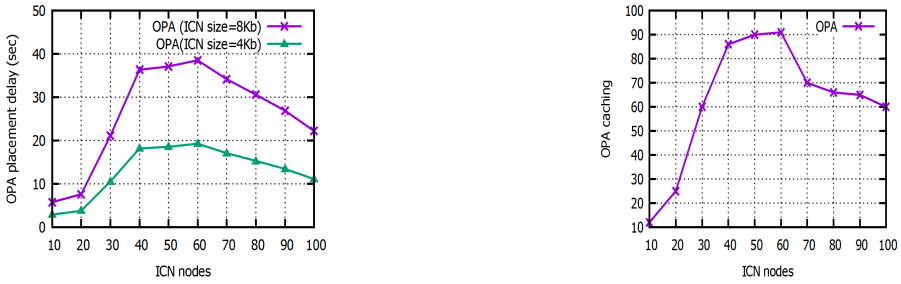


Figure F-4 – ICN/IoT placement (in network caching) cost

Fig. F-3 depicts the execution time of OPA for our scale-free network. It demonstrates the feasibility and the efficiency of OPA as results are of the order of the second (6 sec). ILP exact solution is tested using CPLEX environment as a proof of correctness of our model. We provide the curvature of this solution which is exponential and explodes when ICN number equals 20.

Fig. F-4 shows the OPA placement cost against ICN node number in scale-free based IoT network. Different ICN sizes (in terms of memory) are used. Fig. F-5a and Fig. F-5b show the OPA placement delay and the OPA caching respectively (OPA caching equals to the number of ICN migrating nodes which represents the caching policy strategy). The curves have an increasing slope up to $|F| = 60$ and then they decrease. This point represents the average ICN node number stabilizing the IoT network according to consumer group interests.



(a) ICN/IoT placement delay

(b) ICN/IoT caching strategy

Figure F-5 – OPA in scale-free IoT network

F.6 OPA efficiency: comparison with IoT networks

IoT networks such as Sigfox [25] are dedicated to low rate wireless data gathering. Several field trials and operational customers have started to use this facility. LoRA corresponds to the wireless part between devices and gateways. As Lora networks cover very large geographical areas, an infrastructure has to be built for data collection and routing. OPA can enhance this core network and provide the flexible ICN function explained before.

In Table F.2, we highlight how an IoT network such as Sigfox could be improved by applying OPA.

F.7 Conclusion and Future Work

We proposed in this chapter using ICN for IoT deployment and we introduced a new notion, which is how to assign ICN functionality to ICN/IoT nodes in a dynamic way based on network load and required services latency. In this context, we presented an Optimal Placement Algorithm (OPA) to enhance the caching deployment by network providers. OPA selects optimal network locations to serve as intermediate IoT publisher and pursue in-network caching. We illustrated IoT benefits from the in-network caching feature in ICN especially when applying our proposed algorithm. And we compared OPA in IoT over ICN with IoT over SigFox network (as an example of a popular IoT network deployment in France). Encour-

Table F.2 – Efficiency comparison for OPA and IoT SigFox network

Metrics	***ICN/IoT assisted OPA***	***Sigfox***
Caching strategy	Migrating ICN/IoT nodes acting as edge/fog computing nodes.	Using cloud data centers.
Delay	Minimize the end-to-end consumer delay	Significant.
Optimization cost	Additional cost of placement of ICN/IoT nodes	Minimum deployment cost.
Bitrate	High throughput network due to the in-network caching feature.	Low throughput network due to the UNB modulation.
Security	Object-based security that allows caching in untrusted intermediate nodes (proxies, caches, etc.).	Session-based security, frequency hopping.
Actuation latency	Bounded in with cache avoidance (OPA helps in routing).	Unbounded
Medium Access Control (MAC) layer	ICN networking stack that implies optimal bandwidth occupancy.	Without collision-avoidance that limits the bandwidth of IoT gateways [25].

aging results assure that OPA is scalable and efficient in terms of placing ICN/IoT nodes in a dynamic way. We will try to compare OPA in IoT over ICN against IoT solutions over WiFi and cellular networks without ICN. Finally, Vios platform will be enriched with ICN dockers.

Bibliography

[1] Emad Abd-Elrahman, Hossam Afifi, Hassnaa Moustafa, Mamadou Tourd Diallo, and Nicolas Marechal. Optimization of tv multicast delivery. *Journal of Electronic Systems*, 3(4):135–147, 2013.

[2] Accellion. Accellion, June 2016.

[3] B. Addis, D. Belabed, M. Bouet, and S. Secci. Virtual network functions placement and routing optimization. In *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pages 171–177, Oct 2015.

[4] Akamai. Content delivery network, June 2016.

[5] Takuya Akiba, Yoichi Iwata, Yosuke Sameshima, Naoto Mizuno, and Yosuke Yano. Cut tree construction from massive graphs. *CoRR*, abs/1609.08723, 2016.

[6] Amazon. Amazon cloudfront, June 2016.

[7] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying. Content-aware caching and traffic management in content distribution networks. In *2011 Proceedings IEEE INFOCOM*, pages 2858–2866, April 2011.

[8] M. Andrews, B. Shepherd, A. Srinivasan, P. Winkler, and F. Zane. Clustering and server selection using passive monitoring. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1717–1725 vol.3, 2002.

[9] Y. P. Aneja, R. Chandrasekaran, and K. P. K. Nair. Parametric min-cuts analysis in a network. *Discrete Appl. Math.*, 127(3):679–689, May 2003.

[10] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 313–324, New York, NY, USA, 2003. ACM.

[11] Oscar Ardaiz, Felix Freitag, and Leandro Navarro. Improving the service time of web clients using server redirection. *SIGMETRICS Perform. Eval. Rev.*, 29(2):39–44, September 2001.

[12] Baruch Awerbuch and Rohit Khandekar. Greedy distributed optimization of multi-commodity flows. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pages 274–283, New York, NY, USA, 2007. ACM.

[13] Baruch Awerbuch, Rohit Khandekar, and Satish Rao. Distributed algorithms for multicommodity flow problems via approximate steepest descent framework. *ACM Trans. Algorithms*, 9(1):3:1–3:14, December 2012.

[14] S. Ayyasamy and S. N. Sivanandam. A qos-aware intelligent replica management architecture for content distribution in peer-to-peer overlay networks. *CoRR*, abs/0912.2296, 2009.

[15] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 383–388, New York, NY, USA, 2003. ACM.

[16] Emmanuel Baccelli, Christian Mehlis, Oliver Hahm, Thomas C. Schmidt, and Matthias Wählisch. Information centric networking in the iot: Experiments with ndn in the wild. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, pages 77–86, New York, NY, USA, 2014. ACM.

[17] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Commun. ACM*, 46(2):43–48, February 2003.

[18] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[19] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[20] Arsany Basta, Wolfgang Kellerer, Marco Hoffmann, Hans Jochen Morper, and Klaus Hoffmann. Applying nfv and sdn to lte mobile core gateways, the functions placement problem. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges*, AllThingsCellular '14, pages 33–38, New York, NY, USA, 2014. ACM.

[21] Pierre-Olivier Bauguion, Walid Ben-Ameur, and Éric Gourdin. Efficient algorithms for the maximum concurrent flow problem. *Networks*, 65(1):56–67, 2015.

[22] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, pages 41–41, Berkeley, CA, USA, 2005. USENIX Association.

[23] Sourjya Bhaumik, Shoban Preeth Chandrabose, Manjunath Kashyap Jataprolu, Gautam Kumar, Anand Muralidhar, Paul Polakos, Vikram Srinivasan, and Thomas Woo. Cloudiq: A framework for processing base stations in a data center. 08 2012.

[24] M. Bjorkqvist, L. Y. Chen, and X. Zhang. Minimizing retrieval cost of multilayer content distribution systems. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–6, June 2011.

[25] Martin Bor, John Vidler, and Utz Roedig. Lora for the internet of things. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, EWSN '16, pages 361–366, USA, 2016. Junction Publishing.

[26] S. Borst, V. Gupta, and A. Walid. Distributed caching algorithms for content distribution networks. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, March 2010.

[27] M. Bouet, J. Leguay, and V. Conan. Cost-based placement of vdpi functions in nfv infrastructures. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–9, April 2015.

[28] N. Bouten, J. Famaey, R. Mijumbi, B. Naudts, J. Serrat, S. Latré, and F. De Turck. Towards nfv-based multimedia delivery. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 738–741, May 2015.

[29] Torsten Braun and Tuan Anh Trinh. *Energy Efficiency Issues in Information-Centric Networking*, pages 271–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[30] James Broberg, Rajkumar Buyya, and Zahir Tari. Metacdn: Harnessing 'storage clouds' for high performance content delivery. *Journal of Network and Computer Applications*, 32(5):1012 – 1022, 2009. Next Generation Content Networks.

[31] B. Carbunar, M. Pearce, V. Vasudevan, and M. Needham. Predictive caching for video on demand cdns. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pages 1–5, Dec 2011.

[32] CCN. The ccnx project, parc a xerox company, June 2016.

[33] cedric.westphal@huawei.com, Christopher Mueller, Andrea Detti, Daniel Corujo, aytav.azgin, Jianping Wang, Marie-Jose Montpetit, Niall Murray, Shucheng LIU (Will), Stefan Lederer, Christian Timmerer, and Daniel Posch. Adaptive Video Streaming over Information-Centric Networking (ICN). RFC 7933, August 2016.

[34] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue 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, pages 1–14, New York, NY, USA, 2007. ACM.

[35] S. A. Chellouche, D. Négru, Y. Chen, and M. Sidibe. Home-box-assisted content delivery network for internet video-on-demand services. In *2012 IEEE Symposium on Computers and Communications (ISCC)*, pages 000544–000550, July 2012.

[36] Bin Cheng, Lex Stein, Hai Jin, and Zheng Zhang. A framework for lazy replication in p2p vod. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '08, pages 93–98, New York, NY, USA, 2008. ACM.

[37] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson. In-network caching effect on optimal energy consumption in content-centric networking. In *2012 IEEE International Conference on Communications (ICC)*, pages 2889–2894, June 2012.

[38] Codeen. Codeen, June 2016.

[39] Richard Cole and Ramesh Hariharan. A fast algorithm for computing steiner edge connectivity. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 167–176, New York, NY, USA, 2003. ACM.

[40] Comodin. Comodin, June 2016.

[41] Luis M. Contreras, Paul Doolan, Hkon Lønsethagen, and Diego R. López. Operational, organizational and business challenges for network operators in the context of sdn and nfv. *Comput. Netw.*, 92(P2):211–217, December 2015.

[42] Doug Cutting and Mike Cafarella. Apache hadoop, June 2016.

[43] J. Dai, F. Liu, B. Li, B. Li, and J. Liu. Collaborative caching in wireless video streaming through resource auctions. *IEEE Journal on Selected Areas in Communications*, 30(2):458–466, February 2012.

[44] Pradyumna Dash. *Getting Started with Oracle VM VirtualBox*. Packt Publishing, 2013.

[45] K. Delgadillo. Cisco distributed director.

[46] Mamadou Tourad Diallo. *Quality of experience and video services adaptation*. PhD thesis, 2015. Thèse de doctorat dirigée par Afifi, Hossam Informatique et télécommunications Evry, Institut national des télécommunications 2015.

[47] Mamadou Tourad Diallo, Frédéric fieau, Emad Abd-Elrahman, and Hossam afifi. Utility-based Approach for Video Service Delivery Optimization. *ICSNC 2014: International Conference on Systems and Network Communication*, pages 5–10, 2014.

[48] Mamadou Tourad Diallo, Nicolas Marechal, and Hossam Afifi. A hybrid contextual user perception model for streamed video quality assessment. In *Proceedings of the 2013 IEEE International Symposium on Multimedia*, ISM '13, pages 518–519, Washington, DC, USA, 2013. IEEE Computer Society.

[49] Mamadou Tourad Diallo, Nicolas Marechal, and Hossam Afifi. A hybrid contextual user perception model for streamed video quality assessment. In *Proceedings of the 2013 IEEE International Symposium on Multimedia*, ISM '13, pages 518–519, Washington, DC, USA, 2013. IEEE Computer Society.

[50] Mamadou Tourad Diallo, Hassnaa Moustafa, Hossam Afifi, and Khalil Ur Rehman Laghari. Quality of experience for audio-visual services. In *UP-TO-US '12 Workshop : User-Centric Personalized TV ubiquitOus and secUre Services*, pages 299–305, Berlin, Germany, July 2012. Fraunhofer FOKUS.

[51] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, September 2002.

[52] W. Ding, W. Qi, J. Wang, and B. Chen. Openscaas: an open service chain as a service platform toward the integration of sdn and nfv. *IEEE Network*, 29(3):30–35, May 2015.

[53] Yuanyuan Dong, Eli V. Olinick, T. Jason Kratz, and David W. Matula. A compact linear programming formulation of the maximum concurrent flow problem. *Netw.*, 65(1):68–87, January 2015.

[54] Fred Douglis and M Frans Kaashoek. Scalable internet services. 5:36–37, 08 2001.

[55] F. Dudouet, P. Harsh, S. Ruiz, A. Gomes, and T. M. Bohnert. A case for cdn-as-a-service in the cloud: A mobile cloud networking argument. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 651–657, Sept 2014.

[56] H. E. Egilmez, S. Civanlar, and A. M. Tekalp. An optimization framework for qos-enabled adaptive video streaming over openflow networks. *IEEE Transactions on Multimedia*, 15(3):710–715, April 2013.

[57] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp. Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. In *Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–8, Dec 2012.

[58] Kayhan Erciyes. *Complex Networks: An Algorithmic Perspective*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2014.

[59] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, June 2000.

[60] Hamid Farhady, HyunYong Lee, and Akihiro Nakao. Software-defined networking: A survey. *Computer Networks*, 81:79 – 95, 2015.

[61] B. Feng, H. Zhou, H. Zhang, J. Jiang, and S. Yu. A popularity-based cache consistency mechanism for information-centric networking. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2016.

[62] D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 2010.

[63] Open Networking Foundation. Openflow, 2013.

[64] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. Pushing cdn-isp collaboration to the limit. *SIGCOMM Comput. Commun. Rev.*, 43(3):34–44, July 2013.

[65] L. Fratta, M. Gerla, and L. Kleinrock. The flow deviation method: An approach to store-and-forward communication network design. *Networks*, 3(2):97–133, 1973.

[66] Michael J. Freedman, Eric Freudenthal, and David Mazières. Democratizing content publication with coral. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, pages 18–18, Berkeley, CA, USA, 2004. USENIX Association.

[67] Norihito Fujita, Yuichi Ishikawa, Atsushi Iwata, and Rauf Izmailov. Coarse-grain replica management strategies for dynamic replication of web contents. *Comput. Netw.*, 45(1):19–34, May 2004.

[68] Marisol García-Valls, Tommaso Cucinotta, and Chenyang Lu. Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture*, 60(9):726 – 740, 2014.

[69] N. Garg and J. Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 300–309, Nov 1998.

[70] P. Georgopoulos, M. Broadbent, B. Plattner, and N. Race. Cache as a service: Leveraging sdn to efficiently and transparently support video-on-demand

on the last mile. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9, Aug 2014.

[71] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, FhMN '13, pages 15–20, New York, NY, USA, 2013. ACM.

[72] Andrew V. Goldberg. A natural randomization strategy for multicommodity flow and related algorithms. volume 42, pages 249 – 256, 1992.

[73] R. E. Gomory and T. C. Hu. Multi-Terminal Network Flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[74] Network Working Group. Nfvi pop network topology: Problem statement, 2016.

[75] K. Guan, G. Atkinson, D. C. Kilper, and E. Gulsen. On the energy efficiency of content delivery architectures. In *2011 IEEE International Conference on Communications Workshops (ICC)*, pages 1–6, June 2011.

[76] Dan Gusfield. Very simple methods for all pairs network flow analysis. *SIAM J. Comput.*, 19(1):143–155, February 1990.

[77] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer. Caching in named data networking for the wireless internet of things. In *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, pages 1–6, April 2015.

[78] Gerhard Haílinger and Franz Hartleb. Content delivery and caching from a network provider's perspective. *Comput. Netw.*, 55(18):3991–4006, December 2011.

[79] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, Feb 2015.

[80] Ramesh Hariharan, Telikepalli Kavitha, Debmalya Panigrahi, and Anand Bhalgat. An $\tilde{O}(mn)$ gomory-hu tree construction algorithm for unweighted graphs. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 605–614, New York, NY, USA, 2007. ACM.

[81] Tanja Hartmann and Dorothea Wagner. Dynamic gomory-hu tree construction - fast and simple. *CoRR*, abs/1310.0178, 2013.

[82] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal. Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc). *IEEE Network*, 28(6):18–26, Nov 2014.

[83] Ching-Lai Hwang and Kwangsun Yoon. Multiple attribute decision making. *Springer-Verlag Berlin Heidelberg*, 186, 1981.

[84] IBM. Ibm ilog cplex optimization studio community edition, version 12.5, 2015.

[85] H. Ibn-Khedher, E. Abd-Elrahman, and H. Afifi. Omac: Optimal migration algorithm for virtual cdn. In *2016 23rd International Conference on Telecommunications (ICT)*, pages 1–6, May 2016.

[86] H. Ibn-Khedher, E. Abd-Elrahman, H. Afifi, and J. Forestier. Network issues in virtual machine migration. In *Networks, Computers and Communications (ISNCC), 2015 International Symposium on*, pages 1–6, May 2015.

[87] H. Ibn-Khedher, M. Hadji, E. Abd-Elrahman, H. Afifi, and A. E. Kamal. Scalable and cost efficient algorithms for virtual cdn migration. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pages 112–120, Nov 2016.

[88] Hatem Ibn-Khedher, Emad Abd-Elrahman, Ahmed E. Kamal, and Hossam Afifi. Opac: An optimal placement algorithm for virtual {CDN}. *Computer Networks*, 120:12 – 27, 2017.

[89] Hatem Ibn-Khedher, Hossam Afifi, and Hassnaa Moustafa. Optimal placement algorithm (opa) for iot over icn. *INFOCOM NOM 2017 :Named-Oriented Mobility: Architectures, Algorithms, and Applications workshop*, (document in press- published online 05 May 2017).

[90] S. Imai, K. Leibnitz, and M. Murata. Energy efficient content locations for in-network caching. In *2012 18th Asia-Pacific Conference on Communications (APCC)*, pages 554–559, Oct 2012.

[91] S. Imai, K. Leibnitz, and M. Murata. Energy-aware cache management for content-centric networking. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 1623–1629, March 2013.

[92] I. S. H. Yeung J. Ni, D. H. K. Tsang and X. Hei. Hierarchical content routing in large-scale multimedia content delivery network. *Proceedings of IEEE International Conference on Communications, 2003 (ICC '03)*, 2:854–859, May 2003.

[93] Van Jacobson. A new way to look at networking, jun 2006.

[94] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. Networking named content. *Commun. ACM*, 55(1):117–124, January 2012.

[95] R. Jain and S. Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31, November 2013.

[96] Manar Jammal, Taranpreet Singh, Abdallah Shami, Rasool Asal, and Yiming Li. Software-defined networking: State of the art and research challenges. *CoRR*, abs/1406.0124, 2014.

[97] Y. Jin, Y. Wen, G. Shi, G. Wang, and A. V. Vasilakos. Codaas: An experimental cloud-centric content delivery platform for user-generated contents. In *Computing, Networking and Communications (ICNC), 2012 International Conference on*, pages 934–938, Jan 2012.

[98] K.L Johnson, J.F Carr, M.S Day, and M.F Kaashoek. The measured performance of content distribution networks. *Comput. Commun.*, 24(2):202–206, February 2001.

[99] U. Kalim, M. K. Gardner, E. J. Brown, and W. c. Feng. Seamless migration of virtual machines across networks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, July 2013.

[100] Jussi Kangasharju, James Roberts, and Keith W. Ross. Object replication strategies in content distribution networks. *Comput. Commun.*, 25(4):376–383, March 2002.

[101] George Karakostas. Faster approximation schemes for fractional multicommodity flow problems. *ACM Trans. Algorithms*, 4(1):13:1–13:17, March 2008.

[102] David Karger, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Brian Kim, Luke Matkins, and Yoav Yerushalmi. Web caching with consistent hashing. In *Proceedings of the Eighth International Conference on World Wide Web*, WWW '99, pages 1203–1213, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[103] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 217–226, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[104] Philip Klein, Serge Plotkin, Clifford Stein, and Éva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, 1994.

[105] Bikash Koley. Software defined networking at scale, 2014.

[106] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2012.

[107] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 169–182, New York, NY, USA, 2001. ACM.

[108] Dirk Kutscher, Suyong Eum, Kostas Pentikousis, Ioannis Psaras, Daniel Corujo, Damien Saucez, Thomas C. Schmidt, and Matthias Wählisch. Information-Centric Networking (ICN) Research Challenges. RFC 7927, July 2016.

[109] N. Laoutaris, G. Smaragdakis, A. Bestavros, I. Matta, and I. Stavrakakis. Distributed selfish caching. *IEEE Transactions on Parallel and Distributed Systems*, 18(10):1361–1376, Oct 2007.

[110] Irwin Lazar and William Terrill. Exploring content delivery networking. *IT Professional*, 3(4):47–49, July 2001.

[111] A. Leff, P. S. Yu, and J. L. Wolf. Policies for efficient memory utilization in a remote caching architecture. In *[1991] Proceedings of the First International Conference on Parallel and Distributed Information Systems*, pages 198–207, Dec 1991.

[112] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 422–431, Oct 1988.

[113] Q. Li, J. Huai, J. Li, T. Wo, and M. Wen. Hypermip: Hypervisor controlled mobile ip for virtual machine live migration across networks. In *2008 11th IEEE High Assurance Systems Engineering Symposium*, pages 80–88, Dec 2008.

[114] A. Lindgren, F. B. Abdesslem, B. Ahlgren, O. Schelén, and A. M. Malik. Design choices for the iot in information-centric networks. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 882–888, Jan 2016.

[115] Aleksander Madry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. *CoRR*, abs/1003.5907, 2010.

[116] M. Mangili, F. Martignon, and A. Capone. Stochastic planning for content delivery: Unveiling the benefits of network functions virtualization. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 344–349, Oct 2014.

[117] Michele Mangili, Fabio Martignon, and Antonio Capone. Optimal design of information centric networks. *Computer Networks*, 91:638 – 653, 2015.

[118] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. Clickos and the art of network function virtualization. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 459–473, Berkeley, CA, USA, 2014. USENIX Association.

[119] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob. Toward an sdn-enabled nfv architecture. *IEEE Communications Magazine*, 53(4):187–193, April 2015.

[120] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *CoRR*, abs/1509.07675, 2015.

[121] Dave Mishchenko. *VMware ESXi: Planning, Implementation, and Security*. Course Technology Press, Boston, MA, United States, 1st edition, 2010.

[122] H. Moens and F. D. Turck. Vnf-p: A model for efficient placement of virtualized network functions. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 418–423, Nov 2014.

[123] C.D. Mortensen. *Communication Theory*. Studies in Communication. Transaction Publishers, 2011.

[124] H. Nam, K. H. Kim, J. Y. Kim, and H. Schulzrinne. Towards qoe-aware video streaming using sdn. In *2014 IEEE Global Communications Conference*, pages 1317–1322, Dec 2014.

[125] NDN. Named data networking, jun 2016.

[126] NetInf. Network of information (netinf), June 2016.

[127] Jian Ni and D. H.K. Tsang. Large-scale cooperative caching and application-level multicast in multimedia content delivery networks. *Comm. Mag.*, 43(5):98–105, May 2005.

[128] Opendaylight. Opendaylight platform, June 2016.

[129] Orange. Bienvenue sur le site du projet dvd2c, 2016.

[130] M P. Szymaniak, Guillaume Pierre, and Maarten van Steen. Netairt: A dns-based redirection system for apache. 02 2018.

[131] Vivek S. Pai, Mohit Aron, Gaurov Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, and Erich Nahum. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the Eighth International*

Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS VIII, pages 205–216, New York, NY, USA, 1998. ACM.

[132] Shrideep Pallickara and Geoffrey Fox. Enabling hierarchical dissemination of streams in content distribution networks. *Concurrency and Computation: Practice and Experience*, 24(14):1594–1606, 2012.

[133] George Pallis and Athena Vakali. Insight and perspectives for content delivery networks. *Commun. ACM*, 49(1):101–106, January 2006.

[134] Kihong Park, H T. Kung, and C H. Wu. Content networks: Taxonomy and new approaches. 07 2002.

[135] A. Pathan and R. Buyya. A taxonomy and survey of content delivery networks,. Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia., Feb. 2007.

[136] Gang Peng. CDN: content distribution network. *CoRR*, cs.NI/0411069, 2004.

[137] Kostas Pentikousis, Borje Ohlman, Daniel Corujo, Gennaro Boggia, Gareth Tyson, Elwyn B. Davies, Antonella Molinaro, and Suyong Eum. Information-Centric Networking: Baseline Scenarios. RFC 7476, October 2015.

[138] Kostas Pentikousis, Borje Ohlman, Elwyn B. Davies, Gennaro Boggia, and Spiros Spirou. Information-Centric Networking: Evaluation and Security Considerations. RFC 7945, September 2016.

[139] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.

[140] G. Pierre and M. van Steen. Globule: a collaborative content delivery network. *IEEE Communications Magazine*, 44(8):127–133, Aug 2006.

[141] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pages 55–60, New York, NY, USA, 2012. ACM.

[142] PSIRP. Publish-subscribe internet routing paradigm, June 2016.

[143] Nadia N. Qadri, Antonio Liotta, Muhammad Altaf, Martin Fleury, and Mohammed Ghanbari. Effective video streaming using mesh p2p with mdc over manets. *J. Mob. Multimed.*, 5(4):301–316, December 2009.

[144] J. Quevedo, D. Corujo, and R. Aguiar. Consumer driven information freshness approach for content centric networking. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 482–487, April 2014.

[145] Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 43–52, Washington, DC, USA, 2002. IEEE Computer Society.

[146] Tomasz Radzik. Fast deterministic approximation for the multicommodity flow problem. *Mathematical Programming*, 78(1):43–58, Jul 1996.

[147] S. Ramakrishnan, X. Zhu, F. Chan, and K. Kambhatla. Sdn based qoe optimization for http-based adaptive video streaming. In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 120–123, Dec 2015.

[148] P. Rodriguez, C. Spanner, and E. W. Biersack. Analysis of web caching architectures: hierarchical and distributed caching. *IEEE/ACM Transactions on Networking*, 9(4):404–418, Aug 2001.

[149] Tiago Rosado and Jorge Bernardino. An overview of openstack architecture. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, IDEAS '14, pages 366–367, New York, NY, USA, 2014. ACM.

[150] E. Rosas, N. Hidalgo, and M. Marin. Two-level result caching for web search queries on structured p2p networks. In *2012 IEEE 18th International Conference on Parallel and Distributed Systems*, pages 221–228, Dec 2012.

[151] E. J. Rosensweig, J. Kurose, and D. Towsley. Approximate models for general cache networks. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, March 2010.

[152] Ponnusamy S P and Karthikeyan Eswaramurthy. Cache optimization on hotpoint proxy caching using weighted-rank cache replacemnt policy. 35, 08 2013.

[153] Mehdi Sabeur, Ghazi Al Sukkar, Badii Jouaber, Djamal Zeghlache, and Hossam Afifi. Mobile party: A mobility management solution for wireless mesh network. In *Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on*, pages 45–45. IEEE, 2007.

[154] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. *SIGOPS Oper. Syst. Rev.*, 36(SI):315–327, December 2002.

[155] Klaus Schneider, Cheng Yi, Beichuan Zhang, and Lixia Zhang. A practical congestion control scheme for named data networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, pages 21–30, New York, NY, USA, 2016. ACM.

[156] Farhad Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, April 1990.

[157] Farhad Shahrokhi and David W. Matula. On solving large maximum concurrent flow problems. In *Proceedings of the 15th Annual Conference on Computer Science*, CSC '87, pages 205–209, New York, NY, USA, 1987. ACM.

[158] Wentao Shang, Zhe Wen, Qiuhan Ding, Alexander Afanasyev, and Lixia Zhang. NDNFS: An NDN-friendly file system. Technical Report NDN-0027, NDN, October 2014.

[159] Jiu sheng PENG and Xiong jian LIANG. Content delivery network and its regulation. *The Journal of China Universities of Posts and Telecommunications*, 13(4):98 – 101, 2006.

[160] J. Shim, P. Scheuermann, and R. Vingralek. Proxy cache algorithms: design, implementation, and performance. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):549–562, Jul 1999.

[161] Manolis Sifalakis, Basil Kohler, Christopher Scherb, and Christian Tschudin. An information centric network for computing the distribution of computations. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, pages 137–146, New York, NY, USA, 2014. ACM.

[162] Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Replication for web hosting systems. *ACM Comput. Surv.*, 36(3):291–334, September 2004.

[163] Suman Ramkumar Srinivasan, Jae Woo Lee, Dhruva L. Batni, and Henning G. Schulzrinne. Activecdn: Cloud computing meets content delivery networks. *Department of Computer Science, Columbia University*, 32(5):1012 – 1022, 2011. Next Generation Content Networks.

[164] T Suresh and K Vekatachalapathy. Popularity aware limited caching for reliable on demand p2p video streaming. 58, 10 2012.

[165] Tamayo-IbnKhedher. vios (virtual infrastructure optimization simulator), 2016.

[166] Xueyan Tang and S. T. Chanson. Coordinated en-route web caching. *IEEE Transactions on Computers*, 51(6):595–607, Jun 2002.

[167] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, and Y. Ji. Performance of probabilistic caching and cache replacement policies for content-centric networks. In *39th Annual IEEE Conference on Local Computer Networks*, pages 99–106, Sept 2014.

[168] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12, April 2006.

[169] Tai-Won Um, Hyunwoo Lee, Won Ryu, and Jun Kyun Choi. Dynamic resource allocation and scheduling for cloud-based virtual content delivery networks. volume 36, pages 197–205, Apr 2014.

[170] ETSI GS NFV V1.1.1. Network functions virtualization (nfv); use cases, 2013.

[171] ETSI GS NFV 002 V1.1.1. Network functions virtualization (nfv); architectural framework, 2013.

[172] ETSI GS NFV-MAN 001 V1.1.1. Network functions virtualization (nfv); management and orchestration, 2014.

[173] Athena Vakali and George Pallis. Content delivery networks: Status and trends. *IEEE Internet Computing*, 7(6):68–74, November 2003.

[174] Rodrigo Aldecoa Dmitri Krioukov Lan Wang Beichuan Zhang Lixia Zhang Vince Lehman, Ashlesh Gawande. An experimental investigation of hyperbolic routing with a smart forwarding plane in ndn. In *IEEE IWQoS Symposium*, 2016.

[175] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli. In-network caching of internet-of-things data. In *2014 IEEE International Conference on Communications (ICC)*, pages 3185–3190, June 2014.

[176] Wikipedia. Mobile agents, 2016.

[177] H. Woo, S. Han, E. Heo, J. Kim, and S. Shin. A virtualized, programmable content delivery network. pages 159–168, April 2014.

[178] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang. Toward a software-based network: integrating software defined networking and network function virtualization. *IEEE Network*, 29(3):36–41, May 2015.

[179] W. Wu and J. C. S. Lui. Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. In *2011 Proceedings IEEE INFOCOM*, pages 1206–1214, April 2011.

[180] W. Wu, R. T. B. Ma, and J. C. S. Lui. On incentivizing caching for p2p-vod systems. In *2012 Proceedings IEEE INFOCOM Workshops*, pages 164–169, March 2012.

[181] Cheng Yi, Jerald Abraham, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. On the role of routing in named data networking. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, pages 27–36, New York, NY, USA, 2014. ACM.

[182] Xin Yu and Z. Kedem. A distributed adaptive cache update algorithm for the dynamic source routing protocol. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pages 730–739 vol. 1, March 2005.

[183] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. Schematizing trust in named data networking. In *Proceedings of the 2Nd ACM Conference on Information-Centric Networking*, ACM-ICN '15, pages 177–186, New York, NY, USA, 2015. ACM.

[184] Guoqiang Zhang, Yang Li, and Tao Lin. Caching in information centric networking: A survey. *Comput. Netw.*, 57(16):3128–3141, November 2013.

[185] Haitao Zhang, Zhehao Wang, Christopher Scherb, Claudio Marxer, Jeff Burke, Lixia Zhang, and Christian Tschudin. Sharing mhealth data via named data networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, pages 142–147, New York, NY, USA, 2016. ACM.

[186] Y. Zhang, A. Afanasyev, J. Burke, and L. Zhang. A survey of mobility support in named data networking. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 83–88, April 2016.

[187] Yanyong Zhang, Dipankar Raychadhuri, Luigi Alfredo Grieco, Emmanuel Baccelli, Jeff Burke, Ravi Ravindran, and Guoqiang Wang. ICN based Architecture for IoT - Requirements and Challenges. Internet-Draft draft-zhang-iot-icn-challenges-02, Internet Engineering Task Force, February 2016. Work in Progress.

[188] Yanyong Zhang, Dipankar Raychadhuri, Luigi Alfredo Grieco, Sicari Sabrina, Hang Liu, Satyajayant Misra, and Ravi Ravindran. ICN based Architecture for IoT. Internet-Draft draft-zhang-icnrg-iot-architecture-00, Internet Engineering Task Force, July 2016. Work in Progress.

[189] Yu Zhang, Hongli Zhang, and Lixia Zhang. Kite: A mobility support scheme for ndn. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, pages 179–180, New York, NY, USA, 2014. ACM.

[190] Mo Zhou, Bo Ji, Kun Peng Han, and Hong Sheng Xi. A cooperative hybrid caching strategy for p2p mobile network. In *Instruments, Measurement, Electronics and Information Engineering*, volume 347 of *Applied Mechanics and Materials*, pages 1992–1996. Trans Tech Publications, 10 2013.

[191] Yipeng Zhou, Tom Z. J. Fu, and Dah Ming Chiu. On replication algorithm in p2p vod. *IEEE/ACM Trans. Netw.*, 21(1):233–243, February 2013.