PERFORMANCE ENHANCEMENT OF CONTENT DELIVERY IN MOBILE NETWORKS

Inaugural Dissertation
of the Faculty of Science
of the University of Bern
and the Faculty of Sciences and Technology
of the University of Coimbra

presented by

André Sérgio Nobre Gomes

from Coimbra, Portugal

Research supervisors:
Professor Dr. Torsten Braun
Institute of Computer Science, University of Bern
Professor Dr. Edmundo Monteiro
Department of Informatics Engineering, University of Coimbra

PERFORMANCE ENHANCEMENT OF CONTENT DELIVERY IN MOBILE NETWORKS

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern
und der Faculty of Sciences and Technology
of the University of Coimbra

vorgelegt von

André Sérgio Nobre Gomes

von Coimbra, Portugal

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik, Universität Bern
Professor Dr. Edmundo Monteiro
Department of Informatics Engineering, University of Coimbra

Der Dekan:
Prof. Dr. Gilberto Colangelo

Bern, 04.10.2016

PERFORMANCE ENHANCEMENT OF CONTENT DELIVERY IN MOBILE NETWORKS

Dissertação Inaugural
da Faculty of Science
of the University of Bern
e da Faculdade de Ciências e Tecnologia
da Universidade de Coimbra

apresentada por

André Sérgio Nobre Gomes

de Coimbra, Portugal

Orientadores da investigação:
Professor Dr. Torsten Braun
Institute of Computer Science, University of Bern
Professor Dr. Edmundo Monteiro
Departamento de Informática, Universidade de Coimbra

Coimbra, 04.10.2016

Abstract

With the recent advances in mobile technology, such as the boom in the usage of smartphones and mobile networks, content demand of mobile users has increased significantly. This exponential increase exposed several limitations on the current mechanisms for content delivery. Namely, the current paradigm for requesting content focuses on resources and not on content as it would be ideal to improve content delivery. Despite the existing efforts to overcome this limitation that greatly affects overall performance and efficiency, there are still open challenges that need to be addressed. The first challenge is how to explore new technologies together with a new content request paradigm, aiming at having deeper integration with existing networks and availability of compute, storage and network resources whenever and wherever they are necessary to handle different amounts of user loads. The second challenge is dealing with the integration of content delivery mechanisms with mobile networks and all its particularities, such as constrained architectures and demanding processing requirements. The third challenge is the usage of multiple radio technologies in a transparent and coordinated manner to improve overall efficiency and performance of mobile networks. The fourth challenge is the extension of content caching to the edge of mobile networks with efficient usage of storage resources and reduced latency for content delivery. The fifth and last challenge is the proper distribution of content among edge caches ensuring adaptability to the mobility of users.

The key contributions of this thesis aim at addressing those challenges, by providing an integrated architecture with a set of strategies, mechanisms and algorithms that tackle the identified problems in detail and span across multiple knowledge domains. A first contribution concerns a cloud-based system for content delivery, which is easily deployable in new locations, integrates with other services and adapts itself to different user loads. Later on, the integration of this system with mobile networks is depicted to bring the enhancements of that system directly to content delivery in mobile networks, and mechanisms to ensure that it is feasible and follows current standards and specifications are described. Afterwards, and considering that multiple network technologies can be used simultaneously, strategies are proposed to efficiently handle load balancing and offloading of content delivery between different radio technologies in mobile networks, ensuring complete transparency for end users and efficient usage of available resources. Next, and because the previous contributions make caching at the edge of mobile networks a reality, caching strategies for the edge of mobile networks are highlighted, focusing on maximizing performance in terms of latency reduction while optimizing storage usage. Finally, content distribution strategies for edge caches based on users' movement and their interests are presented. These aim at improving edge caching by trying to guarantee that content is cached where it yields the greatest benefits for nearby users.

Results gathered from the evaluation of the contributions of this thesis demonstrate that they bring major benefits for content delivery and that performance is greatly enhanced. These improvements are very important from multiple perspectives, as all the involved stakeholders, from mobile users to content providers and mobile operators, may benefit at different levels such as quality of experience, satisfaction and costs.

Resumo

Com os recentes avanços na tecnologia móvel, tais como o aumento estrondoso da utilização de smartphones e redes móveis, a procura de conteúdos por parte de utilizadores móveis tem aumentado significativamente. Este aumento exponencial expôs várias limitações nos mecanismos actuais para a entrega de conteúdos. Nomeadamente, o actual paradigma para solicitar conteúdos foca-se em recursos e não nos conteúdos como seria ideal para melhorar a entrega de conteúdos. Apesar dos esforços já existentes para ultrapassar esta limitação que afecta de forma substancial e genérica a performance e eficiência, ainda existem desafios em aberto para serem endereçados. O primeiro desafio é como explorar novas tecnologias em conjunto com um novo paradigma para pedido de conteúdos, com o objectivo de ter maior integração com as redes existentes e a disponibilidade de recursos de computação, armazenamento e rede quando e onde forem necessários, para assim acomodar diferentes quantidades de carga de utilizadores. O segundo desafio prende-se com a integração de mecanismos para entrega de conteúdos nas redes móveis e todas as particularidades que isso acarreta, nomeadamente arquitecturas limitadas e requisitos de processamento exigentes. O terceiro desafio é a utilização de múltiplas tecnologias de rádio de forma transparente e coordenada para aumentar de forma global a eficiência e performance das redes móveis. O quarto desafio é a extensão de caches de conteúdos até à orla das redes móveis com vista a uma utilização eficiente de recursos de armazenamento e latências reduzidas na entrega de conteúdos. O quinto e último desafio é a distribuição adequada de conteúdos entre caches na orla das redes móveis, assegurando adaptabilidade à mobilidade dos utilizadores.

As contribuições chave desta tese têm como objectivo endereçar os desafios referidos anteriormente, providenciando uma arquitectura integrada com um conjunto de estratégias, mecanismos e algoritmos que tratam dos problemas identificados em detalhe e abrangem múltiplos domínios do conhecimento. Uma primeira contribuição diz respeito a um sistema baseado na nuvem para entrega de conteúdos, o qual é facilmente implementável em novas localizações, integra com outros serviços e adapta-se a si próprio a diferentes cargas de utilizadores. De seguida, é apresentada a integração deste sistema com redes móveis e são descritos mecanismos para garantir a sua praticabilidade e adequação às normas e padrões existentes, para assim trazer as suas melhorias directamente à entrega de conteúdos em redes móveis. Depois, e considerando que múltiplas tecnologias de rede podem ser utilizadas em simultâneo, são apresentadas estratégias para efectuar balanceamento e transferência de carga entre múltiplas tecnologias de rádio em redes móveis, assegurando total transparência para os utilizadores finais e uma utilização eficiente dos recursos existentes. Mais tarde, e porque as contribuições anteriores tornam a existência de caches na orla das redes móveis uma realidade, estratégias para caching na orla das redes móveis são realçadas, focando-se em maximizar a performance em termos de redução de latência e na optimização da utilização de recursos de armazenamento. Finalmente, estratégias para a distribuição de conteúdos em caches na orla das redes móveis baseadas em movimento e interesses dos utilizadores são apresentadas. Estas visam a melhoria das caches na orla das redes móveis ao tentar garantir que os conteúdos são colocados em caches onde irão trazer os maiores benefícios possíveis para utilizadores na imediação.

Os resultados obtidos da avaliação das contribuições desta tese demonstram que as mesmas trazem benefícios relevantes para a entrega dos conteúdos juntamente com um aumento significativo da performance. Estas melhorias são muito importantes de múltiplas perspectivas, dado que todas as partes interessadas, desde utilizadores móveis até produtores de conteúdos e operadores móveis, podem beneficiar a diferentes níveis tais como qualidade da experiência, satisfação e custos.

Contents

Contents **i**

List of Figures **v**

List of Tables **vii**

1 Introduction **3**
- 1.1 Overview . . . 3
- 1.2 Problem Statement . . . 5
- 1.3 Contributions . . . 7
  - 1.3.1 Content Delivery as a Network Service . . . 7
  - 1.3.2 Integration of Information-Centric Networking Into Long-Term Evolution Mobile Networks . . . 8
  - 1.3.3 Offloading and Load Balancing using Information-Centric Networking . . . 9
  - 1.3.4 Edge Caching Strategies . . . 10
  - 1.3.5 Follow-Me Content Distribution Strategies . . . 11
  - 1.3.6 Summary of Contributions . . . 12
- 1.4 Thesis Outline . . . 13

2 Related Work **15**
- 2.1 Content Delivery Mechanisms . . . 15
  - 2.1.1 Host-Centric Paradigm . . . 16
  - 2.1.2 Multicast . . . 16
  - 2.1.3 Content Delivery Networks . . . 17
  - 2.1.4 Mobile Content Delivery Networks . . . 19
- 2.2 Information-Centric Networking . . . 20
  - 2.2.1 Naming and Security . . . 21
  - 2.2.2 Application Programming Interface (API) and Routing . . . 21
  - 2.2.3 Caching . . . 22
  - 2.2.4 Data-Oriented Network Architecture (DONA) . . . 22
  - 2.2.5 Publish-Subscribe Internet Routing Paradigm (PSIRP) . . . 23
  - 2.2.6 Network of Information (NetInf) . . . 23
  - 2.2.7 Content-Centric Networking (CCN) . . . 23

2.2.8 Named Data Networking (NDN) 24
2.3 Cloud Computing 25
2.3.1 Cloud Computing and its Principles 25
2.3.2 Mobile Cloud Networking 26
2.4 Mobile Networks 31
2.4.1 Long-Term Evolution 32
2.4.2 5th Generation (5G) Networks and Network Functions Virtualization (NFV) 33
2.4.3 Offloading and Load Balancing 34
2.5 Enhanced Caching Strategies 36
2.5.1 Edge Caching in Mobile Networks 36
2.5.2 Follow-Me Cloud 37
2.6 Summary 40

3 Evaluation Platforms 41
3.1 Cloud Computing Testbed 41
3.1.1 Hardware 42
3.1.2 Software 44
3.2 Emulation Platform 45
3.2.1 Emulation Details 45
3.3 Simulation Platform 46
3.3.1 Simulation Workflow 46
3.3.2 Extra Frameworks 46
3.4 Summary 48

4 Content Delivery as a Network Service 49
4.1 Motivation 49
4.2 Architecture and Mechanisms 50
4.2.1 Research Issues 50
4.2.2 High-Level Design 51
4.2.3 Low-Level Design and Implementation 54
4.3 Evaluation 60
4.3.1 Evaluation Scenarios 61
4.3.2 Evaluation Results 63
4.4 Summary 65

5 Integration of Information-Centric Networking into Long-Term Evolution Mobile Networks 67
5.1 Motivation 67
5.2 Architecture and Mechanisms 68
5.2.1 Traffic Interception 69
5.2.2 Traffic Filtering and HTTP Compatibility 70
5.2.3 Traffic Forwarding within the EPC 70
5.3 Evaluation 71

5.3.1 Experimentation Scenario and Parameters 71
5.3.2 Results 73
5.4 Summary 77

6 Offloading and Load Balancing using Information-Centric Networking 79
6.1 Motivation 79
6.2 Architecture and Mechanisms 80
6.3 Evaluation 83
6.3.1 Experimentation Scenario and Parameters 83
6.3.2 Results 86
6.4 Summary 90

7 Edge Caching Strategies 91
7.1 Motivation 92
7.2 Architecture and Mechanisms 93
7.2.1 Data Structures 94
7.2.2 Delete Queue Processing 94
7.3 Evaluation 96
7.3.1 Experimentation Scenarios and Parameters 96
7.3.2 Cache Hit and Miss Rates 98
7.3.3 Deletion Times 101
7.4 Summary 103

8 Follow-Me Cloud Content Distribution Strategies 105
8.1 Motivation 105
8.2 Architecture 106
8.3 Decision Mechanisms 109
8.3.1 "Where" Decisions 109
8.3.2 "What" Decisions 111
8.3.3 Other Decisions 113
8.4 Evaluation 114
8.4.1 Standalone Implementation Experiments 114
8.4.2 Testbed Experiments 123
8.4.3 Simulation Experiments 128
8.5 Summary 137

9 Conclusions and Outlook 139
9.1 Challenges Addressed in the Thesis 139
9.2 Thesis Summary 140
9.3 Future Work 142

Acronyms 144

Bibliography 151

List of Publications **165**

List of Figures

1.1 Summary of Thesis' Contributions 12

2.1 CDNs Architecture 17
2.2 Micro and Macro Datanceters in MCN [Bohnert et al., 2012] . . . 26
2.3 Mobile Cloud Networking (MCN) Architecture [Bohnert et al., 2012] 27
2.4 MCN Overall Architecture - Technical Lifecycle [Sousa et al., 2016] 28
2.5 MCN Architecture - Logical Entities [Sousa et al., 2016] 29
2.6 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) Evolved Packet System Architecture [Alcatel-Lucent, 2009] 32

3.1 Cloud Computing Testbed Servers 42
3.2 Cloud Computing Testbed Switches 43
3.3 Cloud Computing Testbed Low Level Architecture 43
3.4 ns-3 DCE Architecture 47
3.5 ns-3 LENA Topologt 47

4.1 ICNaaS Architecture 52
4.2 Service Orchestrator Diagram 55
4.3 Service Instance Initial Design, Deployment and Provisioning . . 56
4.4 Service Instance Scaling Out 56
4.5 Service Instance Scaling In 56
4.6 Service Instance Disposal 57
4.7 ICN Manager Registration/Initial Configuration 58
4.8 ICN Manager Configuration Update 59
4.9 CCN Server Diagram 59
4.10 FMC Manager Diagram 60
4.11 Communication between CCN Servers and the FMC Manager . . 60
4.12 ICNaaS 1-2-2 Tree Topology 61
4.13 ICNaaS 1-1 Tree Topology 63
4.14 ICN Cloudification CPU Load Comparison 64
4.15 Cloudified ICN Content Transfer Performance 64
4.16 ICNaaS Scaling Performance 65
4.17 ICNaaS Content Transfer Performance with Scaling 65

5.1 IP Packets Interception 69
5.2 CCNx Integration in EPS 71
5.3 Experimentation Scenario 72
5.4 Test Results - OFDMA Modulation 74
5.5 Test Results - SC-FDMA Demodulation 75
5.6 Test Results - RX PRACH 76
5.7 Test Results - Content Transfer Performance 77

6.1 CCNx Load Balancing 81
6.2 Test Scenario Architecture 84
6.3 Test Results - Scenario 1 87
6.4 Test Results - Scenario 2 87
6.5 Test Results - Scenario 3 88
6.6 Test Results - Content Sizes Impact 89
6.7 Test Results - LTE Usage 89

7.1 Hierarchical Caching in 3GPP LTE Mobile Network 92
7.2 Additional data structures for persistent storage 94
7.3 Delete Queue Processing 95
7.4 Network Scenario. 97
7.5 Hit and Miss Rates for Web Server and YouTube Scenarios 99
7.6 Hit and Miss Rates for Web Server and YouTube Scenarios (cont.) 100
7.7 Deletion times and number of deletions 102

8.1 Follow-Me Cloud Architecture 107
8.2 Evaluation testbed 114
8.3 Number of Files Correctly Selected for Migration 118
8.4 Volume of Content Correctly Selected for Migration 119
8.5 Percentage of cache-hits in the Normal Scenario 120
8.6 Percentage of cache-hits in the YouTube Scenario 120
8.7 Percentage of cache-hits in the WebServer Scenario 121
8.8 End-to-End Orchestration 123
8.9 User's Satisfaction Plot 126
8.10 Download Time in seconds for the WebServer Scenario 126
8.11 Download Time in seconds for the YouTube Scenario 127
8.12 Evaluation Setup 129
8.13 Optimal Solution in Ranking 133
8.14 Cache Hit Rates 134
8.15 Average Content Download Latency - Web Server 135
8.16 Average Content Download Latency - YouTube 135
8.17 Aggregated Core Bandwidth Usage 137
8.18 Execution Times of Components 137

List of Tables

2.1 MCN vs European Telecommunications Standards Institute (ETSI) NFV 30

4.1 ICNaaS Testbeds' Characteristics 62

6.1 Shaping Environments 84

7.1 Evaluation parameters. 97

8.1 Parameters for Content and Requests 115
8.2 Configuration Criteria in MCDM Algorithms 116
8.3 Knapsack Results for each Scenario 118
8.4 Processing Time of FMC Decision Algorithms 121
8.5 Processing Time of FMC Decision Algorithms 122
8.6 Processing Time of FMC Decision Algorithms 122
8.7 Weight Set #1 130
8.8 Weight Set #1 with Cost 130
8.9 Weight Set #2 130
8.10 Weight Set #2 with Cost 131
8.11 Evaluation Parameters 132

Preface

The work hereby presented was carried out as part of my doctoral degree program in co-supervision between the University of Bern, Switzerland and the University of Coimbra, Portugal. During the time of my employment at both institutions, I mainly worked in the EU FP7 Mobile Cloud Networking project. I owe a great thank you to all the public institutions involved and to all the responsible people that made this doctoral program and this project possible despite all the challenges and obstacles.

I would like to express my gratitude to both my supervisors, Prof. Dr. Edmundo Monteiro and Prof. Dr. Torsten Braun, who provided the most insightful advices and encouragements. It is due to their motivation to carry out outstanding research and publish results in high-level conferences and journals that my work could reach this stage. I would also like to leave a word of appreciation to Prof. Dr. Peter Kropf from the University of Neuchâtel for agreeing to be the co-examinator of this thesis and to Prof. Dr. Oscar Nierstrasz from the University of Bern for agreeing to be the president of the examination board.

Many thanks are also due to my fellow colleagues at the CDS group of the University of Bern and at the LCT group of the University of Coimbra. Special thanks go to Bruno Sousa, Carlos Anastasiades, David Palma, Florian Antonescu and Zan Li for all the cooperation over the years and the lengthy discussions on the several topics I addressed in this thesis. Likewise, I am thankful to BSc student René Gadow and MSc student Vitor Fonseca who contributed to selected implementations and evaluations described in this thesis during my supervision of their works. A last special thank you within the CDS group goes to Daniela Schroth, the group's secretary, who was always available and pleasantly helped with all my queries.

I would also like to acknowledge the exceptional cooperation within the EU FP7 Mobile Cloud Networking project and leave a thank you note to the colleagues from other institutions who significantly contributed to achieve the very good results that we did. In particular, I would like to express my recognition to Álvaro Rodríguez, Georgios Karagiannis, Lúcio Ferreira, Mohammad Valipoor, Mónica Antunes, Paolo Crosta, Santiago Ruiz and Tarik Taleb for all their availability, collaboration and support in the course of the project.

Finally, I am deeply grateful to my family and all my friends for their presence and support. I am honestly grateful to my parents Maria and Sérgio Gomes, and also to my grandparents Aurora Marques and Henrique Gomes for all the love and motivation they always gave me during my entire life. Special thanks go to Margarida "Maggie" Silva for her wonderful support and friendship throughout the writing of this thesis, to my very good friends Ana Frias, Bruno Santos, Carina Gomes, Francisco Matos, João Henriques, Joana Soares, Luís de Matos, Ricardo Ruivo and Sara Silva for always motivating me and for spending an amazing time together, and also to all the others who are not directly mentioned here but, in one way or the other, made this entire work possible.

Chapter 1

Introduction

The superordinate topic that spans across the different parts of this thesis is driven by the research question how to deliver content efficiently (using computing, storage and network resources in an efficient manner and delivering the best performance possible in terms of latency) to mobile users by leveraging newly developed concepts. This chapter briefly introduces into the related fields, namely content delivery, cloud computing and mobile networks. It also portrays briefly the fundamental challenges faced in the different parts of the this thesis, summarizes the essential contributions and outlines the course of the subsequent chapters.

1.1 Overview

Delivering information to users, namely content, has long become a subject of interest for the research community. Initially, as the adoption of the Internet was low, there was not much content available and the number of connected devices was small, which meant that requirements and challenges were only a few. Therefore, a simple host-centric paradigm on which a device would connect to another and get the content was enough to deliver information within reasonable quality parameters.

Later, with the evolution of technology, growing interest from the society for multimedia content and increasing adoption of network connected devices, requirements changed and improvements were required. Within those improvements, we may consider solid technologies that are widely used nowadays and that we still rely on to deliver content to users. Examples are Internet Protocol (IP) Multicast [Wu et al., 2001] and Content Delivery Networks (CDNs) [Qiu et al., 2001].

CDNs can be considered well integrated into the Internet, accounting by some estimates for over 40% of the current web traffic [DiVi Networks, 2012]. Generically, they consist of geographically distributed caches for content, which are populated and used to avoid that all the requests go to the content source, ensure that load is split among different resource pools and also that latencies to access content are low (greater geographic proximity between content source and users).

While CDNs are widely adopted and rely on the existing paradigm of the Internet, glsip Multicast poses more challenges and does not have the same adoption. While it is used often inside the operators' networks to stream multimedia content, it requires that all the users request the same content at the same time to leverage its main functionality – content is only sent one time by the source to a multicast group, and all the destination devices read it at the same time, thus saving bandwidth and ensuring efficient content delivery. Therefore, it does not scale and does not address the requirements of today's content delivery characteristics, neither in terms of experienced multimedia quality nor in terms of latency.

One may then think that CDNs are the way to go, and that has been the truth for a long period of time. However, CDNs have severe limitations derived from the host-centric paradigm they rely on. Also, in their current form, they are mostly used at the Internet core, in particular for inter-domain connections. That leaves load within the entire operator's network, and does not take advantage of edge technologies such as Mobile Edge Computing (MEC) [Ahmed and Ahmed, 2016]. Such limitations drive the research community to try and find new concepts that can overcome the challenges and adapt to the dynamic, demanding and distributed reality of today's networks, while increasing performance and saving costs for operators. In particular, mobile networks pose the greatest challenges for scalability and are the focus of most initiatives. This is explained with the boom in the usage of mobile devices and the increasing content demand of mobile users [Cisco Systems, 2015], which cause mobile networks to struggle the most to keep up with the pace of evolution.

As dealing with these issues poses several important research problems, a number of possible solutions have been proposed so far. One of them is the integration of CDNs into the Evolved Packet System (EPS) architecture used by the recently adopted LTE (or 4th Generation (4G)) [Astely et al., 2009] mobile networks standard. This integration is highly dependent on the proposed cloudification of the mobile networks with concepts such as NFV [Demestichas et al., 2013] to turn them into dynamic software components, and first enables the cloud principles related with scalability and flexibility, needed essentially to deal with very dynamic and demanding content transfers and user mobility.

However, the biggest advantage for content distribution is that, with cloudification, virtualization of resources is a reality. This allows different components, either radio or content related, to be co-located with each other and hence closer to the users. This concept of extending the current CDNs, which are called Over The Top (OTP) CDNs because of their limitation in range (do not go beyond the interconnection of providers into operator's networks), into the mobile networks and taking the cloud principles across the border of the big data centers is often called Mobile CDNs [Yousaf et al., 2013], as it allows data to be cached and deployed closer to the mobile user and data to be moved at the same time as the user.

Still, strategies based on CDNs are not the most efficient way to obtain content while not used together with other enhancements. Considering video, for instance, the current host-centric paradigm of searching for something and then access a specific resource on a specific server is not efficient at all for the users and the network. For the users, it is troublesome to search for content. For the network, it requires complex Domain Name System (DNS) strategies to find the appropriate content copy and does not exploit in-network caching, is not scalable after a while, lacks proper security and poorly handles mobility. Hence, the Information-Centric Networking (ICN) [Jacobson et al., 2009] paradigm emerges, providing a change in the current paradigm of requesting content. While currently, when a user wants to request a content object, it has to query a specific server for very specific content, the same is not needed with ICN.

With ICN, a user sends an Interest message that typically specifies the content producer and the name of the content object, depending on the naming scheme (content-centric paradigm). This message is then routed by ICN routers, which have forwarding tables based on content naming. These routers will find the content object (if available) and return it directly to the user from its original location or from one of the existing caches, without requiring any further interaction. This approach has multiple advantages over the traditional web paradigm, such as performance improvements, extended security, improved scalability mechanisms and support for seamless mobility.

Considering the previously described concepts and their key functionalities, several research challenges exist and will be addressed in this thesis. In the following sections (Section 1.2 and Section 1.3), these challenges are described in detail and the contributions of this thesis to tackle them are highlighted and briefly portrayed.

1.2 Problem Statement

In this thesis, the issue of enhancing the performance of content delivery in mobile networks is addressed. As different domains are involved, multiple technologies can be used and different parts of the architecture need to be considered, the challenges are also diverse. These challenges are described below.

The first challenge has to do with having an easily deployable service for content delivery, which scales and follows other important cloud principles such as on-demand instantiation. The challenge consists of automating all the mechanisms that set up the content delivery system (e.g. routes, configurations, etc.), virtualizing it without too much impact in its performance and also adjusting its scale according to the load in an efficient manner, which saves resources without compromising end user experience. With such a service, operators and other entities are able to use the cloud and quickly profit from all the functionalities provided by the work of this thesis, especially considering that there will be near 0-effort in bringing the concepts to their networks and that the service will adjust in real-time to different loads, needs and topologies. In this thesis, a framework to have ICN integrated with mobile networks by exploiting the cloud concepts is proposed (c.f.

Section 1.3.1), including mechanisms to be automatically deployed, automatically adapted to the load and flexible enough to support the remaining contributions.

The second challenge, of which all the next challenges depend on, is how to integrate ICN into the LTE EPS, i.e. its core network architecture, without affecting its functionality and avoiding all the details that can be seen as a violation of 3GPP specifications that both carriers and hardware vendors use today. This integration enables new functionalities and allows leveraging new concepts, but performance can also be affected and any architecture or mechanisms need to ensure it is not affected significantly. Moreover, the tradeoff between impact and gain must be satisfactory to foster the adoption by mobile operators and therefore ensure that in the future the contributions of this thesis can be used in real world scenarios. The contributions highlighted in Section 1.3.2 include an architecture and mechanisms to have ICN together with LTE without a major impact in the underlying infrastructure, showing that this integration is feasible and has benefits towards the end goal of enhancing content delivery performance.

The third challenge is related to the usage of radio resources and how to perform load balancing of different radio technologies. Every time a user requests content, content has to somehow be delivered to that user. If no coordinated management is done between the network and content delivery mechanisms, no optimization is done and resources are not used efficiently. Coordinated management of resources can then be considered as an enabler for the user to get the most performance and satisfaction, but needs mechanisms and algorithms to decide how to prioritize content over different links, to be aware when communication is going to happen (specially relevant when mobility is a reality), when to aggregate content transmissions and how to combine different radio technologies to deliver a single piece of content as quickly and efficiently as possible. The contributions described in Section 1.3.3 provide strategies to use multiple radio technologies simultaneously with ICN and without depending on the underlying technologies, achieving not only offloading from one technology to the other but also load balancing.

The fourth challenge concerns caching strategies. As a number of features need to be considered in topologies such as the tree topology of LTE, such as hierarchical caching levels, policies for optimized caching and caches distribution are required. Here the emphasis goes for the caching locations (can depend on load, energy, pricing, etc.), the decision process to know when to cache (and how to cache) and the factors to take into account when prioritizing content to be cached. Moreover, content eviction strategies must exist to make sure that storage space is efficiently used towards the goal of increasing performance and at the same time saving resources. The contributions of this thesis towards edge caching (c.f. Section 1.3.4) range from tree distribution strategies to content eviction at a single cache location based on a number of factors.

The last challenge, and perhaps one of the most demanding, is how to deal with user mobility. When the user moves from one location to the other, a number of actions may happen/be required: the cached private content (or public if considering group mobility, as usually users only have interest on the same content if it is public) should move with the user, content fetching requests should be kept alive and fulfilled at the new location and a new ICN infrastructure may need to be instantiated on-demand [Antonescu et al., 2013]. Strategies to trigger content migration and resources allocation need to be researched, as they will greatly influence both the performance and the efficiency of resources usage, namely the load of the network infrastructure. In this thesis, mechanisms were developed in order to efficiently copy content across locations depending on the mobility of users, their interests and other meaningful criteria (c.f. Section 1.3.5).

In the end, summarizing all these research challenges, the main challenge behind this work can be described as to develop algorithms, mechanisms and functionalities to ensure that content is delivered efficiently with awareness of the underlying conditions, while a the same time saving on resources and increasing performance for end users.

1.3 Contributions

In the course of this thesis, multiple contributions are motivated, described and evaluated. These all fit into the single goal of enhancing the performance of content delivery to end users in mobile networks, and achieve that goal by thoroughly facing the challenges presented in the previous section. They range from creating mechanisms and an architecture to support new technologies and concepts at a lower level to strategies and algorithms that work at a higher level to provide more efficient mechanisms for content delivery while increasing performance and improving end user experience.

1.3.1 Content Delivery as a Network Service

The first contribution is towards the usage of Future Internet (FI) concepts to have content delivery as a part of the network, i.e. a network flexible service, and achieving better efficiency and performance. Hence, concepts such as ICN, NFV [Demestichas et al., 2013] and cloud computing arise to make that service a reality that can fit into today's systems and require an integrated architecture with mechanisms to support them together and enable the usage of their functionalities while taking advantage of their flexibility, scalability and performance benefits. In this thesis it is envisioned that ICN can be made up of virtualized network components and may be deployed anywhere, anytime and without any hassle. This calls out for cloud computing concepts, in particular the automatic elasticity needed to deal with dynamic loads and the ease of deployment coming from on-demand services instantiation in the cloud. Hence, the corresponding contribution is mainly a service named Information-Centric Networking as a Service (ICNaaS). This service was developed as part of the Seventh Framework Programme (FP7) MCN project, and it brings all the advantages related with the integration of ICN with LTE together with integration with other cloud services, which are helpful to support the goals of this thesis. Namely, it integrates with a monitoring service (useful to get metrics for automated scaling decisions) and with a mobility prediction service (useful to get mobility prediction information for mobility-related caching strategies).

In Chapter 4 these contributions are described in detail. First, the motivation is described based on the research challenges pointed out in Section 1.2, which include the self-configuration of the ICN service (topology, content routes, etc.) and algorithms to efficiently scale the service according to different loads. After, the internal architecture of the ICN service and its mechanisms are detailed. Namely, a component that instantiates the service on-demand and scales using efficient algorithms that analyze its load makes it an effortless process to have a ICN layer (cloud principles are followed). Also, using a different component, the underlying network topology together with content providers information are taken into account to make decisions that allow the configuration of ICN routers with appropriate content routes, policies and components. Afterwards, the service, its mechanisms and the overall framework are evaluated using a cloud computing testbed described in Section 3.1 to verify their functionality, performance and capabilities. Experiments are conducted in different testbed scenarios, evaluating standalone metrics and also end-to-end performance. Results show that performance for end users can be improved when scalability is required, and also that virtualizing components does not have a major impact in terms of overall performance.

1.3.2 Integration of Information-Centric Networking Into Long-Term Evolution Mobile Networks

Having established a whole framework to have ICN in the cloud and part of a NFV-enabled architecture, and considering that the goal is to enhance performance of content delivery in mobile networks, the next logical step is to make sure it can be integrated into LTE mobile networks. That requires an integrated architecture and new mechanisms, and those need to be envisioned. Namely, the support for NFV and Cloud Radio Access Network (C-RAN) [Haberland et al., 2013] needs to be exploited, and extra information coming from the network and the integration of components needs to be used to guarantee that specifications are met. Using these concepts, a modified architecture and integration mechanisms, the integration of ICN and LTE is achieved and proven feasible. That, together with the contributions described in subsection 1.3.1, completes the low level support for advanced and enhanced content delivery mechanisms that run in higher layers and deliver the much needed improvements envisioned by the work developed in this thesis.

Later in this thesis, in Chapter 5, these contributions are outlined. Initially, the motivation is depicted according to the problems mentioned in Section 1.2. This obviously includes the need to have ICN within LTE mobile networks to achieve improvements, but also addressing the limitations of 3GPP specifications towards extra functionalities, performance impact minimization and the existence of end user performance improvement gains. To handle the challenges, an architecture and corresponding mechanisms are depicted. This architecture intercepts IP traffic at the LTE base station dual stack. However, this traffic has to be efficiently filtered in order to separate all ICN-related traffic from other traffic, and then to forward matching traffic to ICN components while other traffic follows the pre-defined path without changes. Mechanisms are proposed to filter that traffic and make the processing of ICN messages more efficient, while minimizing the impact on the LTE base stations and ensuring functions of the existing 3GPP-defined architecture can run when a request is cached and not routed through the predefined path.

To evaluate the proposed architecture and its mechanisms, experiments are run in different scenarios. These are carried out in a simple testbed environment, using a reference ICN implementation and a LTE network emulator – OpenAirInterface (OAI) [Nikaein et al., 2014]). The goal is to verify the impact on the performance of LTE-related functionalities is low while proving that performance can be improved when comparing to existing technologies. Results help to validate that goal, as impact is not very relevant in terms of processing and performance gains for users are quite good.

1.3.3 Offloading and Load Balancing using Information-Centric Networking

The contributions in the previous sections already provide an environment where ICN and LTE mobile networks operate together, but users may have more connectivity options available at a given time. For example, they may also have Wi-Fi. This brings more options to improve performance and, at the same time, reduce costs for mobile operators and save resources. As different radio technologies often have different architectures and protocols, a transparent and seamless way to exploit their simultaneous usage is required. Therefore, the contributions in this section aim at using ICN as an overlay to support offloading and load balancing over different technologies. This higher layer support for these functionalities ensures that they can be used no matter the constraints imposed by the underlying technologies, while leveraging extra information about content deliveries to aggregate transmissions and split content chunks over different links. Such split can be efficiently planned so that performance and efficient usage of links are maximized.

In Chapter 6 this contribution is illustrated in greater depth. First, a motivation is presented based on the assumptions defined in the previous paragraph. Later on, the architecture and load balancing/offloading mechanisms are described. These include heuristic-based algorithms that support multiple mobile network technologies simultaneously and perform load balancing/offloading among them, thus achieving improvements over existing solutions that are limited by underlying technologies. Existing works, such as Wi-Fi offloading, which are partially specified by 3GPP, are considered as a first step. However, improvements in terms of technology selection, load balancing execution and ICN forwarding strategies are proposed to enhance the results. They deliver seamless connectivity that takes into account the conditions and status of mobile networks, and not only split traffic but also distribute the load to attain greater speeds, efficient usage of resources and battery savings for mobile devices.

To evaluate the proposed mechanisms and included algorithms, simulation is performed. Using the simulation platform described in Section 3.3, different link conditions were tested and performance testes were made in order to compare different approaches. The results show that performance can be improved in most scenarios and that a high percentage of LTE resources can be traded (and thus saved) for much cheaper Wi-Fi resources.

1.3.4 Edge Caching Strategies

Having established a framework to have new concepts that improve content delivery in mobile networks, higher layer mechanisms are required to deliver further enhancements. A proposed solution is to deliver caching at the edge of mobile networks, but that creates challenges that were previously mentioned in Section 1.2. Namely, the distribution of content from its source to caches at the edge of mobile networks, prioritization of content caching and management of cache usage. As edge-cached content will yield the greatest benefits if it is cached for longer periods of time, persistent caching at the edge becomes a key factor. Also, a distribution tree that takes into account different caching levels and different availability of resources is required. The proposed mechanisms manage caching at the edge considering the constraints in terms of storage and efficient satisfaction of users requests. Moreover, they create a content distribution tree with caches from the source and the core of the mobile network until the edge, ensuring that storage space is efficiently used at all times and that the storage of more popular content is prioritized as the edge is closer, i.e. in higher proximity to end users, where it will yield the greatest benefits.

In Chapter 7 these mechanisms are depicted and evaluated. In Section 7.1 they are motivated based on the benefits they bring. After, in Section 7.2, they are described in greater detail. They include efficient placement and replacement algorithms, which were developed in order to maintain a threshold of persistent memory utilization (low availability of resources at the edge of mobile networks). By taking into account popularity over time, a delete queue is maintained and deletion operations are triggered whenever thresholds are reached. This approach applies together with the typical volatile caching, and when used in a tree topology such as the one common for LTE, it brings clear benefits by having the most requested content at the leafs or other lower levels of the tree. Then, in Section 7.3, an evaluation is performed using the simulation platform described in Section 3.3. This evaluation includes experiments in different traffic scenarios and using different parameters, aiming at evaluating the impact on cache hit rates while being able to efficiently use storage. It is shown that cache hit rates increase, and that performance overhead for the devices is not major. Results also allow to conclude that performance for end users will be greatly improved, as popular content will be served from nearby edge caches instead of remote copies with much higher latency.

1.3.5 Follow-Me Content Distribution Strategies

Considering the previously described contributions towards edge caching, one has to assume that there is much room for improvement when the environment is mobile and users are very dynamic, either in terms of request patterns, either in terms of mobility across different geographic locations. In order to greatly maximize user satisfaction by means of lower latencies (high cache hit rates) and to have more optimization of the usage of cache storage space, strategies have to consider not only the interests of current end users at a given location, but also interests of users coming from other locations. As popularity of content and users interest on content have both locality and homophily factors, the first step is to detect or predict user mobility, which can be achieved using algorithms and mechanisms already present in the literature. Using this mobility information together with other criteria such as popularity of content requested by the moving users and available storage space, decisions can be made to copy content from one location to others and thus optimize the availability of content towards the requests of users in a much quicker way. Therefore, the concept of content following the users interested in it becomes a reality, and content within caches is always more optimized towards the requests and preferences of nearby users. The contributions presented in this thesis towards this goal consist of algorithms to perform these decisions and mechanisms to support them.

In Chapter 8 these algorithms and mechanisms are presented. First, they are motivated by taking into account the high dynamicity of mobile networks and the very high degree of mobility of their users. Later on, they are depicted in detail and their design is justified. These algorithms include existing and improved decision techniques, which are bound to be efficient and run efficiently. Decisions are: if migration should occur, where should it happen to, what should it include, when should it be done, and how should it happen. After answering these questions and having made the decisions, mechanisms obtain content from the nearest cache or, in the worst case scenario, copy it from the source location. Afterwards, an evaluation is carried out to assess which decisions algorithms perform better, and also how they affect the network and the mobile users. The goal is to evaluate the improvements brought by these content distributions strategies for the network and for end users, while evaluating their negative impact. The evaluation is carried out in multiple experimentation scenarios, both in simulation using the platform described in Section 3.3 and in a cloud computing testbed environment described in Section 3.1. Results demonstrate that user satisfaction with content delivery is much improved, mostly because cache hit rates are higher than with typical edge caching strategies and because available resources are exploited efficiently.

1.3.6 Summary of Contributions

In Fig. 1.1 a summary of the previously described contributions is depicted.

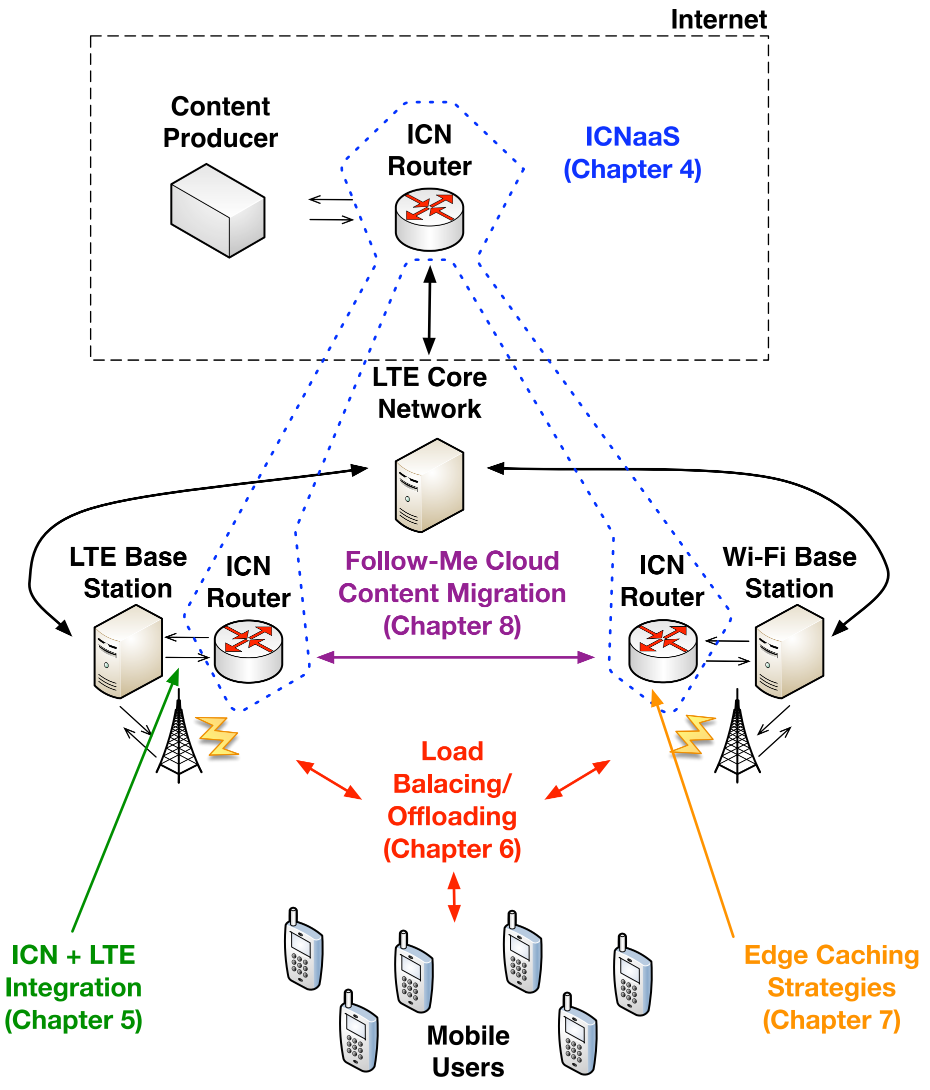

Internet
Content Producer
ICN Router
ICNaaS (Chapter 4)
LTE Core Network
LTE Base Station
ICN Router
Follow-Me Cloud Content Migration (Chapter 8)
ICN Router
Wi-Fi Base Station
Load Balacing/ Offloading (Chapter 6)
ICN + LTE Integration (Chapter 5)
Mobile Users
Edge Caching Strategies (Chapter 7)

Figure 1.1: Summary of Thesis' Contributions

In the diagram, an LTE EPS architecture (more details in Section 2.4) is represented with all the contributions highlighted for better understanding. In a nutshell, the goal is that mobile users get content from content producers in the most efficient, fast and reliable way possible. This involves a close interaction between the components of LTE mobile networks and content delivery systems. A first contribution aims at having a cloud-based service for the flexible and on-demand deployment of ICNaaS, a service for content delivery with a new innovative paradigm that brings several benefits such as auto-scalability, flexible and automated deployments, integration transparency, legacy compatibility, etc. (see Chapter 4). In order to achieve the desired integration between LTE mobile networks and ICNaaS, which enables benefits such as edge caching and lower latencies for content retrieval, mechanisms that support it without performance impact and standards violation are required and described in Chapter 5. As this integration becomes a reality, and it can be even used together with other non-3GPP radio technologies such as Wi-Fi, the question on how to perform offloading and load balancing between the different available technologies arises. The objective is to provide it in a transparent way, without relying on underlying protocols and their known limitations. Hence, a load balancing scheme based on ICN is proposed in Chapter 6. Once this integrated base system is set, two other contributions are proposed to enhance content delivery and exploit its new features. A first one contains improved edge caching strategies based on hierarchical caching and cache usage optimization towards content popularity (Chapter 7, while the second advances further and proposes mechanisms that change content distribution among edge caches according to parameters such as user mobility, content popularity, availability of storage resources, etc. (see Chapter 8). Both of them aim at increasing cache hit ratios while using available storage and network resources in an efficient manner, which lead to improved experience for end users in terms of latencies to download content and also cost savings for mobile operators.

1.4 Thesis Outline

The structure of this thesis is outlined below on a chapter by chapter basis.

Chapter 2 introduces and discusses the most important and most significant related works of the domains approached in this thesis.

Chapter 3 describes the key evaluation platforms used for the experiments performed to evaluate the proposed contributions.

Chapter 4 portrays the framework to support new FI concepts that greatly improve content delivery in mobile networks. This framework exploits ICN, Cloud Computing and NFV.

Chapter 5 describes an architecture and corresponding mechanisms for the integration of ICN in LTE mobile networks, evaluating its feasibility and advantages.

Chapter 6 includes the description and evaluation of mechanisms to support offloading and load balancing in different radio technologies regardless of the underlying architectures and protocols. These rely on ICN and related forwarding strategies to split content chunks and aggregate content requests.

Chapter 7 depicts mechanisms to improve edge caching and the content distribution from the source (core) until the edge of mobile networks. It includes cache management and optimization.

Chapter 8 surveys decision techniques to make decisions towards content migration between caches based on user mobility, content popularity and other key factors.

These decisions aim at greatly improving experience of end users by converging to the optimized usage of caches quicker, thus going further from simple edge caching strategies.

Chapter 9 concludes the thesis and discusses both ongoing and future work in the domains that were approached in the course of this work.

Chapter 2

Related Work

This chapter discusses the most important and most significant related work we rely upon for our contributions presented in the subsequent chapters of this thesis.

The chapter commences with Section 2.1 portraying the current mechanisms for content delivery in both fixed and mobile networks. Section 2.2 then discusses ICN, a new paradigm and concept that changes focus from hosts to content/services and aims at improving content delivery both for the users and network operators. It also describes the different implementations and approaches, comparing them and highlighting benefits/drawbacks to enable the choice of the most appropriate one for our work. Later, Section 2.3 describes the cloud computing concept and its principles together with additional concepts and improvements under development to leverage the existing technology. One of them, MCN, is highlighted as it brings one of the pillars for the work to be carried out in this thesis - an infrastructure that intersects both mobile networks and cloud computing.

Afterwards, as this work focus on the usage of mobile networks, Section 2.4 describes the current technologies used for mobile networks, recent advances in these networks and also offloading/load balancing mechanisms. Finally, in Section 2.5, strategies to enhance content delivery in mobile networks are described. These include edge caching, and they focus especially on advanced strategies to populate those edge caches and mechanisms to ensure they are used efficiently.

2.1 Content Delivery Mechanisms

As previously described in Section 1, for years there were developments in terms of mechanisms to deliver content in networks, either fixed or mobile. From different paradigms to clever usage of resources, leveraging of technologies and their functionalities led to a number of solutions that greatly improved the distribution of content and its delivery to end users. In the subsections below, the main existing approaches are detailed.

2.1.1 Host-Centric Paradigm

The current Internet concept has it focus on the hosts, meaning it is host-centric [Xylomenos et al., 2014]. Users access resources at a given location (servers), and routing is done based on IPs. Therefore, routes are found using algorithms such as Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) [Griffin et al., 2002] to access particular hosts in a given network, which then may have the resources requested by users. This paradigm, although rather inefficient for the usage given today to mobile and fixed networks focus, which now have to deal with great amounts of content that create massive loads, is still heavily used and solutions exist to make it more scalable and transparent for end users. Those solutions will still be used for many years to come, but they are deemed to be replaced altogether as more evolved ones become standards.

2.1.2 Multicast

Multicast streaming is a one-to-many relationship between a server and the clients receiving the stream [Wu et al., 2001]. With a multicast stream, the server streams to a multicast IP address on the network, and all clients receive the same stream by subscribing to the IP address. Hence, content can be delivered from a single broadcast publishing point.

As there is only one stream from the server regardless of the number of clients receiving the stream, a multicast stream requires the same amount of bandwidth as a single unicast stream containing the same content. Using a multicast stream then preserves network bandwidth and can be useful for low bandwidth networks. However, issues with multicast are several-fold.

First of all, it requires special routers as nodes on the network to pass the single data stream on. It has to build up a tree of these special routers, so that it (or the network) can program those routers so that only a single data stream is passed between them. Obviously, only multicast routers can be linked in this tree. This is generally known as tunneling - the special routers tunnel the multicast data stream between them over the normal internet. Then, each receiver must be able to identify its nearest multicast router so that it can receive a unicast of the data stream from that router. With that, the network may require modifications so that multicast and regular network traffic coexist effectively.

Second, as there is a single stream, the audience must all be interested in the same content at the same time, similarly to typical television channel streams. Therefore, growing markets such as Video On Demand (VoD) [Thouin and Coates, 2007] cannot be explored using this concept. This limits the scope of multicast and its adoption, and it is therefore not widely used.

2.1.3 Content Delivery Networks

With the tremendous growth of web traffic in the Internet, efficient distribution of content has long become a big concern within both the research community and industry (network operators, content providers, etc.). In the early stages of the web, caching strategies were considered to be the most efficient approach to reduce network traffic and the latency for end users. These were typically implemented in proxy servers, which are nothing but local servers that sit between the users and Internet servers to transparently handle all the web requests and cache frequently accessed content. As today, the main challenge was how to efficiently use the available storage space in order to minimize latency while maximizing cost savings and traffic reduction at the core.

More recently, these strategies have evolved and are now handled transparently at the core of the Internet by CDNs providers [Qiu et al., 2001]. Companies such as Akamai [Akamai, 2015] and CloudFlare [CloudFlare, 2015] have their own geographically distributed infrastructure and work together with content providers and network operators to efficiently deliver content to end users all over the globe. As depicted in Fig. 2.1, their CDNs architecture essentially consists of an origin server, i.e. a point where the content is originally deployed, and a number of geographically distributed proxy servers where popular content is replicated. These proxy servers may serve a small number to a large number of users, depending on the size of the region and its population. Therefore, an hierarchical architecture is usually followed to ensure a better replication of the content, increased performance and higher robustness of the network. In addition, CDNs give the content provider a degree of protection from Denial of Service (DoS) [Liu, 2009] attacks by using their large distributed server infrastructure to absorb the attack traffic.

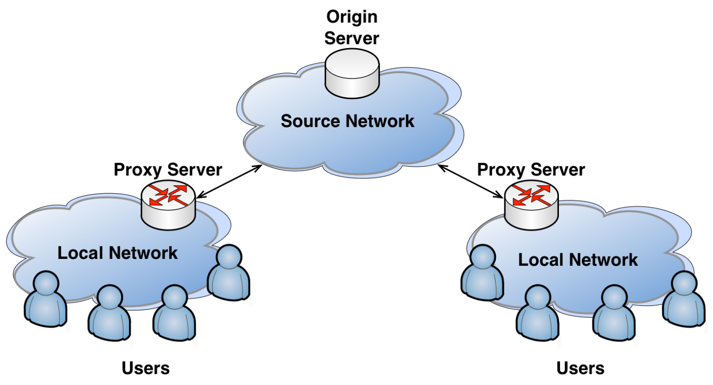


Figure 2.1: CDNs Architecture

The way hierarchical CDNs architectures [Ni et al., 2003; Yang and Chi, 2007] work is very straight forward. DNS [Shaikh et al., 2001] or anycast routing [Park and Macker, 1999] are used to route the requests of a particular hostname/IP to the nearest/fastest proxy server, and one of two things may happen: the requested content is cached and is delivered back to the user, or the requested content is not available and it has to be requested from another server. In the latter option, the worst case scenario means that the content has to be fetched all the way from the origin server, but it can also be retrieved from a nearby proxy server. Either way, content can be directly sent to the user from its source or sent to the proxy server that the user queried in the first place, which then forwards it to the user.

In CDNs architecture and system designs, the following key challenges exist:

- Which content to cache?
- How many times should content be replicated in the network? And where to put the copies?
- Which proxy server should be selected when a user makes a request?
- How content should be routed from its source to the user?

Several proposals already exist to deal with these challenges, and they are described in more detail below.

Cached Content and Replication

Deciding on content to cache/replicate at different servers is one of the biggest challenges for CDNs, and has been the target of a number of previous works. Most common algorithms attempt to maximize the hit ratio for stored content, which will minimize the cost and latency of delivering content to local users. Therefore, an object is stored in the cache when there is a cache miss, and there is the possibility that existing objects need to be replaced to free space for the new object. Objects selected to be replaced are typically the ones with lowest hit count or least recently requested, but a combination of both metrics and also file size can be used [Balamash and Krunz, 2004].

Several algorithms [De Vleeschauwer and Robinson, 2011; Serpanos et al., 2000] take into account information about content objects by tracking every single request for them. Then, when there is a cache miss, policies allow the server to decide whether to cache an object or not. Decision is made based on information such as access frequency, date and time of last access, size and free space at the cache, and the function score is compared to the score of objects already in cache. Moreover, principles such as temporality, i.e. when is something important, can be explored when not caching full objects, allowing for greater efficiency in cache usage by caching object's parts that are more important for users in a given period of time, or even predicted to be required soon in the future (e.g. multimedia content for streaming).

Many optimizations can still be made on this matter, and recent works attempt at tackling them. For instance, some proposals consider network information as a cost model for object caching and others automatically adapt their servers' location and capacity dynamically to face different workloads and usage patterns from the users [Chen et al., 2012; Tran and Tavanapong, 2005]. However, with the proliferation of mobile networks and devices, CDNs and their existing models become too static and inefficient to deal with the increasing amount of content requests and very high mobility rate of users.

Cache Placement, Cache Selection and Routing

Cache placement, cache selection and routing are CDNs design aspects that were analyzed and addressed extensively in the literature. However, problems such as the optimal number of caches and where they should be located in the network have been proved to be NP-complete problems for most network topologies [Gupta and Tang, 2006]. Therefore, proposals to address the problem are either based on specific topologies such as the tree topology [Nuggehalli et al., 2006; Li et al., 1999] or near-optimal solutions for general topologies [Taha and Kamal, 2003].

As for server selection and routing, existing strategies select the closest cache, the cache server with the lowest load or simply use shortest path routing [Bakiras, 2005]. The latter can be either based on static metrics such as hop count or alternatively dynamic metrics that assess the delay of a given path. Among all the possibilities, selection and routing is still inefficient and can become quite complex in certain network topologies with a great amount of users. Therefore, it is desirable that selection is not only transparent to the user but also more efficient at the cache servers, avoiding as much as possible the collection of metrics and complex decisions.

From the description of current CDNs architectures and mechanisms, one may conclude that although CDNs are widely used and deliver great performance together with cost savings, they are still rather limited mainly by the focus on host-centric paradigms and also due to their cache placement usually far from the networks edge. Drawbacks include reduced to inexistent mobility support, low adaptability to dynamic networks, low scalability (expensive), very little security for end users and inefficient performance gains if considering the entire path from content providers to end users [Vakali and Pallis, 2003]. Therefore, other proposals need to be considered, either by extending current CDNs or shifting paradigm to a more enhanced architecture.

2.1.4 Mobile Content Delivery Networks

A current trend in development and in particular research is to integrate cache servers belonging to CDNs with the mobile operator's core network or even with its access and backhaul network, in order to move caches server and content sources closer to mobile clients.

Although it has the potential to address a few of the shortcomings of traditional OTP CDNs, namely the network distance from end users, it still has limitations inherent to the usage of the same architecture. First, it still relies on the same host-centric paradigm with its associated drawbacks. Second, it is arguable that using protocols such as Hyper Text Transfer Protocol (HTTP) – the basis of CDNs operation – is feasible at resource-constrained environments such as the edge of mobile networks.

Other optimizations of mobile CDNs [Yousaf et al., 2013] come in the form of mobile device's mobility pattern detection and the knowledge of mobile operator's transport network topology. In the literature, is is proven that the delivery path may turn into a sub-optimal and costly delivery path [Siris et al., 2013] if changes occur due to network dynamicity and user mobility. In this regard, research results prove the gain in the deployment of CDNs distributed cache servers and show the transport costs during content delivery in dependency of the cache location and the transport network characteristics [Liebsch et al., 2012; Liebsch and Yousaf, 2013]. Those results also motivate the relocation of the content delivery endpoint in the network (CDNs Serving Point) to keep delivery paths short and associated transport costs low, but then issues such as session continuity arise and that brings a new set of problems.

2.2 Information-Centric Networking

With the several limitations of the current host-centric paradigm, such as reduced mobility support, high vulnerability to security threats and inefficient routing based on host addresses, a new concept was first proposed by Van Jacobson in 2009 - ICN [Jacobson et al., 2009; Pavlou, 2011; Ahlgren et al., 2012]. ICN can be considered a new FI paradigm, that shifts the focus from hosts to information. Contrary to the host-centric paradigm where users have to query a specific host in the network for a given resource, with ICN the focus is no longer in hosts or routing between hosts. Indeed, information is stored in nodes spread across the network (routers) and routing is done based on the name of the information (content, services, etc.). When a user wants to request a piece of information (object), it will simply query the network to which it is attached to and nearby routers will take care of the request, route it if necessary and deliver the object back to the user.

As ICN is a fairly recent concept, which is still under definition, it is important to describe the elements implemented by the existing architectures that are considered essential for their design. These elements are the Named Data Object (NDO), Naming and Security, API, Routing and Forwarding, and Caching, and they will be detailed below.

Before describing the elements that are part of ICN's architectures, it is important to take note of some of the relevant nomenclature changes from current host-centric paradigms. One of the changes is the concept of "source", which before was the host where the content comes from and now becomes the producer/publisher of the content (origin server in CDNs). ICN also assumes caching, and caching can be either volatile or persistent. Volatile caches, usually stored in Random Access Memory (RAM) are always named caches, whereas persistent caches are called repositories.

The NDO is one of the most important aspects of ICN, and it consists of an abstraction of all types of objects available on the Internet. Examples are multimedia objects (e.g. videos, images, music, etc.), web pages and static/live multimedia streams. With this abstraction of naming all the objects in a similar way, every single object available, either persistently or not, is uniquely identified by its name, regardless of its type, location and how it was transmitted.

2.2.1 Naming and Security

Naming and Security are closely related to the NDO abstraction and are also a key part of the ICN concept, which means that they are as important to ICN as the host identification is to the current host-centric paradigm. Two important aspects have to be considered: uniquely identifying each object and ensure that each object can be signed and trusted regardless of the router that delivered it. Therefore, each NDO can be either a complete file or a chunk and will always have an unique name - the prefix together with a signature. To ensure that a signature exists and an object can be trusted (related to a trusted source), two different schemes can be considered: a flat namespace and a hierarchical approach. The former is a self-certifying scheme where each NDO can be verified without a third-party validation, and is done by hashing the content of the object and embedding the hash in the name of the NDO. The latter uses a hierarchical scheme similarly structured to that of an Uniform Resource Locator (URL), which also enables the aggregation of routing information. Opposed to the flat namespace scheme, it is human readable, easily manipulated and understood by users. However, it requires that a validation is done by another entity in the hierarchy, and that may be a drawback in terms of additional overhead. Still, both approaches greatly improved the security of ICN networks, and together with other mechanisms bring several security advantages over the host-centric paradigm [Smetters and Jacobson, 2009].

2.2.2 API and Routing

As for the API, most ICN approaches assume that content has to be published by the producer/publisher, become available in the network and only afterwards may be requested by the user. This is a straightforward process, and it is based on the NDO of the content object. However, other approaches use a different method, where the consumer first registers interest for a particular NDO and the consumer is only later notified when the NDO is available for retrieval. To support both these methods, a clearly defined API is needed both to request and publish content objects, and that API should also specify the protocols to be used in order to communicate with the different architecture components.

The routing and forwarding features provided by ICN have also two different methods, depending in the used naming scheme and the usage of aggregation. One of the methods (coupled approach) uses a Name Resolution Service (NRS) that provides the location of repositories in the network as well as the NDOs of objects they contain. When a user sends a request, the NRS resolves the NDO to a number of sources, routes the request to those sources and one of them will send the NDO's chunks back to the user. The other method (decoupled approach) does not need a component such as the NRS, but rather directly routes the requests from the user to the available sources in the network based on the NDO's prefix (hierarchical). When a source receives the request, it will also deliver the content chunks back to the user through the routers along the opposite path. For both methods, different routing algorithms can be used and each ICN approach considers their own. Also, as routing is done based on content names and sessions are not established between the source of the content and the user interested on it, ICN brings major advantages in terms of mobility support, i.e. a user can move to a new location and still continue transferring chunks of the desired content object without any sort of interruption [Kim et al., 2012; Ravindran et al., 2012].

2.2.3 Caching

The caching aspect is also a key feature of the ICN concept. ICN proposes that every router has a cache, and every node in the network is considered a router, no matter if it is a user device, an actual router or a content source. With such feature, ICN brings in-network caching to another the entire network, and that may be leveraged to improve and create new strategies for edge caching (described later in this thesis). Also, this cache supports all the types of content, which means it can be used by an object and therefore improve several applications. Yet, all the approaches typically do not have advanced strategies, and routers store every content object chunks that go through it until the cache space is filled, at the point where a Least Recently Used (LRU) replacement policy is used.

As previously discussed, ICN is a concept under development and definition that at this point only sets some guidelines for the aspects to consider when developing an ICN architecture. Five example approaches will be presented, as well as their chosen solutions for handling the most relevant topics of ICN.

2.2.4 DONA

In DONA's [Koponen et al., 2007] approach, the producers register available content objects into the resolution infrastructure (set of NRS nodes). When a user wants an object, it sends a request to the network (named Find packet) with the name of that object. This request is routed by one of the NRS nodes until it reaches the object's source, which can either be the publisher or a router with the object in its cache. Afterwards, the object can be routed back to the user using the reverse path or using a more optimized route. However, if an optimized (shorter) route is used, the caching feature of ICN is not fully exploited as the object will be cached in a smaller number of routers. Another important aspect is that the registration of objects in NRS nodes has expiration times, meaning that producers have to keep renewing the registration of their objects periodically. However, they can also opt to register their prefix instead of every object, significantly reducing the complexity of the operation.

2.2.5 PSIRP

In PSIRP [Dominguez et al., 2011], users have to register their interest (subscribe) on content so that the network will make the content available (publish). This means that matching between the publications and the subscriptions is made by a rendezvous system - when a producer wants to make a NDO available, it publishes it and when the user wants to get an object, it subscribes to it. Matching is done by the rendezvous system, and the system is able to store subscriptions for a certain time until the content becomes available. If content is published and becomes available before the expiration of the subscription, it will be sent back to the user (subscriber). Otherwise, if the subscription is not renewed, it will be simply discarded.

2.2.6 NetInf

NetInf [Dannewitz et al., 2013] implements the two methods for routing described by ICN, the NRS and the name-based routing. However, it uses a hybrid system that leverages the benefits of both approaches to select the optimal available source while allowing the aggregation of routing information. When a producer wants to make content available, it can either register the NDO in the NRS or use a routing protocol to announce the new routing information to the routers in the network. Even if the producer does not register the object in the NRS, a router with that NDO may do it and both methods will still be available for the users. As with DONA, a request (in this case named Get packet) is transmitted by the user to the network, and it will use named-based routing to until it reaches a router with the NDO in its cache or the NDO producer. As well as with DONA, this approach has a major drawback if using the NRS approach - optimal path communication between the source and the user will mean that caching is not fully exploited.

2.2.7 CCN

In the CCN [Jacobson et al., 2009; Perino and Varvello, 2011] concept, all nodes are able to publish NDOs, either by having their own repository or by publishing to other repositories available in the network. The routing protocols are name-based, and allow that any router in the network is able to publish or find NDOs. As CCN uses an hierarchical naming scheme, the routing protocols can also perform aggregation and therefore reduce overhead.

CCN is also very efficient in terms of security, as it uses a scheme of public key cryptography where trust in keys can be achieved by a mechanism based on the Public Key Infrastructure (PKI). When a user wants a content object, it sends a request (Interest packet) that is forwarded by routers until it expires or until the content object is found. As with other approaches, content objects can be found either by reaching the producer of the content or a router with the content object in its cache - the Content Store (CS). To keep track of Interest packets and forward them between routers, two different structures are used. The Pending Interest Table (PIT) registers all the Interest packets processed by a router, and an entry is kept until the Interest expires or the content is retrieved back to the router that requested it. If an Interest is not resolved by the CS or was not already in the PIT, it is processed using the Forwarding Information Base (FIB). The FIB contains information about name-based routing, and allow the Interest packet to be sent (using unicast or multicast) to the next router by using the NDO's prefix. When content is returned from the source by using the reverse path, the entry in the PIT is cleared and content is forwarded to the original requester (another router).

Apart of the described features, CCN recently introduced custom forward strategies for the routing of Interest packets [Zhang et al., 2015]. These strategies make possible the existence of load balancing, redundancy and dynamic path selection depending on different conditions/metrics, enabling greater performance, flexibility, robustness and savings of network resources.

2.2.8 NDN

The NDN [Yuan et al., 2012] is a FI architecture project funded by the United States of America's National Science Foundation that aims to develop a new Internet architecture that can leverage the strengths and address the weaknesses of the Internet's current host-based paradigm. It was first based on CCN, but forked and followed its own path with its own research directions. Namely, its design is more optimized towards performance (e.g. improved PIT queuing algorithms to control its size). Despite being at a very early stage, it is currently similar to other approaches described above: it names data instead of their locations, it secures the contents rather that the communication channel and decouples trust in content from trust in hosts. Also, it studies challenges for the FI such as scalability, performance, trust models, security and privacy.

Based on the analysis of the previous approaches, we decided to select CCN as the base for our work at an initial stage. The main reasons are: it is a broad approach with support for different characteristics and features, the community is quite large (great support and frequent updates), its source is open for the academic community and, as it exists for some years, it is already quite stable and suitable to use in more realistic environments. At a later stage, and for simulation purposes, NDN was also used to the the availability of its own simulation framework (see Section 3.3).

2.3 Cloud Computing

2.3.1 Cloud Computing and its Principles

Cloud computing [Foster et al., 2008; Rimal et al., 2009] can be described as a model for enabling ubiquitous network access to a shared pool of configurable computing resources.

It essentially provides solutions for users and enterprises to store and process their data in third-party data centers. Such solutions rely on the sharing of physical resources (using virtualization) to achieve efficiency and resource scaling, similar to what happens with electricity grids. Many different types of services can run in this shared infrastructures, with huge advantages that became key features:

- **Agility** is amplified by the possibility of easily adjusting infrastructure resources.
- **Cost savings** for entities that use resources. If a public cloud model is used, infrastructures belong to a third party entity and are used simultaneously by different customers. This means that resources are used more efficiently and customers will only pay for their actual needs, not for the entire infrastructure. Likewise, management and operational responsibilities are almost always shifted from the entities to the cloud provider, and thus operational expenditures are much lower.
- **Device and location independence** is a reality, as users can access anything, anywhere with any device they got.
- **Maintenance** of cloud computing applications is much simpler as they can be accessed from anywhere and remain exactly the same.
- **Multitenancy** allows the sharing of resources, which translate into cost savings (as mentioned), optimal dimensioning of resources and efficiency of utilization of resources.
- **Performance** increases as better infrastructure can be present and always dimensioned for the load posed by cloud applications.
- **Reliability** is much better due to the usage of redundant Data Centers (DCs), which can be a key point for critical business applications (e.g. banking, utility companies, etc.).
- **Scalability and elasticity** are a reality due to on-demand request of resources and automatic adaption to different work loads.
- **Security** can be improved because data is stored at a central location with increased security measures, but privacy and control issues can also be raised as concerns.

The definition provided by the National Institute of Standards and Technology (NIST) [Mell and Grance, 2011] includes the following five essential characteristics:

- **"On-demand self-service:** A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider."

- **"Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations)."
- **"Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand."
- **"Rapid elasticity:** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear unlimited and can be appropriated in any quantity at any time."
- **"Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service."

2.3.2 Mobile Cloud Networking

The basic assumption of MCN, a European Union FP7 Large-Scale research project [MCN Consortium, 2015] where the concepts of this thesis will also be validated, is the existence of a cloudified Radio Access Network (RAN), cloudified mobile core networks as well as Micro and Macro Data Centers (DCs) that support a cloud-based infrastructure (depicted and described in Fig. 2.2).

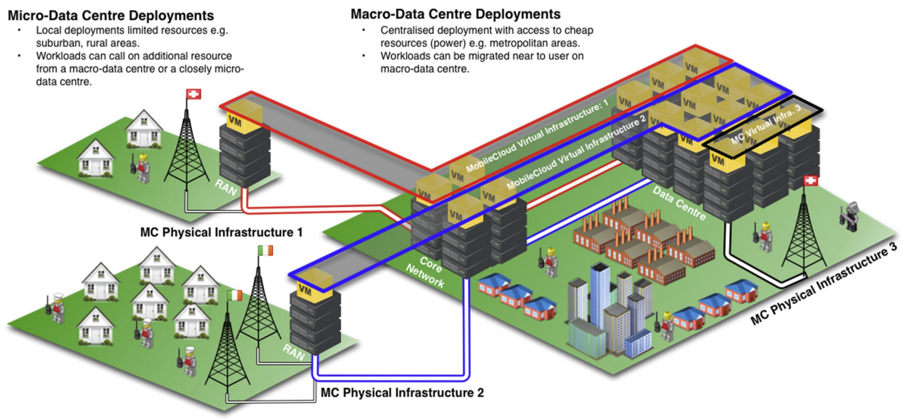

Micro-Data Centre Deployments
• Local deployments limited resources e.g. suburban, rural areas.
• Workloads can call on additional resource from a macro-data centre or a closely micro-data centre.
Macro-Data Centre Deployments
• Centralised deployment with access to cheap resources (power) e.g. metropolitan areas.
• Workloads can be migrated near to user on macro-data centre.
VM
RAN
MobileCloud Virtual Infrastructure: 1
MobileCloud Virtual Infrastructure 2
MC Virtual Infra. 3
Data Centre
Core Network
MC Physical Infrastructure 1
MC Physical Infrastructure 2
MC Physical Infrastructure 3

Figure 2.2: Micro and Macro Datanceters in MCN [Bohnert et al., 2012]

Ownership and operation of the RAN, mobile core networks and both these types of DCs can be related to several different entities. For instance, a Mobile Network Operator (MNO) can own and control the entire infrastructure and therefore fully manage its components and technologies. In another example, a different company/entity may act as a provider of the services provided by the infrastructure without owning or even operating any of its physical components. Such company would have to sign business agreements with the owner(s) of the physical infrastructure, which can for instance be MNOs that operate in the desired geographic areas or infrastructure providers that own/control DCs in strategic locations, either micro or macro.

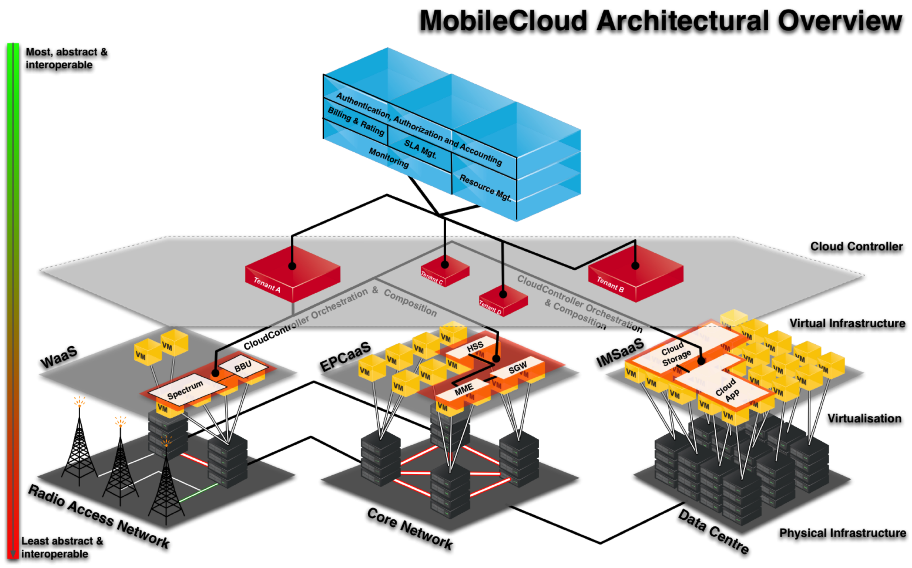


Figure 2.3: MCN Architecture [Bohnert et al., 2012]

The innovation here is that a MCN provider exploits the MCN architecture (see Fig. 2.3) to compose and operate virtual end-to-end infrastructures and platforms on top of several different and separated physical infrastructures belonging to different providers, thus providing an end-to-end service architecture that has no geographic boundaries and brings new potential benefits.

One of these benefits is the previously mentioned virtualization of the entire mobile network's infrastructure (except the actual radio antennas) by exploring concepts such as NFV (see subsection 2.4.2). With that virtualization and subsequent cloudification, many enhancements and new features can be explored and, besides cost savings, performance and efficiency can be much improved from the end user perspective.

MCN has its own cloud architecture, based on related approaches and also standards, but with differences to simplify certain functions and at the same time provide higher flexibility for the project's goals. This architecture is described below.

MCN Overall Architecture

The MCN overall architecture [Edmonds et al., 2013][Edmonds et al., 2015] has used a service-oriented architecture as a guideline. With this service-based architecture, the key entities and lifecycle are common to all the services within the MCN framework. The goals of this architecture are essentially to save costs by using cheap infrastructure and adjusting the load, while at the same time not being dependent on underlying infrastructure platforms and also exploiting new and innovative concepts that bring major improvements to the provided services.

While the technical part is guided and governed by the business phase decisions and agreements between providers, it has a common lifecycle for all services (see Fig. 2.4).

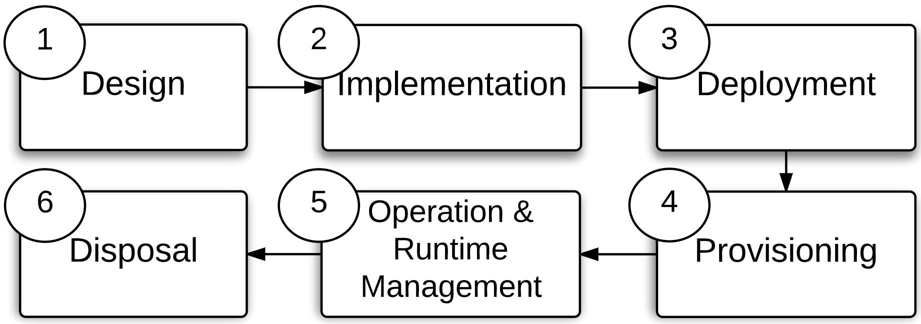


Figure 2.4: MCN Overall Architecture - Technical Lifecycle [Sousa et al., 2016]

- **Design**: It is the initial phase for of the design of each service's architecture, implementation, deployment, provisioning and operation solutions.
- **Implementation**: When the designed architecture, functions, interfaces, controllers, APIs and others are actually implemented.
- **Deployment**: Creation of the required resources in the cloud infrastructure, such as creation of any Virtual Machine (VM) and extra components (e.g. networks, routers, etc.) in order to make the service available for use.
- **Provisioning**: Set up of the service environment (e.g. interfaces, networks, routers, components). Activation of the service so that the users can access it and profit from it.
- **Operation and Runtime Management**: This phase includes scaling and reconfiguration of service instance components.
- **Disposal**: Release and destruction of all components and resources associated with the service.

An important challenge for MCN was to present a consistent, common and standards based control/management interface. To achieve this all components are compliant to the Open Cloud Computing Interface (OCCI) standard [Edmonds et al., 2012]. The following key MCN architectural entities, illustrated in Fig. 2.5, are described below.

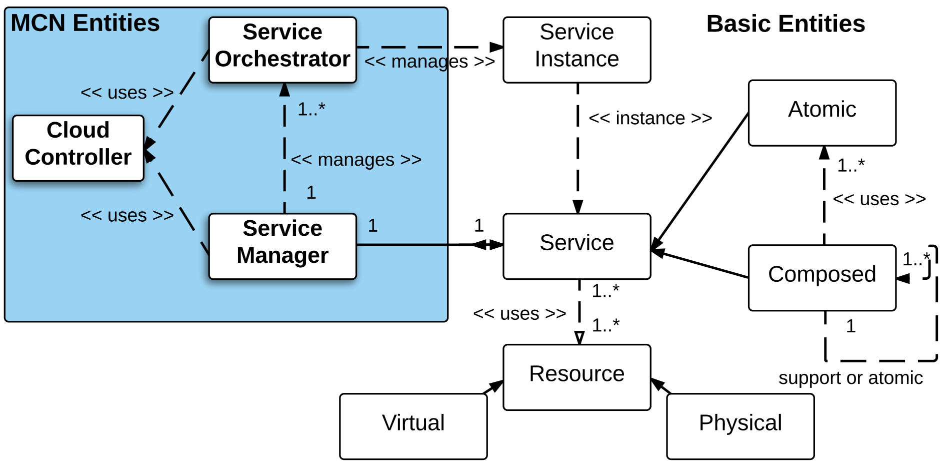


Figure 2.5: MCN Architecture - Logical Entities [Sousa et al., 2016]

Service Manager (SM): provides an external interface that presents one or more service types that can be instantiated by an Enterprise End User (EEU). It is responsible for managing a collection of service orchestrators and by deploying then whenever an EEU requests an instance of one of them.

Service Orchestrator (SO): The SO oversees the end-to-end orchestration and composition of a service instance. Once it has been created by the SM and it is running, it manages the instance and its components in an isolated fashion (separated from other instances). The key logic of this component so as its base is shared among MCN services, but a part of it is domain and service specific. Namely, the metrics for scaling decisions, the algorithms to take care of those decisions and the generation of templates (Service Template Graph (STG) with the service specifics and corresponding Infrastructure Template Graph (ITG) mapping to infrastructure) for all the components to be deployed for each instance of the service.

Cloud Controller (CC): abstracts from specific technologies that are used in the technical reference implementation. It allows the SO to use different cloud platforms, either public or private, no matter what the technologies and APIs are. As it is OCCI-based, it's considered to follow the industry standards and can be easily adapted to support even more infrastructures. All its interfaces enable the interaction of the SO with the cloud platforms for any type of operations.

MCN and NFV

MCN and its architecture development started prior to the publication of ETSI NFV standards [ETSI, 2016]. Although the approaches have different terminology, they share the same concepts to provide Virtual Network Functions (VNFs) and services.

According to Table 2.1, the previously described MCN architecture and its components support the same functionalities as NFV. The NFV infrastructure is provided

Table 2.1: MCN vs ETSI NFV

ETSI NFV	**MCN**
NFVI	Distributed and heterogenous cloud infrastructures.
VIM	CC
VNF Managers	SO
VNF Descriptors	STG and ITG
OSS/BSS	MCN Support Services
NFV Orchestrator	SM
Element Management	Each service has its own logic to manage each of its components.

with different types of software and hardware, Network Functions Virtualization Infrastructure (NFVI) is provided to the Management and Orchestration layers using the CC and from there to the users using the Virtualized Infrastructure Manager (VIM) interface.

The remaining approach also maps. MCN services (or VNFs in the case of NFV) have their own SOs (VNFs Managers) to orchestrate the necessary resources in terms of computing, storage and networking via the CC and using STG/ITG (VNFs Descriptors). The SO also coordinates with other services, such as support services (Operations Support System (OSS)/Business Support System (BSS)), and APIs are exposed to EEUs by the SM to allow the deployment of composed services (Network Services as per NFV). Moreover, each service has its own logic to manage its internal components (Element Management).

Despite the similarities, there is a clear distinction between MCN and NFV. MCN handles multi-tenancy and service deployment in distributed cloud infrastructures, while NFV focuses on more limited scenarios, such as multi-tenant hosting scenarios and privately owned infrastructures. This means that MCN is more heterogeneous, and can be exposed to public clouds together with any other distributed private clouds, i.e. does not map to a single infrastructure.

MCN Services

MCN consists of 12 main services. While some are base services, others can be considered support services as they are not essential for the mobile cloud infrastructure to be running and serving users. Both these types of services are described below:

- **Infrastructure as a Service (IaaS)** is the main service in MCN. It provides the cloud infrastructure, either private or public, to allocate resources for all the other services.
- **Radio Access Network as a Service (RANaaS)** is the service responsible for providing the virtualized components that interact with radio antennas and support glslte base stations in the cloud. Users with mobile devices connect to the network using the components provided by this service.

- **Evolved Packet Core as a Service (EPCaaS)** is the service that provides the Evolved Packet Core (EPC) for LTE mobile networks. Without it, RANaaS does not function as it contains most of the functionalities within the LTE architecture (see Section 2.4).
- **IP Multimedia Subsystem as a Service (IMSaaS)** is the service that provides voice and other multimedia services in LTE mobile networks. Its components interface and rely on EPCaaS to operate.
- **Digital Signage System as a Service (DSSaaS)** provides functionalities to stream multimedia content such as video in a standardized and efficient manner. It relies on other services for the actual distribution of content.
- **Load Balancing as a Service (LBaaS)** is a support service to provide load balancing between multiple instances of other services' components. It is limited to common protocols, but it is very simple to use and is often available from IaaS.
- **Monitoring as a Service (MaaS)** is a support service to monitor specific metrics of other services and provide both processed information on request and automated triggers to other service that want to monitor their load. This information is then used to provide fault tolerance and scalability.
- **Domain Name System as a Service (DNSaaS)** is a support service to provide name resolution using existing DNS implementations but with scalability and simple setup.
- **Authentication, Authorization and Accounting as a Service (AAAaaS)** is a support service to provide accounting, and then authentication and authorization mechanisms to ensure secure communication between services and components.
- **Rating, Charging and Billing as a Service (RCBaaS)** is a support service used to provide the cloud principle of pay-as-you-go, providing bills according to the resources used by the service instances of each EEU.
- **Mobility Prediction as a Service (MOBaaS)** is a support service responsible for the prediction of users' mobility in mobile networks. It uses information from past mobility received from EPCaaS and uses algorithms to predict users' movements within a certain accuracy.
- **ICNaaS**, as the name states, provides ICN as a support service for content delivery in mobile networks. This service is the base of this work and it is described further with details and evaluation in Chapter 4. In MCN it is widely used by DSSaaS to deliver its streamed content.

2.4 Mobile Networks

A variety of mobile networks technologies have surfaced in the last decades, from Wi-Fi to Worldwide Interoperability for Microwave Access (WiMAX) [Ghosh et al., 2005] and from Global System for Mobile communications (GSM) to LTE. The latter, as a widely adopted mobile network technology, is currently the focus of many research works and initiatives. It is therefore the main technology explored in the work of this thesis.

2.4.1 Long-Term Evolution

LTE [Astely et al., 2009], commonly referred as 4G, is a standard for wireless communication of high-speed data for mobile phones and similar devices. It is based on the General Packet Radio Service (GPRS)/Enhanced Data rates for Global Evolution (EDGE) [Furuskar et al., 1999] and Universal Mobile Telecommunications System (UMTS)/High Speed Packet data Access (HSPA) [Rinne et al., 2009] network technologies, increasing the capacity and speed using a different radio interface together with core network improvements. The standard is developed by the 3GPP and is specified in a set of documents that contain its initial specification together with later enhancements.

LTE is mostly regarded as an upgrade for operators of 3rd Generation (3G)/UMTS networks, and by using different frequencies, bands, modulations and bandwidths, it delivers great speed improvements together with a more efficient usage of radio spectrum. In this work, the key point is its EPS architecture and how content delivery mechanisms can be integrated with it, providing placement of functionalities closer to the edge (as far as its base stations) and using network metrics and other information to transform content delivery in mobile networks. This architecture is briefly described in the next section.

Architecture

The EPS architecture, depicted in Fig. 2.6, includes the User Equipment (UE), the RAN, the EPC and other applications. Its components are the following:

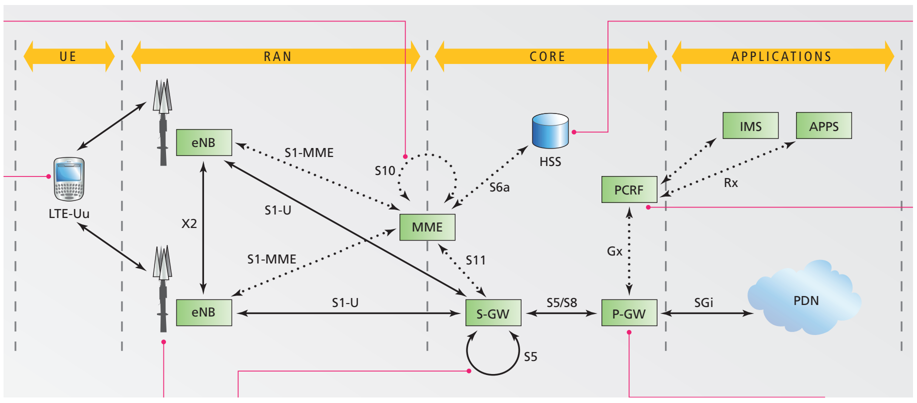


Figure 2.6: 3GPP LTE Evolved Packet System Architecture [Alcatel-Lucent, 2009]

- **UE:** This is the device at the end user side, containing support for the LTE uplink and downlink interfaces and a Universal Subscriber Identity Module (USIM) for authentication.
- **Evolved Node B (eNB):** It is the LTE radio base station. It handles radio resource management, forwarding of the data plane to serving gateway and signaling of the control plane via the Mobility Management Entity (MME).
- **MME:** It is a key component for the control plane and signaling in the architecture. It handles Non-Access Stratum (NAS) signaling that includes bearer management and attachments, security, mobility signaling, selection of PDN Gateway (P-GW) and Serving Gateway (S-GW), handovers, roaming and authentication.
- **S-GW:** It is a key component for the data plane after the eNB. It handles local mobility (between local eNBs), lawful interception, traffic forwarding and routing, Quality of Service (QoS) and charging.
- **P-GW:** It is a key component for the data plane between a group of S-GWs and the Packet Data Network (PDN). It also provides lawful interception, packet filtering, traffic forwarding and IP address allocations.
- **Home Subscriber Server (HSS):** It is the component that is responsible to handle subscribers data. The QoS profile, roaming information, Access Point Names (APNs) and other important data are stored here.
- **Policy and Charging Rules Function (PCRF):** It is the component that enables interfacing with network applications. It establishes policies together with the P-GW, which specify rules for applications depending on the subscriber profile.

2.4.2 5G Networks and NFV

With the improvement of LTE towards LTE-Advanced and a 5G [Agyapong et al., 2014] of mobile networks getting ready to surface, other concepts are getting attention from the research community and explored to become part of the new 5G standard. One of them is NFV [Demestichas et al., 2013], a network architecture concept that proposes using the technologies of Information Technology (IT) virtualization to virtualize entire classes of network node functions into building blocks that may be connected to create communication services.

NFV relies upon, but differs from, traditional server-virtualization techniques, such as those used in enterprise IT. A VNF may consist of one or more virtual machines running different software and processes, on top of standard high-volume servers, switches and storage, or even cloud computing infrastructure, instead of having custom hardware appliances for each network function. For example, a virtual session border controller could be deployed to protect a network without the typical cost and complexity of obtaining and installing physical units. Other examples of VNFs include virtualized load balancers, firewalls, intrusion detection devices and

Wide Area Network (WAN) accelerators.

The initial perception of NFV was that virtualized capability should be implemented in data centers. This approach works in many – but not all – cases. NFV presumes and emphasizes the widest possible flexibility as to the physical location of the virtualized functions. Ideally, therefore, virtualized functions should be located where they will be the most effective and least expensive. That means a service provider should be free to locate NFV in all possible locations, from the data center to the network node to the customer premises. This approach, known as distributed NFV, has been emphasized from the beginning as NFV was being developed and standardized, and is prominent in the recently released NFV Industry Specification Group documents.

Altogether, NFV is a key requirement to bring efficient content delivery mechanisms such as ICN to LTE mobile networks. One of the NFV concepts that enables that is C-RAN) [NGMN, 2013][Haberland et al., 2013], which brings the possibility to virtualize the entire LTE radio infrastructure except the antennas. It extends the cloud computing concept (described in the subsection below) to the RAN, and at the same time explores the modularity of the components together with the usage of general purpose hardware infrastructure.

At the same time, MEC, as a key 5G network enabling technique, enables a cloud-based IT service environment at the edge of mobile networks, much like the micro DC concept used in MCN. It provides benefits such as ultra-low latency, precise positional awareness and agile applications, being foreseen as an essential building block for the 5G mobile networks. Moreover, it also plays a key role in supporting new business solutions based on smart devices and Machine-to-Machine (M2M) communication. From services and applications to content, they can all be accelerated by taking advantage from increased responsiveness at the edge of the network.

Such facts are enablers for the co-location of ICN functionalities in proximity to (and integrated with) eNBs of LTE mobile networks. There are two reasons for them to be enablers. First, with legacy networks, co-location of software components within its specialized hardware platforms would not be feasible. Second, as the eNBs' components are implemented directly in hardware through Field Programmable Gate Arrays (FPGAs), there are no modular software interfaces to intercept traffic at a given point of processing and redirect it to the ICN layer.

2.4.3 Offloading and Load Balancing

Load balancing in terms of bandwidth and offloading to other networks are hot topics in the literature today, as networks become overloaded with traffic and devices need to use multiple networks simultaneously or even be offloaded from one network technology to another.

Such mechanisms are typically not very well defined, and become complex to meet a set of requirements and be used efficiently. In LTE networks, offloading to Wi-Fi can be partially achieved by following 3GPP-defined standards, and must be supported by the core of the network. As defined by 3GPP [Martiquet, 2012][Ali, 2013], the usage of an Evolved Packet Data Gateway (ePDG) in the core network enables the operator to integrate 3GPP technologies with non 3GPP technologies (such as Wi-Fi) and to perform offloading from one to another. This standard, however, does not specify strategies to decide on offloading or to eventually combine both technologies and perform load balancing when delivering content. To decide which mobile devices to offload, algorithms maximize user satisfaction by increasing the average data rate in a heterogeneous network while still using the resources with high efficiency [Skowron et al., 2012].

Although relevant proposals that deal with actual load balancing by using both radio technologies simultaneously do not exist, works with similar goals exist and can be exploited to reach a possible solution. An interesting approach downloads the content object first by a device using the cellular connection and then shares the content via ad-hoc Wi-Fi with nearby mobile devices [Detti et al., 2012]. This proposal minimizes the usage of the cellular network and mostly improves the satisfaction of users, but it also has major drawbacks. First, users have to be close to each other and must request the same content object within the same time frame. Then, the device sharing the content object will drain its battery empty very quickly due to extra transmissions and it may also be not powerful enough to deal with a big number of users. Moreover, existing Wi-Fi infrastructures would not be harnessed. This point is very important for operators nowadays, as they made big investments to have Wi-Fi hotspots widely available, either by having them deployed at public places or by enabling a second Wi-Fi network in their customers' home routers.

Another approach that indirectly deals with this issue considers that two concepts are met: Enhanced Data rates for Global Evolution (NC) and ICN [Montpetit et al., 2012]. By using inherent caching of ICN together with its capacity to send Interest messages over multiple interfaces, NC principles are exploited: different segments (using linear combinations) can be delivered over multiple paths and then reconstructed to have the requested content objects. This approach does in fact use multiple links simultaneously, but does not take into account that links have different capacities and loads, and also does not have the flexibility to deal with such constraints. That is because the usage of the links depends on the number of available segments at each path, which can differ for multiple reasons and will not have a relation to the policy for load balancing. It has, however, the advantage of using ICN, abstracting the network layer and making it usable with different network stacks (e.g. Wi-Fi and LTE) at the same time.

Nonetheless, there are proposals that describe strategies for multi-path forwarding of CCN Interest messages without NC [Rossini and Rossi, 2013]. The referred proposal aims at improving reliability of CCN, by using strategies that increase the probability of obtaining the content object without too much impact on the network. Although the proposal is quite valid for reliability purposes and may in fact be used to combine different radio technologies to make sure the user gets the content, it does not address the topic of load balancing of data delivery after multiple Interest messages are sent over different interfaces. Therefore, a research challenge for this thesis can be derived from this shortcoming.

2.5 Enhanced Caching Strategies

A cache can be defined as temporary storage of web documents, such as web pages and multimedia content, to reduce bandwidth usage, server load, and perceived lag. It works by storing copies of objects passing through it, so that subsequent requests can be satisfied directly from the cache and avoid longer paths. As caches typically need to be quite fast, their size cannot be large and efficient mechanisms are needed to store the highest-value content while replacing previously cached content that bears no benefit to users or networks. As caching has the biggest impact on performance when it is closer to the content requester (lower latency), strategies are needed to ensure that caches can be placed at the edge of the networks and always have content with significance to nearby users. In the subsections below, these features are described in more detail.

2.5.1 Edge Caching in Mobile Networks

According to Cisco Systems [Savic, 2011], one of the key issues to tackle the extreme demand for content is how to perform content caching and delivery closer to the edge of mobile networks. Current proposals, however, only address caching as close to the edge as the EPC because it provides greater flexibility coming from a full IP stack and resource availability.

Zhu et al. [Zhu et al., 2013] describe EPCache, a video caching framework that provides solutions to decide what to cache and how to cache, namely where caches should be located in the EPC architecture. Ramanan et al. [Ramanan et al., 2013] looked at HTTP caching at the EPC S-GWs. By caching content based on its popularity and type, they aim at achieving optimal performance of the caching server. However, they do not evaluate how caching and tapping into the EPC interfaces affect processing resources and latency. In general, we may conclude that, although such proposals are valid, they fail at least at addressing issues posed by this integration. Moreover, they do not reach the edge and stay only within the EPC, thus not fully exploiting the benefits of having caches within the mobile networks.

Other proposals look at the problem from a theoretical point of view, overlooking technical challenges and rather approaching the problem of deciding what to cache and where to do it to be effective. The work from Gaito et al. [Gaito et al., 2013] is a good example of such an approach. It proposes how to decide on placement of data kiosks, which are nothing else but caches located within a cell. With that decided, they state that the number of cells can be minimized and traffic can be reduced. In fact, they propose mechanisms to use mobility patterns as a way of predicting people's movement and hence where resources need to be deployed and caching needs to be available.

The aforementioned proposals may seem good enough in terms of performance and cost savings, and even easier to manage in terms of integration, but studies demonstrate that performance can be significantly improved and cost savings can be much higher if caching is done even closer to the edge. EPC caching introduces a good amount of bandwidth and cost savings, but the most savings can be obtained by caching at LTE base stations [Sarkissian, 2013], i.e. at the so called eNBs. With caching at eNBs, and because less backhaul capacity is needed, Operational Expenditures (OPEX) are reduced by 27% to 36%, the largest amount of savings in all the business cases selected by LSI Corporation for its study. Also, latency is significantly impacted by this type of caching and backhaul traffic experiences a major reduction, allowing faster content downloads whether they are cached or not.

As for edge caching based on ICN, it has been subject to extensive research in recent years. For instance, it has been shown that popular content tends to be cached at the leafs of the network [Psaras et al., 2011] and, therefore, allocating more storage resources to edge routers than to core routers is beneficial in terms of performance [Psaras et al., 2012] and energy consumption [Lee et al., 2010]. To avoid redundant caching, several strategies have been proposed such as limiting the number of cached copies along the path to one [Eum et al., 2012], probabilistic caching based on distance from the content source [Psaras et al., 2012], or apply network coding to ensure content diversity caches [Wu et al., 2013; Wang et al., 2014a]. Other approaches are based on coordination for content deletion, e.g., pushing deleted content one-level upstream the caching hierarchy [Li et al., 2012] or adapting the number of cached chunks based on the file's popularity [Cho et al., 2012].

However, storage in the CS is limited and cached content is only available for a limited amount of time to support faster retransmissions in case of collisions and to synchronize multiple concurrent (or slightly time shifted) requests. If content should be available for a longer time, e.g., for Delay-Tolerant Networks (DTNs) [Fall and Farrell, 2008] or to ensure high availability and performance similar to CDNs, it should be persistently stored and the current proposals fail not only to address that type of storage but also to have efficient cache replacement policies.

2.5.2 Follow-Me Cloud

With the dynamicity of today's mobile networks in terms of topology and user mobility, strategies that maintain content cached nearby users that may be interested on it are increasingly required. Therefore, one may develop strategies that take into account the mobility of users to decide on placement/migration content at mobile networks edges.

The need for these strategies is supported by recent studies [Wittie et al., 2010; Wang et al., 2014b], which report that user interests in social media content have significant impact on its locality and homophily characteristics. This means that people geographically close to each other may have common or similar interests of content objects (locality) and also the users are both clustered by regions and interests (homophily), which is also called “birds-of-a-feather” effect [Choudhury et al., 2010].

At the moment only a few strategies like this exist, and they still have quite a few shortcomings. One proposal assumes that mobility of the user is considered for services placement and scaling [Antonescu et al., 2013]. In this case, orchestration of distributed cloud services is done by predicting user mobility, i.e. more or less resources are allocated if the system predicts that users will move to/from the location of each small DC. However, migration of services from one location to another is not considered.

In this direction, one very important concept towards migration strategies, Follow-Me Cloud (FMC), was first proposed by Taleb et al. [Taleb and Ksentini, 2013b]. It essentially considers that small DCs are present closer to the edge of mobile networks and proposes that services are deployed in close geographical proximity to users. Hence, when users move to a different location, those services should be migrated and follow the user. To handle the decision, several different models can be used. An analytical model based on Markov Decision Processes is proposed [Taleb and Ksentini, 2013a; Ksentini et al., 2014]. That model considers that user positions must be found in order to have services instantiated in the optimal DC, and a random walk mobility model is used to determine such position in the future by giving a probability to the user being in a neighbor cell. Migrations are then triggered when the user is n hops away from the previous optimal data center, a system that is modeled using Continuous-Time Markov Chains and aggregating states that show the same behavior to reduce the decision space. Also, when considering if data migration should be done, factors such as class of the user, load policies, service migration costs and service migration duration need to be analyzed, with cost and service disruption to be minimized, and user to be connected to the optimal DC as often as possible. This approach is valid and reflects a good solution to the problem of service migration, but some issues remain. For instance, the system only has a 1-dimensional mobility model, and a single destination has to be considered when the user may be moving and passing by multiple destinations in the selected period of time. At the same time, nothing is said about which services to migrate and whether group mobility is more than single user mobility, as one may argue that the cost to migrate services for a single user may not be justified. Likewise, the proposals only address services and do not deal with specificities of content.

As far as strategies to deal with content replication and migration are concerned, works as the one from Liu et al. [Liu et al., 2011] look at the problem in Peer-to-Peer (P2P) networks from the provider perspective and, because they are hierarchical in that case, they can be extrapolated to other types of networks such as LTE. Considering a typical hierarchical CDNs architecture, the problem of how to distribute content among multiple network nodes to avoid network traffic at higher layers of the hierarchy and improve latency arises. To handle this problem, decisions have to be made in order to copy (or migrate) content from higher layer caches to lower layer caches. Such decisions are made based on the cost of migration and the frequency that a particular content object was requested at a given location in a certain time interval. The objective is to maintain cache storage space used as much as possible and deliver content with the greatest possible efficiency, which is a NP-complete problem. With the proposed strategies, authors demonstrate that it is possible to get a high benefit value (saved overhead) in most cases. However, one may argue that such a system may not scale when the hierarchy is complex (as when we approach the edge of the network) because the number of nodes is very large and the system requires global information about content in every cache and requests made at every location. Meanwhile, decisions to copy content are only made with a low frequency. In very dynamic networks, in which users keep moving and the topology may be changing, it is important that decisions are made more often and before mobility happens (proactive). If decisions are not made quickly and before users move, possible benefits are not fully exploited and the solution may not bring advantages over existing and far less complex approaches.

Another proposal, by Vasilakos et al. [Vasilakos et al., 2012a], uses proactive migration strategies for content, i.e. migration is triggered when it is predicted that the user will move to a neighbor location. Using proxies at most 1 hop away from the user, authors propose that pre-fetching of subscribed content is done between proxies whenever it is predicted that the user will disconnect and move to the region of another proxy. At the same time, it is assumed that all the possible destinations of the user are known, as well as the probability of each of them. With this knowledge, a decision has to be made in order to select the destination proxies for the content while minimizing cost (migration cost and cache storage) and maximizing benefit (latency and cache hit ratios). Results show that delay gains are high, but authors fail to address issues that would impact large networks. First, selection of proxies is made with a very small number of criteria that always take the same weights. More advanced selection mechanisms could be used to improve the benefit of the decision. Second, the paper does not discuss replacement policies at destination proxies. If a cache gets full, no migration can be performed unless there is a policy to replace existing content. Third, the system is based on very simple mobility prediction of a single user. It is questionable that all the content subscribed by one user should be pre-fetched, as it may turn out insignificant when comparing to the overall gain of the remaining users at the location when cache space is used for content popular among them.

However, in similar problems, Multiple-Criteria Decision-Making (MCDM) [Xu, 2012] has been a successful approach to make decisions in the presence of multiple, usually conflicting, criteria. As it is not tied to a specific problem, it can be applied to a very diverse range of scenarios and problems, from simple decisions to complex problems. At the same time, it supports multiple weighted criteria and typically returns a finite number of solutions when dealing with a selection/assessment problem. Therefore, it fits the issue of selecting content for migration and selecting alternative destinations to put that content into.

2.6 Summary

In this chapter, related work pertaining to this thesis was described and discussed. From technologies to projects and concrete scientific proposals, they provide a deeper understanding of the motivation behind the work of the thesis. Namely, technology limitations, unexplored combinations of technologies and issues not addressed by existing proposals allow us to derive research challenges (c.f. Section 1.2) that lead to the contributions depicted in the following chapters.

To summarize, the first challenge is to leverage cloud computing and NFV to integrate content delivery within mobile networks and hence gather benefits not previously exploited by existing technologies and paradigms. The second challenge is the integration of content delivery (using ICN) within LTE mobile networks, exploring the benefits of C-RAN and MEC to extend content delivery mechanisms closer to the edge and hence closer to the user. The third challenge is towards the usage of higher level and content-aware strategies to perform offloading and load balancing between radio technologies, overcoming the limitations imposed by protocols, technologies and architectures. The fourth challenge is to explore hierarchical caching with new mechanisms and algorithms that enhance content delivery at the edge of mobile networks, thus increasing cache hit ratios (lower latencies for users, cost savings for operators). The fifth challenge is to explore the FMC concept towards the optimization of content placement at the edge of mobile networks, leading to improvements over more common edge caching strategies.

Below, in Chapter 3, we first describe the evaluation environments that were used in the validation of our contributions towards these challenges.

Chapter 3

Evaluation Platforms

In this chapter, the evaluation platforms used to validate the contributions of this thesis are presented. These platforms include a cloud computing testbed and a simulator, used for different purposes and in different scenarios. These ensure that the contributions are validated in terms of performance and feasibility, as well as that they can be compared to existing solutions and proposals to verify that improvements exist and that they bring additional value over them. Additionally, they allow for detection of problems that affect the expected outcome, allowing for extra iterations in the design and implementation to mitigate those issues and also bring extra enhancements in terms of performance.

The most important platform is the cloud computing testbed, as it was used in the context of the MCN project and provided an environment to run an end-to-end setup with all the different components, i.e. a very realistic scenario in many ways. Section 3.1 discusses this testbed in terms of hardware and software, providing an overview of all the effort that was taken to make it run and able to fit large scale experiments. However, as some evaluations require a scale that is not possible within the testbed or even feasible to implement solely with existing software components, simulation was also used. The simulation platform, together with its extensions, is described in Section 3.3.

3.1 Cloud Computing Testbed

The cloud computing testbed used for the evaluations in this thesis was established in the Institute of Computer Science of the University of Bern, using both existing hardware and newly purchased equipment. Since it was first established in 2014, multiple iterations took place to increase capacity, improve performance and introduce new features. It had many engineering challenges, its open source software platform is still not very mature. However, after handling each of the challenges that many times appeared during the experiments, it proven to be fast, reliable and of extreme importance towards the evaluations to be carried out. This applies both to the work developed in this thesis and to the entire work of the MCN consortium.

In the sections below, the testbed is described with more detail. First, in Section 3.1.1 the hardware and the low level architecture are depicted. Later on, in Section 3.1.2, the platform used to established the testbed and its software components are briefly highlighted.

3.1.1 Hardware

The cloud computing testbed includes 5 different equipments: 2 DELL servers (Fig. 3.1), 1 DELL Storage Area Network (SAN) and 2 DELL 10Gb Ethernet switches (Fig. 3.2).

Figure 3.1: Cloud Computing Testbed Servers

The specifications of the equipments are as follows:

- Custom Server: 4 physical Central Processing Unit (CPU) cores (single threaded) at 2.83 GHz, 8GB of RAM, two 1Gb Ethernet interfaces and one 250GB hard drive.
- DELL PowerEdge R320 Server: 10 physical CPU cores (2 threads each) at 2.40 GHz, 64GB of RAM, two 1Gb Ethernet interfaces, four 10Gb Ethernet interfaces, and two 800GB solid state drives in Redundant Array of Independent Disks (RAID) 1 for redundancy and extra read speed. This equipment includes redundant power supplies.
- DELL PowerEdge R530 Server: 24 physical CPU cores (2 threads each) at 2.50 GHz, 192GB of RAM, four 1Gb Ethernet interfaces, four 10Gb Ethernet interfaces, and two 1TB hard drives in RAID 1 for redundancy. This equipment includes redundant power supplies.
- DELL PowerVault MD3600i SAN: two independent Internet Small Computer Systems Interface (iSCSI) controllers, each with two 10Gb Ethernet interfaces

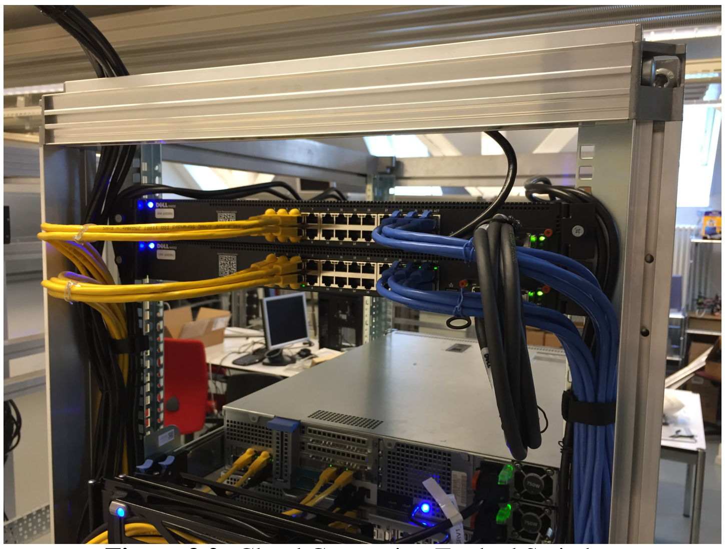
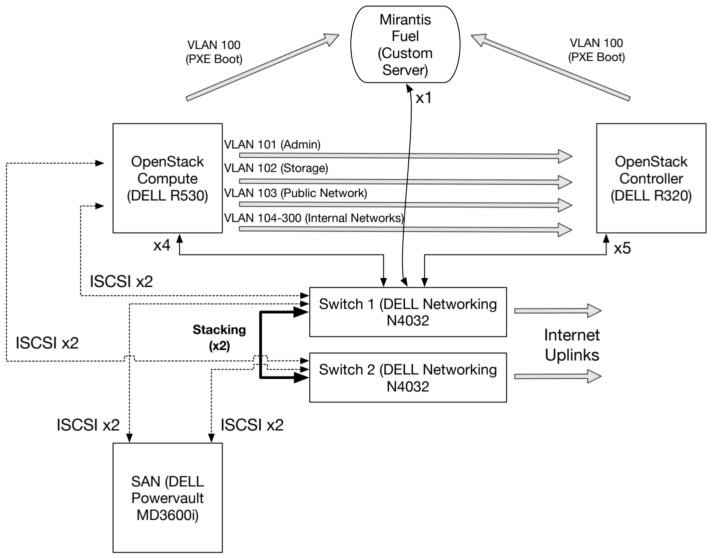


Figure 3.2: Cloud Computing Testbed Switches

and access to all the hard drives. The hard drives are six 600GB 15k Rotations Per Minute (RPM) Serial Attached SCSI (SAS) disks and six 3TB 7.2k RPM SAS disks. This equipment includes redundant power supplies.

- DELL Networking N4032 Switches: each switch has twenty four 10Gb Ethernet ports, redundant power supplies and two 40Gb Ethernet stacking interfaces.

This is the hardware in the latest stage, already after upgrades and iterations. Initially it was neither so powerful nor redundant. Following the typical design principles of other cloud infrastructures and also using personal background's knowledge, the low level architecture depicted in Fig. 3.3 was achieved.


Mirantis Fuel (Custom Server)
VLAN 100 (PXE Boot)
VLAN 100 (PXE Boot)
x1
OpenStack Compute (DELL R530)
VLAN 101 (Admin)
VLAN 102 (Storage)
VLAN 103 (Public Network)
VLAN 104-300 (Internal Networks)
OpenStack Controller (DELL R320)
x4
x5
ISCSI x2
Switch 1 (DELL Networking N4032
Stacking (x2)
ISCSI x2
Internet Uplinks
Switch 2 (DELL Networking N4032
ISCSI x2
ISCSI x2
SAN (DELL Powervault MD3600i)
Figure 3.3: Cloud Computing Testbed Low Level Architecture

From the figure we may see that the servers each play a different role: one is the controller node (handling orchestration, management tasks and interfacing), another is the compute node (handling all the processing tasks). As for storage, the SAN is attached to the compute node using 4 redundant links, providing both load balancing and ensuring that if an interface, switch, cable or any other component fails, there will be no data loss or corruption. Also important is the connection between the controller node and the compute node, which allows for all the orchestration and computing operations to occur and for networking to exist between the Internet (through an uplink) and the virtual instances running in the compute node.

3.1.2 Software

Together with the hardware and corresponding low lever architecture described in the previous section, a cloud platform was used. In this case, OpenStack [OpenStack Foundation, 2016] was selected due to its wide adoption, open source code and industry impact. OpenStack is a free and open source cloud computing platform, providing IaaS. It consists of components that control both the hardware compute pools, storage and networking resources. Users can access the platform using a web interface (Horizon) or one of its many Representational State Transfer (REST)ful APIs. It began in 2010 as a joint effort between Rackspace and National Aeronautics and Space Administration (NASA), and currently it is managed by its own foundation (OpenStack Foundation) with the support of more than 500 companies.

Its main software components are the following:

- **Keystone:** This is the central authentication component in OpenStack, responsible for registration of all other services and components. All the operations in OpenStack start from this component and its API.
- **Glance:** This OpenStack storage system is a component responsible for the management and storage of cloud images and snapshots.
- **Nova:** The OpenStack computing component handles all the computing operations in direct connection with the virtualization infrastructure.
- **Neutron:** The OpenStack networking component that handles networking in OpenStack by managing physical interfaces, virtual interfaces, virtual switches and virtual routers.
- **Heat:** This component handles orchestration from templates in OpenStack. It enables automatic and seamless deployment of instances and their required components, together with the establishment of relations between all of those.
- **Horizon:** As interfacing is not always done with APIs, this is the OpenStack dashboard. It consists of a web interface to manage the entire system from the end user perspective.

To simplify the deployment of all the components in a Linux operating system, multiple tools have been developed in the community. However, there is not a *de facto* tool that can satisfy all the requirements while deploying a custom environment. The most complete tool for OpenStack assisted deployment is Mirantis Fuel [Mirantis, 2016], and it served as a base for the deployment of this testbed. It guides the user through a series of web-based configuration pages, and with a single click it deploys the environment in the available nodes. However, sometimes it presents deployment problems and bugs that seriously hinder the efforts of deploying the OpenStack platform. Also, even though it has evolved much since 2014, it does not handle custom and advanced configurations such as ISCSI configuration for access to the SAN (with Multipath for redundancy), Lightweight Directory Access Protocol (LDAP) authentication to have centralized accounting and plugins installation/configuration for Virtual Private Networks, Load Balancing and other relevant features. These have to be performed manually, sometimes with new challenges due to lack of documentation, different operating systems, etc.

3.2 Emulation Platform

OAI [Nikaein et al., 2014] is an open-source experimentation and prototyping platform created by the Mobile Communications Department at EURECOM. It aims at giving researchers a platform to develop new ideas in the area of mobile communications. It is an emulator of LTE, and its functionalities (base stations, user equipments, core network) are achieved using a software radio front end connected to a computer for processing. Moreover, it can completely emulate the components of LTE without the radio front end. This means that it can be used for more complex emulations if the necessary computing resources are available, such as the one in Chapter 5.

3.2.1 Emulation Details

Together with its real-time operation using a software radio front end, the full protocol stack of LTE can be run in a controlled environment for the realistic validation and performance evaluation of innovations at different levels. It is still representing the behavior of the technology, as it respects the processing and timing requirements. It contains two different emulation modes: PHY Abstraction mode that relies on a physical layer abstraction unit that generates error events in the channel decoder and Full PHY, where computation becomes intensive and a real physical signal comes together with an emulated channel, respecting all the real-time requirements. As each node also contains its own protocol stack and support for IP, it can be connected to external traffic sources such as the content delivery system proposed in this thesis.

In addition to these functionalities, OAI also contains the 3GPP channel models with path loss, shadow fading and stochastic small scale fading. These interact with the mobility generator to calculate different channel models and interference patterns, depending on the user's location in regard to the eNB. This creates a more realistic environment, and any emulations will have users in different settings and using different amounts of processing resources.

3.3 Simulation Platform

When it comes to network simulation, few platforms are widely used by researchers. One of them is Network Simulator (NS) [ns-3 Project, 2016b], a discrete event simulator with three versions so far (ns-1, ns-2 and ns-3). They are mostly used for research and teaching, and are free software with open source code available online. As the work of this thesis requires advanced functionalities and frameworks that ns-3 offers towards the simulation of different networks, it was selected as the base simulation platform.

3.3.1 Simulation Workflow

The process to run a simulation in ns-3 is divided into the following steps:

- **Topology definition:** Easily create and define basic entities and their network relationships.
- **Model development:** Models are added to simulation in a simplified way, e.g. User Datagram Protocol (UDP), IP Version 4 (IPv4), point-to-point devices and links, applications.
- **Node and link configuration:** This is the step where models set their default parameter values (e.g. packet size, Maximum Transfer Unit (MTU), etc.
- **Execution:** In this step the simulation is run. The system generates events and gathers data requested by the user.
- **Performance analysis:** After the simulation is finished, data requested by the user is available in a trace with timestamps. This data is ready to be processed with external tools.
- **Graphical Visualization:** In this optional step, both trace data and processed data obtained from the simulation output can be displayed in the form of graphs using graph tools such as GNUPlot.

3.3.2 Extra Frameworks

The ns-3 simulator has a wide variety of modules and frameworks that can be added to simulate different networks and components, and in the evaluations present in this thesis the modules described in the following paragraphs played a major role.

Direct Code Execution (DCE) [ns-3 Project, 2016a], used in Chapter 6 for the evaluation, is a framework developed for ns-3 that includes all the necessary components to run existing implementations of network protocols and/or applications within ns-3. This is quite important when the software components to be tested do not exist or are not fully implemented for the simulation, allowing real code to be run inside simulation without any changes. It is also a way to abstract processing times and consider only the network, as simulation is stopped before external code is run. A diagram depicting the inner architecture of DCE is depicted in Fig. 3.4. In the case of this thesis, DCE was used to perform evaluations where simulation is required but real-world code (not ready for simulation) was available.

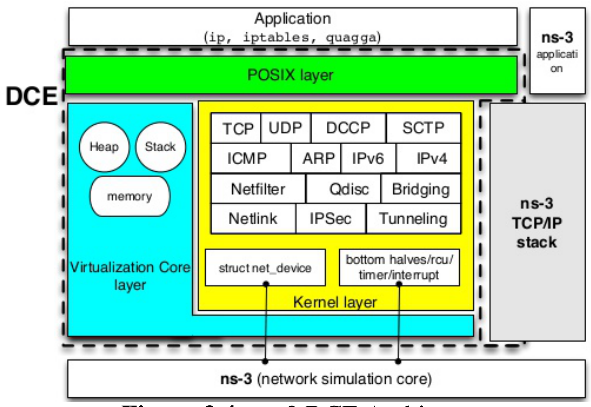


Figure 3.4: ns-3 DCE Architecture

LENA [Mezzavilla et al., 2012], used to establish the LTE network for simulations in Chapter 8, is an open source framework for ns-3 that enables the simulation of a LTE mobile network with all the required components (see Fig. 3.5). Initially developed to design and evaluate the performance of radio resource management algorithms, radio schedulers, load balancing, mobility management, etc., it can also be used to support other simulations that require a large-scale realistic LTE mobile network. In the evaluations performed in this thesis, it played a major role to simulate large-scale networks where LTE mobile networks are required but the scale prevent the experiments from being run in testbed environments.

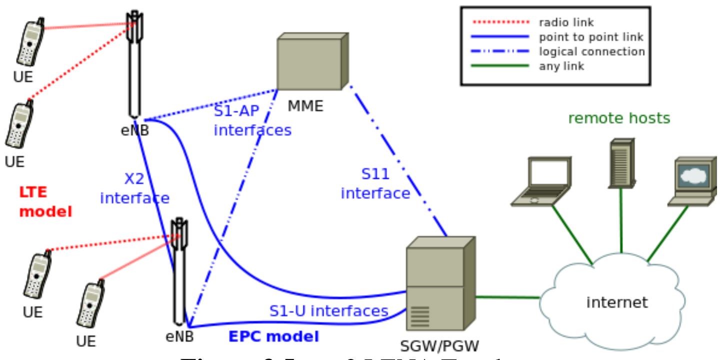


Figure 3.5: ns-3 LENA Topologt

Finally, ndnSIM [Afanasyev et al., 2012], used to establish the NDN layer for simulations in Chapter 8, is a framework based on the general philosophy of typical ns-3 network simulations. It provides all the components to run NDN networks within ns-3 through an abstraction layer that eases the work for its users. As it is an independent protocol stack, it can be installed in any simulated ns-3 node. It also includes basic traffic generation applications and repositories, together with tools to get NDN-specific statistics. However, for the experiments performed in this thesis, it had to be slightly modified in order to fit the requirements. More details about these modifications and why they were required are given in the corresponding evaluation description.

3.4 Summary

In this chapter, the different platforms for the evaluation of the contributions of this thesis were described. Each has its own purpose, depending on the goals set for the validations to be obtained. While the testbed platform described in Section 3.1 is ideal for the evaluation of real implementation frameworks and service such as the one in the following chapter (Chapter 4), it becomes limited for other evaluations where simplification is required to assess individual performance of components and the feasibility of their operation. This is the case in Chapter 5, where the feasibility tests require that the scenario is limited but at the same time includes the constraints that would affect it in a real-time and real-world setting. Hence, in that particular chapter, it makes sense to use the emulation platform described in Section 3.2.

At the same time, experiments such as the ones necessary for the evaluation of the contributions depicted in Chapters 6, 7 and 8 require only that small proof of concepts are developed. These can either be evaluated in small testbeds/clusters to test individual components (c.f. Chapter 7) or put into a simulated environment (c.f. Chapters 6 and 8) where they can be validated for their performance in large scale scenarios using the simulator described in Section 3.3, as these are the scenarios where they will make a difference for end users and where their scalability can become a real challenge. Moreover, and because for Chapter 8 the required platform was partially available with a real implementation, this chapter also includes a validation in the testbed environment depicted in Section 3.1 in order to validate its performance in a end-to-end realistic scenario.

Chapter 4

Content Delivery as a Network Service

Content delivery is often a service provided on top of networking infrastructures, meaning it needs to fit into them and be easily put into place. The best way to handle services today is cloud computing, as following its principles guarantees that a number of requirements are met from the start (e.g. on-demand instantiation and rapid elasticity). As FI concepts such as ICN grow in importance to handle content delivery, there is a rationale that drives a choice to virtualize their network components and create a service that can easily be deployed together with the existing underlying mobile network infrastructures. Therefore, a new service (ICNaaS) can be used to improve the performance, efficiency and ease of retrieving content in mobile networks by providing it with extra functionalities. For this chapter, the goal was to design the architecture of this service, implement the necessary mechanisms and evaluate it in the realistic scenarios of the MCN project. In Section 4.1 a brief motivation for this work is presented. Section 4.2 describes its internal architecture and mechanisms. Section 4.3 presents the evaluation of this service. Finally, Section 4.4 concludes the chapter and sums up the achievements of this work.

4.1 Motivation

Nowadays there are numerous cloud platforms, frameworks and environments. These support a wide range of services from scratch, but also support the easy development of new applications that must run in the cloud and follow its principles. However, it is still possible to achieve higher layers of abstraction and simplify the most common part in the development of applications. That was one of the goals in the MCN project, which led to the development of a whole framework for the development of improved versions of existing services, but also for the development of new content-focused services like ICNaaS. Having such a NFV-like framework simplifies the task of creating ICNaaS at a generic stage, but there are still specific requirements that demand for new mechanisms. Namely, the mitigation of the performance impact when virtualizing certain network components (routers and their internal functionalities), compatibility with legacy protocols based on the host-centric paradigm to support existing systems and devices, self-configuration of ICN related parameters on instantiation and automatic scalability based on ICN-specific statistics. These were challenges that needed to be addressed, and they are covered in the next section.

4.2 Architecture and Mechanisms

ICNaaS [Aad et al., 2013; Ruiz et al., 2014; Corici et al., 2015; Carella et al., 2015] can be used to improve the performance, efficiency and ease of retrieving content in mobile networks. In this section, the design aspects of this service and its mechanisms are described, showing its overall performance in a virtualized environment, on-demand deployment and scaling capabilities [Gomes et al., 2016c; Sousa et al., 2016]. With the approach of providing ICN components in the cloud, benefits such as the close integration with other cloud services could be obtained in the scope of platforms such as the one from MCN (c.f. Section 2.3.2). This makes it easy for an EEU to request and deploy those services according to its needs. Also, leveraging the advantages of cloud principles and pushing the boundaries of existing content delivery technologies enables the addition of new concepts and mechanisms to bring significant improvements. For this work, the select ICN approach was CCN (see Section 2.2.7) due to its relevance in the research community and its open source code.

4.2.1 Research Issues

The research issues to be tackled in this chapter can be split into three different problems: cloudification of ICN (offering it as a service), improved compatibility with legacy systems and usage of an integrated platform to support added functionalities such as automated scalability and enhanced content delivery mechanisms.

The first issue concerns the design and development components to handle the entire process of getting the ICN service ready to use for the EEUs without a need for installation or configuration of components. The EEUs only needs to provide the network topology for the deployment, and they will get a running service that manages itself in terms typical operation and scaling. As a research question, the whole scaling and dimensioning process needs to be addressed. It is the goal of the design and its mechanisms to adjust the scale of the service according to the load triggered by content requests of the users while efficiently using resources, ensuring the best performance for end users in terms of content delivery. Moreover, mechanisms need to be put into a component that handles the whole ICN topology, both in terms of forwarding/routing and regular caching operations.

The second issue, addressing the integration with legacy systems, is rather important for the end users, as most of them are not aware of ICN protocols and how to communicate using them. Therefore, a component with new mechanisms that enable a seamless connection with other legacy protocols and, at the same time, let users use ICN (taking advantage of its benefits) without being aware of its existence. The entire process can then be considered transparent and operates in the same way as transparent HTTP proxies would in currently deployed infrastructures, but for ICN and handling all the issues of converting requests from one paradigm to the other in an efficient manner. These components, their mechanisms and the overall architecture serve as the basis for the work presented in Chapter 5.

The third issue concerns how the service integrates into existing platforms. In the case of ICNaaS, the service is integrated into the MCN platform and can easily be integrated with other services. In fact, it takes advantage of existing services (e.g. IaaS, MaaS, MOBaaS, etc.) to perform its operation and also provides interfaces for other services to take advantage of the ICN protocols and the developed functionalities. This integration serves the purposes of service scaling according to the monitored load, but also for enhancements in content delivery such as the ones described later in this thesis in Chapter 8 where mobility prediction information is used to make decisions regarding content distribution. This requires specific components that handle all the gathered information and use mechanisms in order to process them and make optimized decisions.

4.2.2 High-Level Design

The high-level architectural design diagram of ICNaaS, based on the overall architecture of MCN described in Section 2.3.2, can be found in Fig. 4.1. Developing it was an iterative process done for the MCN project but also for this thesis, and lead to multiple changes along the way. However, only the final version is depicted here for the sake of simplicity. This architecture is specific to the protocols of ICN and the deployment of its components, and also enables the existence of multiple enhancements in content delivery present in the different contributions of this thesis.

Besides the MCN generic components, the core of an ICNaaS instance consists of five main components:

- **SO** is the component responsible for the orchestration of the service. Besides the generic functionalities described as part of the MCN architecture, it includes all the mechanisms for the automatic deployment/disposal of an ICNaaS (SO Execution), together with the mechanisms for the scaling operations and related decisions (SO Decision).
- **The ICN Manager** is the component responsible for the coordination of the ICN layer, controlling and configuring all the CCN routers with a global vision of the network topology. Whenever a new CCN router joins the network, it contacts the ICN Manager to register itself and obtains its initial configuration. This component is also generic for any ICN approach besides CCN, hence the name.

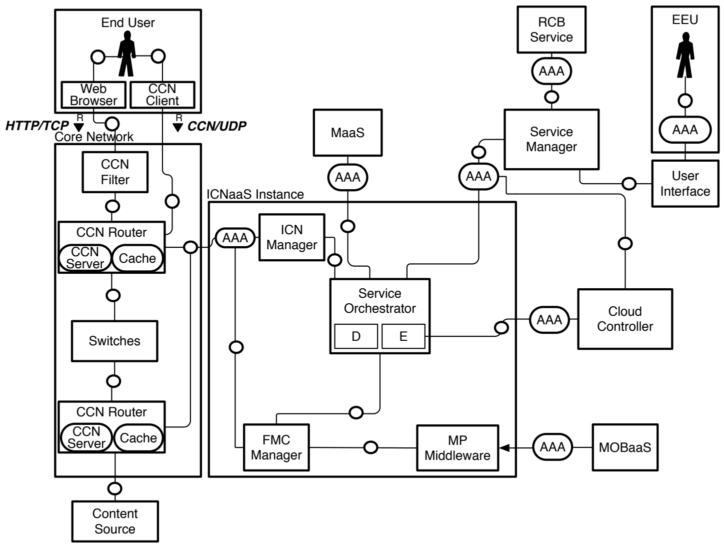


Figure 4.1: ICNaaS Architecture

- **Management Agent** is the component responsible for providing a web interface to the ICN Manager, which can be used by the EEU to perform an advanced setup of the service. For example, it may modify the topology, manage Interest forwarding routes (i.e. FIB entries) or obtain new endpoints to add new content to the network.
- **The CCN Filter** is a proxy and packet classification filter. It supports the following features/functionalities:
  1. It has a proxy used to translate HTTP GET request to CCN Interest and CCN data packet (content) to HTTP traffic.
  2. It contains a packet classification filter that is used to filter CCN related traffic from HTTP based traffic. Essentially, if a packet comes in the list of known Transmission Control Protocol (TCP)/UDP ports, initial header decoding is performed to assess if traffic is CCN-compatible. If not, HTTP header decoding is performed and if traffic is HTTP it may be proxied using translation to CCN messages.
  3. Multi-user support is included, meaning the CCN filter is able to handle requests from different clients simultaneously.
  4. In a later version HTTPS support was added, enabling users to use encrypted web traffic.
  5. As large file transfer support was required, fragmentation was also implemented.

  This component is not described further this thesis, as it was developed by other partners in the context of the MCN project [Corici et al., 2015]. Other components, however, were all developed by us.

- **The CCN Router** component includes several daemons to manage requests and, the repository, running on a machine. The main goal of this component is to handle content requests (Interest messages), which correspond to routing these requests, or resolving requests if the content is available at the local cache. The content is provided from the cache of the CCN daemon or from a repository (CCN repository). When a content request is resolved, either locally or remotely using forwarding, the router logs locally that this request was processed at that moment. The other daemon (CCN Server) then provides extra processing and forwards the information, as described below.

Additionally, to support the functionalities described in Chapter 8, three other components exist:

- **MP Middleware** is a simple component to interface with MOBaaS and handle mobility information retrieval periodically. It includes a timer to request for predictions of the users's mobility, and each request is done to the API of MOBaaS using a web service specification. All request include a destination API call, meaning that this component informs MOBaaS about the FMC Manager's address to deliver mobility information. This allows for load balancing on a possible future scalability of the FMC Manager.
- **CCN Server** is a daemon part of the CCN Router. It consists of multiple threads that perform different actions and as stated above, it has two responsibilities, namely sending the log information to the database and receiving as well as performing instructions coming from the FMC Manager. For the first part, it has two threads, one that is continuously reading from the pipe, and another that periodically checks if it has data to upload, and if it has, sends it going back to sleep again after. For the second, it has a thread listening for incoming connections and, for every new incoming connection, instantiates a new thread that will perform all the necessary steps and end after everything is done.
- **FMC Manager** is a centralized processor of information and will be the brain of the content migration mechanisms described in Chapter 8. It performs two main actions: receive the log data sent by every router and store them into the database; receive messages from MOBaaS about users that are going to move, specifying some details about source, destination and number of users. The decision mechanism of the FMC Manager will then take into account this information, the data stored in the database and other data that can come from other components and decide if it should, where and what it should migrate.

Initially, when an EEU wants an instance of the service, it contacts the SM. This component has a catalog of ICN service combinations that can be offered, and upon request it will deploy a SO by contacting the CC (see Section 2.3.2), which is the component that manages the cloud platform. Once the SO has been deployed, it will use its Execution (E in Fig. 4.1) sub-component to deploy all the remaining components of the service instance in the following order:

1. The ICN Manager component to handle automated management of the entire ICN layer.
2. The CCN Filter, which converts HTTP Requests into CCN Interest messages and HTTP Responses into CCN Data messages.
3. The CCN Routers that implement a subset of ICN functionalities, in particular CCN.
4. The FMC components (introduced and described further in Chapter 8).
5. The Management Agent, which provides an interface for the EEU to have fine-tuned manual control over the service instance.

At the same time, the SM handles dependencies of the ICN service's components, i.e. required external services to perform some of its functionalities. MaaS is used to monitor components and provide load metrics to the Decision (D in Fig. 4.1) sub-component for scaling in and scaling out decisions that allow the service to have a close to optimal number of resources to efficiently handle the load at a given time. The second component with a dependency is the FMC Manager, which uses MOBaaS to get input related to user mobility, either predicted or detected. To handle authentication between components, AAAaaS is also required. Finally, RCBaaS is used by the SM to charge and bill the EEU for the resources used by its service instances. This dependency is required to meet the requirement of pay-as-you-go defined in the cloud concepts.

4.2.3 Low-Level Design and Implementation

In this section, the low-level design of the ICNaaS components together with its implementation details are presented. It includes both class and sequence diagrams, together with descriptions on the internal mechanisms of each component, thus providing a detailed explanation on how ICNaaS and its components work. These were developed according to principles from the MCN project but, apart from the FMC components (developed together with a Master's Student, Vitor Fonseca [Fonseca, 2015]), they are solely developed as part of this thesis work.

Service Orchestrator

The SO implementation, depicted in the figure below, consists of two main packages, which are tightly related and hence need to be used together: the web service package (application) and the functional package (SO itself). The first has a single class named Application, which handles the HTTP requests made to the available API. The second has one main class (ServiceOrchestrator), which groups the two other classes (ServiceOrchestratorDecision and ServiceOrchestratorExecution). As the names state, these are the classes with the parameters and methods responsible for the decision and execution modules of the SO, respectively. The

SO is instantiated by the SM, and it is also the SM that interfaces with the SO API to trigger its actions.

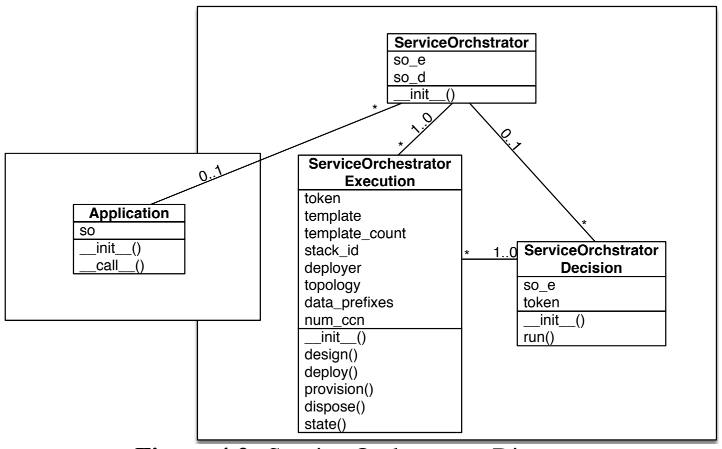


Figure 4.2: Service Orchestrator Diagram

The processes directly handled by the SO, i.e. its low level design, are the following:

- **Service Instance "Initial Design, Deployment and Provisioning":** The process of getting a service instance to run has three stages: design, deployment and provisioning. The first step for the EEU is to provide the design for the service instance, which is then processed by the web service (from files to memory structures) and provided to the SO Execution. The EEU then gets a confirmation about the outcome of this operation. Then, the EEU will ask for the deployment of the service instance, which is requested ultimately to the CC in the form of a heat template. As this is a non-blocking operation, the EEU will need to check the status of the deployment and only after it has been completed it may ask for the provisioning, which marks the end of the process and means that the service instance is running and configured.
- **Service Instance "Run-time Management":** During run-time of the service instance, two scaling operations can happen: scaling out and scaling in. In the first one, depicted in the figure below, the SO Decision module decides that the service instance must be scaled out to deal with a load increase. It then provides a new design to the SO Execution module and asks for its deployment, which will trigger the deployment of a heat template in the CC. The SO Decision then has to wait for the deployment of the new stack, periodically checking the state of its deployment process. As soon as it is completed, it may ask for the provisioning, which will configure the new CCN Routers and start their daemons.

  The scaling in operation is very much alike, although the main difference is that after the new design is provided to the SO Execution, the SO Decision will only ask for the disposal of the stack corresponding to the CCN routers which

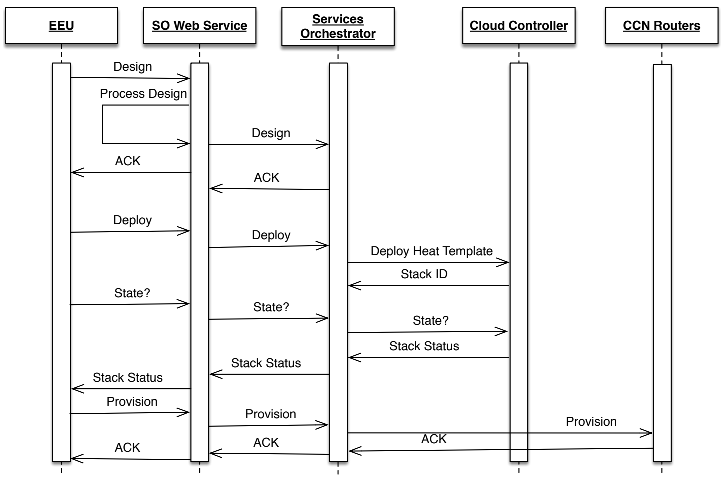


Figure 4.3: Service Instance Initial Design, Deployment and Provisioning

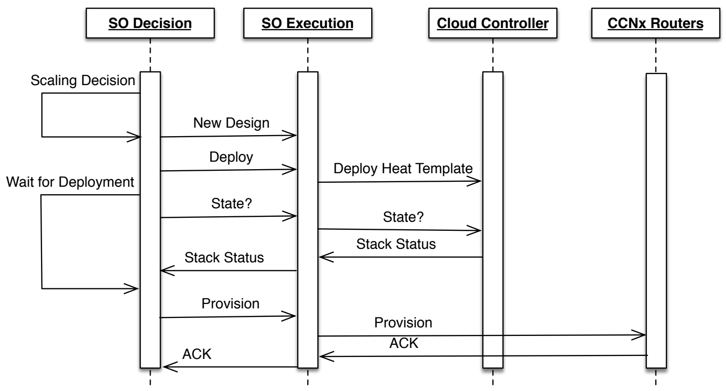


Figure 4.4: Service Instance Scaling Out

are not being needed anymore.

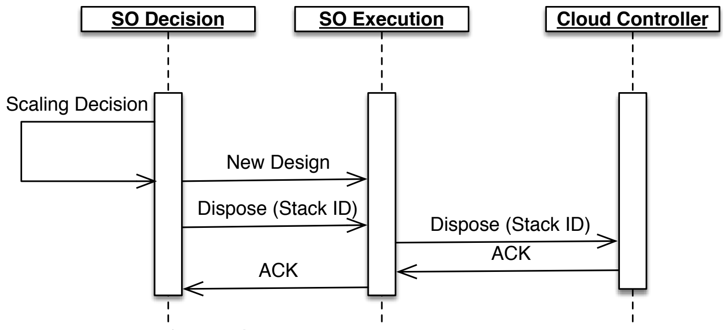


Figure 4.5: Service Instance Scaling In

- **Service Instance "Disposal":** Whenever the EEU does not need the service instance anymore, it may ask for its disposal. Using the SO Web Service, it requests the disposal of the service instance and the SO will ask the CC to dispose of all the stacks associated with this service instance. Finally, the EEU will monitor the process asking for the state of the service instance, and if the disposal request succeeds, it will get a confirmation that no stacks are deployed at that given moment.

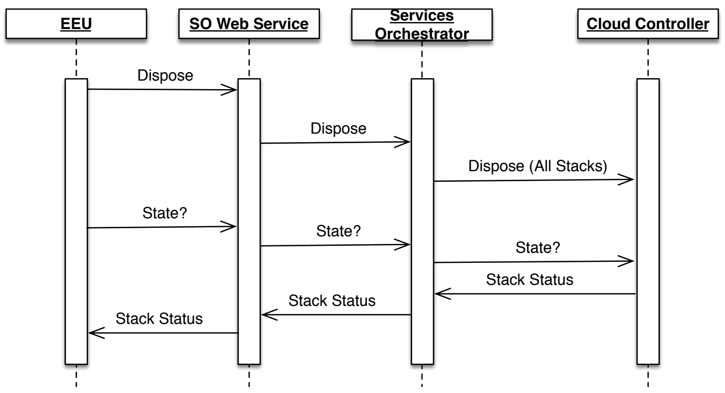


Figure 4.6: Service Instance Disposal

ICN Manager

The ICN Manager is a component with a simple architecture, which handles the registration and configuration of CCNx routers using the Secure Shell (SSH) protocol as underlying technology. Configuration may happen at two different stages: at registration of the CCN Router or at a later time (reconfiguration). The two different possibilities are illustrated below.

CCN Server

The CCN Server's implementation is composed of three classes, as depicted in Figure 4.9: CCNServer, Monitor and ServerProcesser. There are other classes that represent the messages exchanged and which are required by Jackson [FasterXML, 2015], a Java library for processing JSON, and a simple thread that reads the log entries from the named pipe shared with CCN components. When CCN Server is initialized, it instantiates a Monitor thread that periodically tries to connect to the FMC Manager and upload the available data. After the Monitor thread has been started, it starts another thread for reading the named pipe. The CCN Server thread initializes a ServerSocket and keeps waiting for incoming connections. When a new connection is established, a new ServerProcesser thread is instantiated and the CCN Server thread keeps listening for incoming connections.

FMC Manager

The class structure with the implementation of the FMC Manager is presented in Figure 4.10. The Monitor class is a thread instantiated at the start and that is

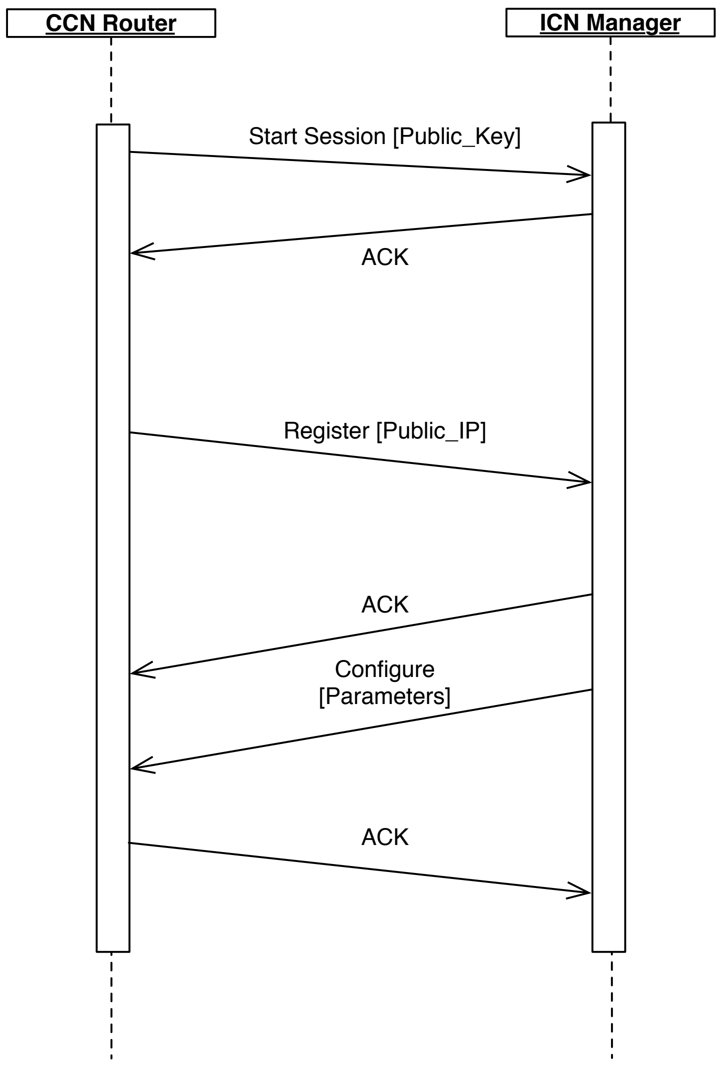


Figure 4.7: ICN Manager Registration/Initial Configuration

responsible to receive the connections from CCN Servers and to store the received data into the database. The FMCManager class is the one responsible to send the messages to the CCN Servers in order to perform the necessary content migrations.

Since the CCNServers and the FMC Manager were designed to interact directly with each other, the sequence of the communications (low-level design) is shown in Figure 4.11. Although displayed in the same diagram, there are two independent sequences that can occur at the same time:

1. The first one concerns the storage of the logged data and is represented by the steps 1 and 2;
2. The other interaction between the two is to perform the relocation of contents and is represented by steps 3 and 4.

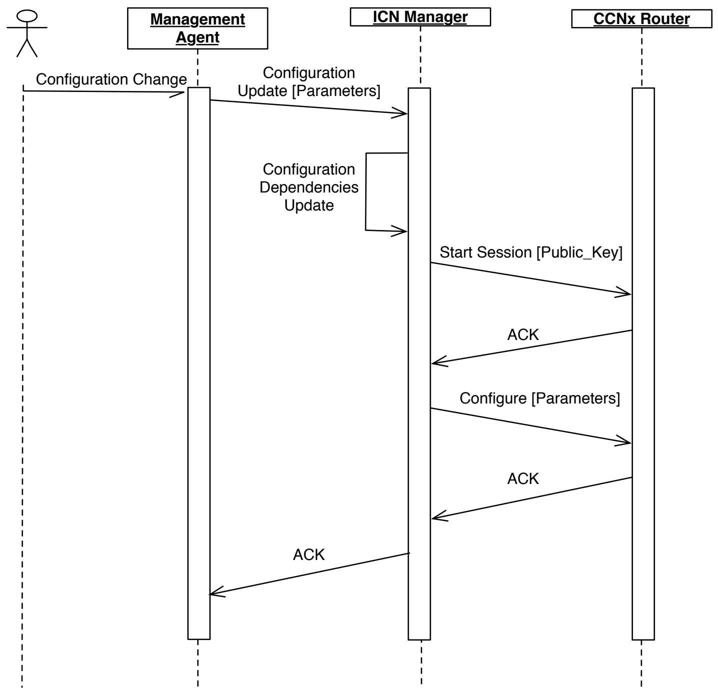

Management Agent
ICN Manager
CCNx Router
Configuration Change
Configuration Update [Parameters]
Configuration Dependencies Update
Start Session [Public_Key]
ACK
Configure [Parameters]
ACK
ACK

Figure 4.8: ICN Manager Configuration Update

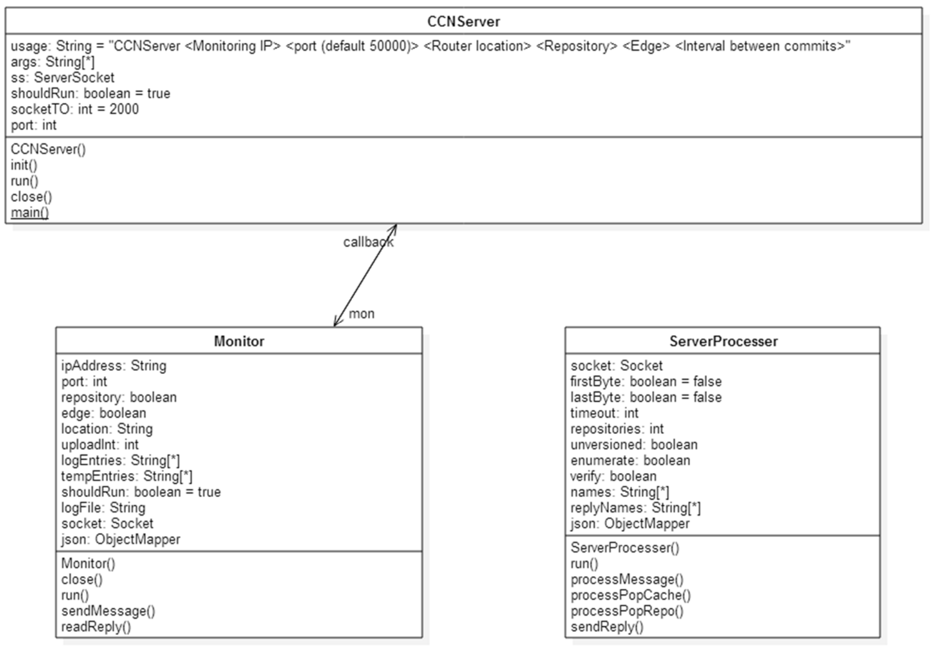


Figure 4.9: CCN Server Diagram

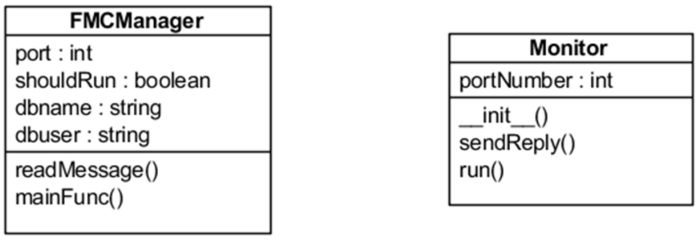


Figure 4.10: FMC Manager Diagram

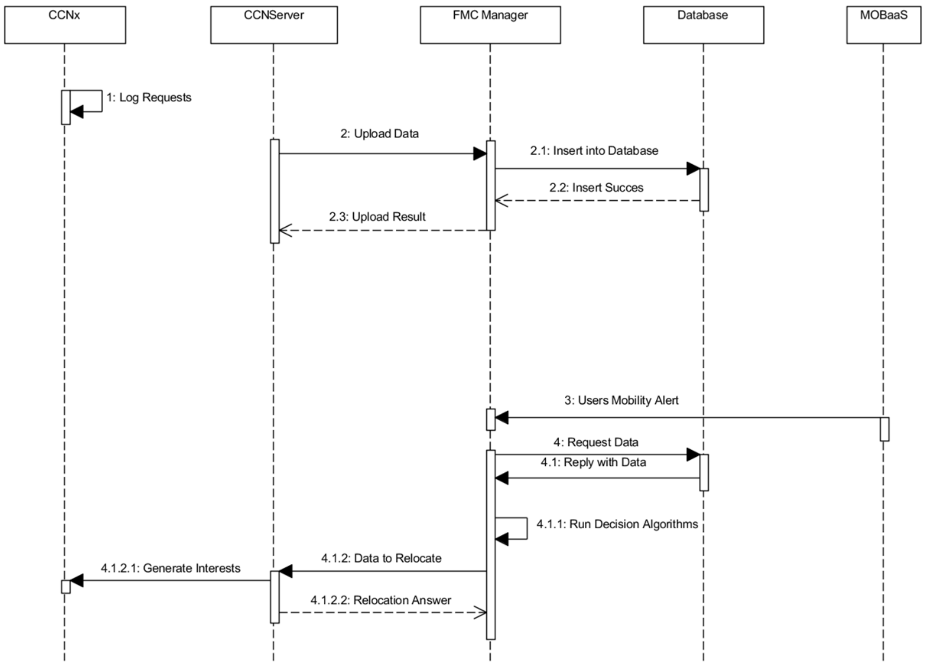


Figure 4.11: Communication between CCN Servers and the FMC Manager

In the following section, this service is evaluated to assess its functionality and performance.

4.3 Evaluation

Evaluating the proposed ICNaaS contributions is quite important, as it is the only way to assess if the functionalities work and cloud principles are actually followed. Moreover, evaluating the overall performance and its impact in content delivery is of utter importance to the work developed in the next chapters. Therefore, a set of evaluation scenarios was prepared and it is described in the next section.

4.3.1 Evaluation Scenarios

To evaluate the service, different evaluation scenarios were created and will be described in the subsections below. The first addresses the impact of cloudifying ICN in its content delivery performance, while the second demonstrates the functionality and benefits of scaling operations.

Cloudified ICN vs Non-Cloudified ICN

To evaluate the performance impact of cloudifying ICN, a set of tests were planned and executed running a default ICNaaS topology (5 CCN Routers in a 1-2-2 tree topology, see Fig. 4.12, i.e. 1 root node, then 2 nodes connecting to the root node and afterwards 2 leaf nodes connecting to both the previous 2 nodes, 1 ICN Manager).

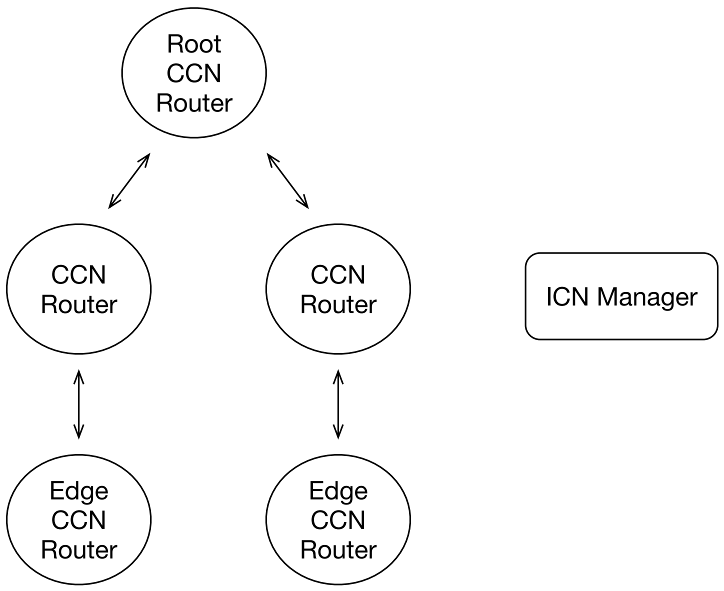


Figure 4.12: ICNaaS 1-2-2 Tree Topology

First, it was important to understand how the CPU load (measured within Linux) would change in comparable hardware, either used directly (bare metal), as private cloud (Cloud Computing testbed described in Section 3.1) or as public cloud (CloudSigma provider). The table below shows the specifications of the different platforms per service component:

	Physical CPU	**Actually Used CPU**	**RAM**	**Storage**
Physical Machine	Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz	2 cores limited to 2.50GHz	4GB	40GB 7.2k RPM desktop hard drive No RAID
Private Cloud	Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz	2 virtual cores @ 2.50GHz	4GB	40GB 15k RPM SAS hard drives RAID 5
Public Cloud	AMD(R) Opteron(TM) 6380 @ 2.50GHz	2 virtual cores @ 2.50GHz	4GB	40GB SSD RAID Unknown

Table 4.1: ICNaaS Testbeds' Characteristics

Therefore, CPU load was measured while varying the load of ICN in terms of Interest messages per second.

Second, the impact for end users had to be evaluated. That meant that content transfer performance had to be compared also for the three different platforms mentioned before (physical machine, private cloud and public cloud). To have different sets of content and minimize the impact of overheads (different with small or big content object sizes), three content mixes of approximately 1GB each were defined:

1. Contains 90% of small content objects and 10% of big content objects (small content objects: between 1 and 5MB; big content objects: between 50 and 100MB).

2. Contains 10% of small content objects and 90% of big content objects (small content objects: between 1 and 5MB; big content objects: between 50 and 100MB).

3. Contains 50% of small content objects and 50% of big content objects (small content objects: between 1 and 5MB; big content objects: between 50 and 100MB).

For each content mix, all the content was transferred and the total time of transfer was summed up for the three platforms considered.

Scaling Operations

In order to evaluate how well scaling is done by ICNaaS and the impact of scaling operations for end users, a set of two different tests was defined. These tests were run in our private Cloud Computing testbed, with each Service Instance Component assuming the characteristics described in Table 4.1, which are the same of the "m1.medium" flavour of OpenStack (2 vCPUs, 4GB of RAM, 40GB of disk). Also, a simple ICNaaS topology was considered (2 CCN Routers in a 1-1 tree topology, see Fig. 4.13, i.e. one root node connected to a single leaf node, 1 ICN Manager), scaling out by adding CCN Routers at the leaf level of the tree.

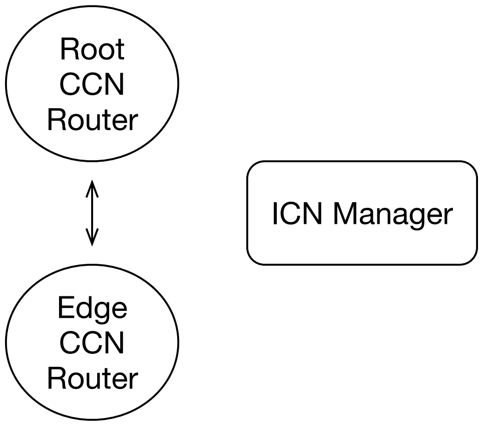


Figure 4.13: ICNaaS 1-1 Tree Topology

A first test aimed at comparing the total time needed for scaling out one machine, from when the metric is read from MaaS until the new component is fully running (scaling out) or disposed (scaling in). As in some cases scaling operations may create more than one instance of the scaled component, the evaluation also considered the scaling out time for different numbers of new component's instances.

As for the second test, the goal was to evaluate how the performance experienced by end users is affected when the service is overloaded and later is scaled to deal with such extra load. Therefore, the content transfer time for a 10MB object was measured every 10 seconds for 900 seconds (15 minutes). In the beginning (moment 0), the service was overloaded and after 300 seconds the scaling decision is made.

4.3.2 Evaluation Results

In the following sections, results for the previously described evaluation scenarios are presented.

Cloudified ICN vs Non-Cloudified ICN

In Figure 4.14, CPU Load is compared for different numbers of CCN Interests/second and for each of the 3 platforms. Clearly, the physical machine delivers the best performance and the CPU is less loaded, followed by the private and then the public clouds. However, the differences are small and sometimes within the error margins, proving that the service can be cloudified without too much impact on performance. In the same figure, one can also observe that even considering a single instance without scaling, the number of interests handled by the daemons for a given time unit is quite big without even reaching 50% of CPU load.

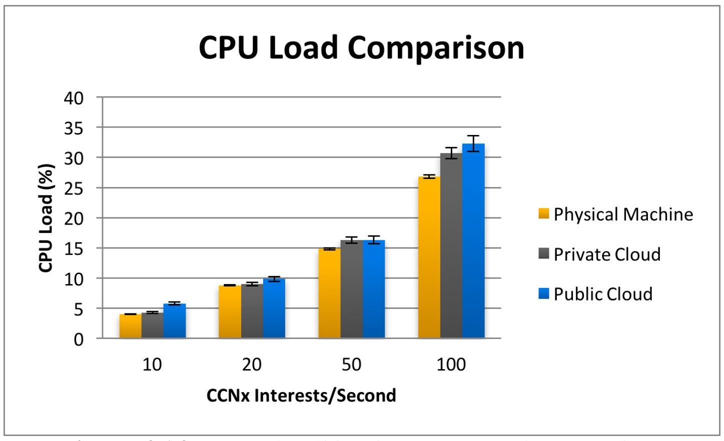


Figure 4.14: ICN Cloudification CPU Load Comparison

In Figure 4.15, again, we see the impact of the different platforms in terms of content transfer performance experienced by the end user. The difference is higher when the content mix consists of a majority of smaller objects, which by the quantity create more overhead and the usage of more processing resources. However, differences are not much higher for the cloud platforms when comparing to the physical machine, and even for Mix 1 they do not go much above a 2 minutes difference for 1GB of small content objects.

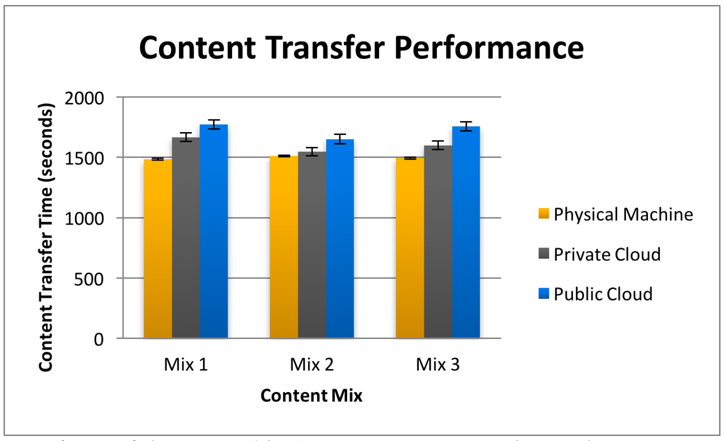


Figure 4.15: Cloudified ICN Content Transfer Performance

Scaling Operations

Below, in Figure 4.16, the total scaling time is compared for both scaling out and scaling in operations, considering a number of new instances of 1, 2, 5 and 10. Scaling in is a generally smooth operation, with the deletion of resources happening fairly quickly (between 10 and 15 seconds), regardless of the number of new instances. As for scaling out, there is a clear increase on the time taken because new resources have to be allocated, instances booted, service re-configured, etc. However, the increase is not linear and it is smooth for the different numbers of new instances being deployed simultaneously, not increasing to values that would affect much the behavior of the service before the scaling operation is finished.

In Figure 4.17, the advantage of scaling the service is clearly seen from the point of view of the end user. While during the first 6 minutes the service was having a highly unstable performance, ranging from good to fairly bad very quickly, after the scaling operation is triggered (upon 5 minutes of high load) and completed (about 85 seconds later) the performance becomes stable and at good levels in terms of content transfer times.

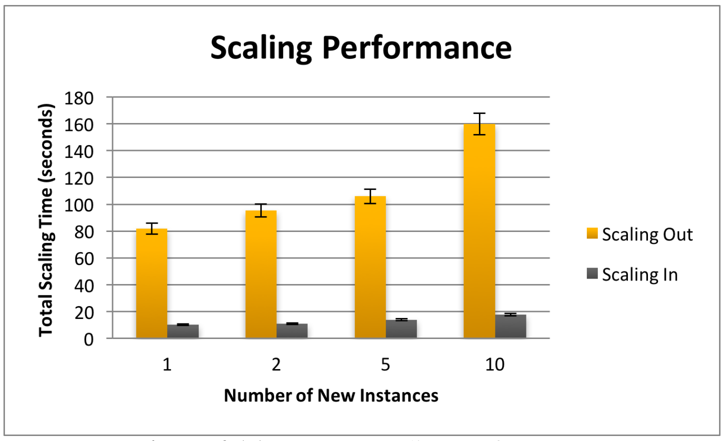


Figure 4.16: ICNaaS Scaling Performance

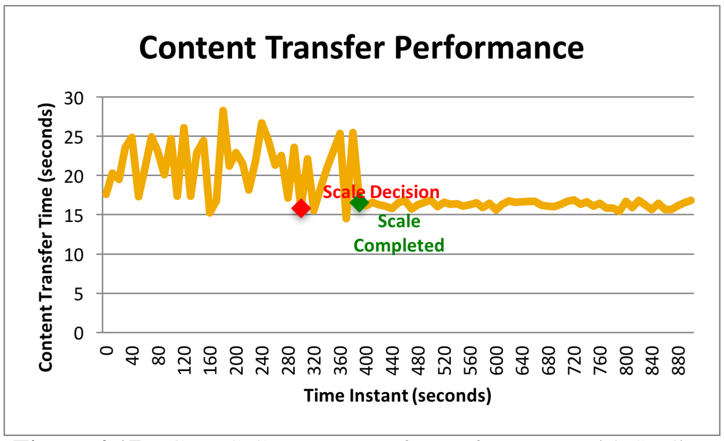


Figure 4.17: ICNaaS Content Transfer Performance with Scaling

4.4 Summary

In this chapter ICNaaS was introduced and described. It brings the possibility to improve the performance, efficiency and ease of retrieving content in mobile networks by creating a support platform for enhanced mechanisms using newly developed FI concepts. By putting together a set of virtualized network components, ICNaaS is provided as an easily deployable and flexible service that follows cloud principles and can be deployed in multiple different hardware platforms without major impact in terms of performance.

The previous sections presented and described the overall architecture, internal components and their interaction and mechanisms to support the required functionalities. These were later on evaluated in a set of scenarios to assess their functionality, performance and flexibility. From the evaluation results, one may conclude that ICNaaS performs as expected and there is not much impact in terms of performance due to the virtualization and cloudification processes. Moreover, it is possible to see that scaling mechanisms and the flexibility they bring is quite important to ensure that the performance is maintained and the system quickly converges whenever load changes.

In subsequent chapters, enhancements possible due to this service are examined and evaluated. These deeply affect the performance and flexibility of ICNaaS in terms of throughput and latency, contributing to the end goal of this thesis of greatly improving the performance of content delivery in mobile networks. In Chapter 5 below, the integration of ICNaaS (its router components) with LTE mobile networks to make enhance content delivery and make it a network funcitionality is described and evaluated.

Chapter 5

Integration of Information-Centric Networking into Long-Term Evolution Mobile Networks

With the current growth of mobile devices usage, mobile networks struggle to deliver content with an acceptable Quality of Experience. This problem strengthens the need for new technologies and concepts that may help enhancing the performance of content delivery while saving resources and using them more efficiently. As ICN is a relevant concept that can be used to create an improved and flexible content delivery system within existing networks, as discussed in Chapter 4, it makes sense to evaluate its possible combination with current LTE networks in order to profit from its benefits at a deeper level and thus make ICNaaS a real mobile networking service. In this chapter, the integration of ICN into those networks is explored and evaluated, considering that ICN's inherent caching feature can be harvested in close proximity to the end users by deploying ICNaaS network components inside the eNB. Apart from the advantages brought by ICN's content requesting paradigm, its inherent caching features placed at the edge also enable lower latencies to access content and reduce traffic at the core network, much in the direction of the work presented in Chapter 7. Therefore, gains can be expected for the performance experienced by the users and savings can be observed at the mobile operator's side. In Section 5.1 a brief motivation for this work is presented. Section 5.2 describes the modified EPS architecture and newly created mechanisms to address the challenges. Section 5.3 presents the evaluation of the proposed architecture and mechanisms. Finally, Section 5.4 concludes the chapter and sums up the achievements of this work.

5.1 Motivation

Mobile networks evolution has been quite intense in the last few years, with major increase of throughput performance and radio resource efficiency. Such evolution is mostly driven by tremendous demand of bandwidth, on the one hand because smartphones and other mobile devices play a major role as content demanders, and on the other hand because traffic-heavy applications are part of the daily life of millions of people. However, satisfying the content requirements of the current number of users while ensuring a great perceived quality for all of them is still a big challenge. With this challenge in mind, one may consider new concepts and technologies for improvements to the performance and efficiency of LTE mobile networks.

To achieve those goals a number of strategies and concepts can be considered. ICN and C-RAN, described in more detail in Chapter 2, are two of those concepts. While a new paradigm such as ICN can bring a range of new benefits that improve scalability and performance, C-RAN is an enabler to bring ICN in its full fledge capacity to LTE mobile networks as it provides the flexibility to integrate components and explore all their capabilities until the very edge of the network. Therefore, co-location and integration of ICN routers within the LTE architecture can support the full range of ICN benefits, with factors that amplify their impact and motive such integration. For instance, caches become much closer to the users requesting content and new edge caching strategies (as described in Chapter 7) become a reality.

That saves network resources at the core (less capacity needed) when downloading the content, while still delivering better end-to-end performance. According to Cisco Systems [Savic, 2011], one of the key issues to tackle the extreme demand for content is precisely how to perform content caching and delivery closer to the edge, and hence this integration and its benefits can help overcoming what today is a big challenge.

This chapter proposes an architecture and mechanisms to provide caching at the eNB level using both ICN and HTTP, despite the research challenges on doing so. First, compatibility with legacy systems and devices must be maintained (e.g. devices not aware of ICN, existing traffic flows, etc.). Second, the performance impact on the eNB should be low so that LTE processing constraints are not affected. Third, performance must be significantly improved and should compensate for the extra complexity while delivering a better quality of service to end users and leading to cost savings for the operator.

5.2 Architecture and Mechanisms

The integration of ICN into LTE has a number of challenges to overcome. The proposal described hereafter [Gomes and Braun, 2015a] aims at addressing these challenges by modifying the data plane flow in the eNB and integrating an ICN implementation into the EPS architecture. To serve as base for this work in terms of ICN, CCNx [PARC, 2015] was chosen as the implementation of ICN. It was also selected because it is a reference in the research community and its code is open source.

5.2.1 Traffic Interception

The 3GPP-defined data plane is designed to be very efficient, robust and include all the functionalities within the core. Hence, current components do not have interfaces to directly access the data plane outside the EPC. However, to get the traffic at the eNB and to to be able to use new paradigms at the edge, traffic interception is required. CCNx, with its current implementation, needs IP and UDP to operate. In the LTE protocol stack, IP is encapsulated inside other protocols for most of its path from the UE through the P-GW. At the EPC, it is inside the General Packet Radio Service Tunneling Protocol - User plane (GTP-U). At the eNB, as depicted in Fig. 5.1, it arrives encapsulated inside the Packet Data Convergence Protocol (PDCP), but there it is extracted and after the included Robust Header Compression (ROHC) it is relayed to be encapsulated inside GTP-U, which is part of the S1-U interface towards the EPC. As this relaying operation is performed, direct access is achieved to IP packets without modifications so far.

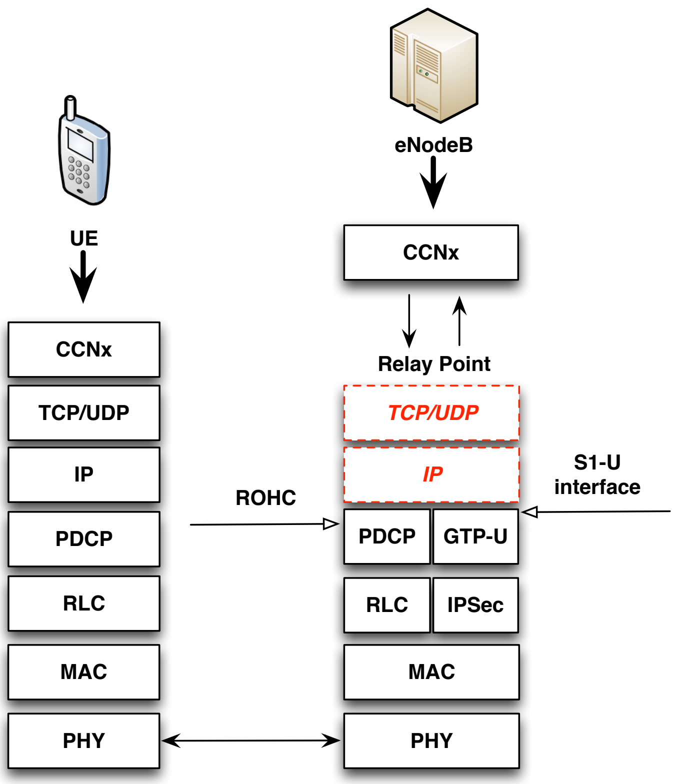


Figure 5.1: IP Packets Interception

5.2.2 Traffic Filtering and HTTP Compatibility

After intercepting traffic, it is important to distinguish between CCNx related and non-CCNx related traffic. CCNx related traffic is forwarded directly to CCNx routers. As for the non-CCNx related traffic, it may also be processed and, if possible, converted from a different protocol (e.g. HTTP) to CCNx by a proxy component.

In Fig. 5.2 a minimalistic representation of the modified EPS architecture is depicted. There are two different cases to be considered: a user with a device that has CCNx support and another user with a legacy device, which is not CCNx aware. In both cases, traffic normally flows until the eNB (red arrows for CCNx, blue arrows for HTTP). At the eNB, and as described in the previous subsection, IP traffic is intercepted at the relaying stage and forwarded to a component named ICN Proxy (purple arrow, letter A), which can also be integrated into the CCNx router. This proxy has a twofold function. First, it filters traffic and sends CCNx related traffic to the CCNx router co-located with the eNB (red arrow, letter C). Second, as an optional feature, if the received traffic is not CCNx traffic but rather HTTP traffic, it tries to convert HTTP requests to CCNx Interest messages using a set of defined mapping rules. A base implementation provided by CCNx works as follows: the URL is parsed, and the domain name (e.g. youtube.com) is considered to be the first part of the Interest message. The second part of the URL (e.g. /sports/soccer.avi) is directly appended to the newly created Interest message (e.g. ccnx:/youtube.com/sports/soccer.avi). In this last case, data coming through CCNx has to be converted back to a normal HTTP response before being sent to the user (blue arrow, letter B), which leads to additional processing time and additional overhead. Hence, the usage of this proxy resource should be avoided as much as possible and non-CCNx traffic should be put back without extra processing (also interface with blue arrow, letter B). In this architecture, that traffic then goes to a transparent HTTP Proxy/Cache, which acts as the HTTP edge cache. If the content object is cached, it can be directly sent back to the eNB and afterwards the user. Otherwise, the destination server is contacted to obtain the content object (can be a HTTP server or a Content Delivery Network server) through the normal EPC flow.

5.2.3 Traffic Forwarding within the EPC

After traffic has arrived at the CCNx router co-located with the eNB, three different possibilities exist and can be observed in Fig. 5.2. These are described below:

1. In the first option, the requested content object is not cached (red flow at the figure, letters E and F). In this case, the Interest message can be forwarded and sent back to the eNB to enter the EPC and follow its forwarding path (letter D in the figure). Then it goes through EPC, and after the P-GW (EPC exit point) it may reach another CCNx router located in the Internet. At this CCNx router, which includes a repository, the content is found and can be delivered back via the reverse route.

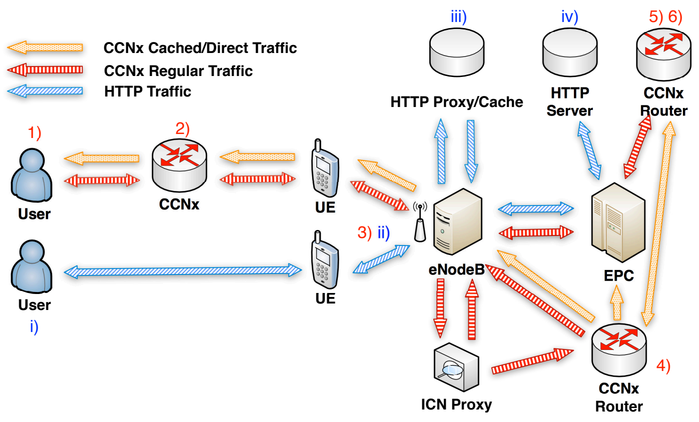


Figure 5.2: CCNx Integration in EPS

2. Another possibility (rightmost orange arrow), and only if routes exist and operator policies allow it, is to forward the Interest messages without sending them back to the eNB and putting them in the regular flow (letters D through F). In such case, the EPC is avoided and additional measures are needed. To be compliant with standards and to be able to charge the traffic to the users, an interface must exist between the CCNx routers and the EPC (grey arrow between CCNx Router with number 4 and EPC). This interface allows traffic charging and also lawful interception, two functionalities which are required by standards and by MNOs.

3. The last case to consider is the case where the requested content object is cached at the CCNx router co-located with the eNB (router number 4). In this case, it is immediately forwarded to the user via the eNB (orange arrow between CCNX Router with number 4 and the eNB). Again, to be compliant with standards and to allow charging, interfaces must exist towards the EPC. However, in this case, lawful interception should not be needed as content was already filtered when it was cached.

5.3 Evaluation

The architecture described in the previous section and its underlying mechanisms and modifications address the identified problems. However, experimentation is needed to validate its usefulness. To test the feasibility and the performance of the proposed solution, a test scenario was defined and will be described below.

5.3.1 Experimentation Scenario and Parameters

The scenario for the evaluation of the proposed system is depicted in Fig. 5.3, with number mappings to the components in Fig. 5.2 for better understanding.

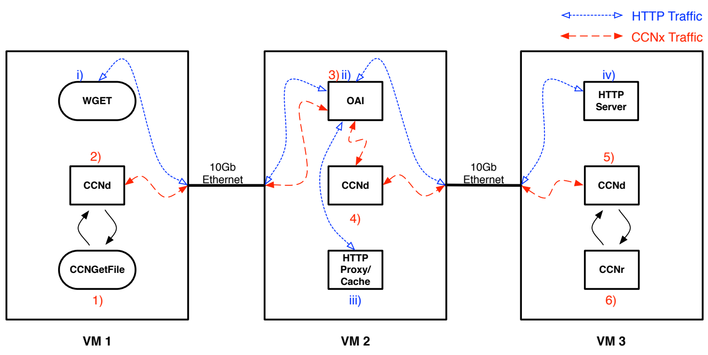


Figure 5.3: Experimentation Scenario

In Fig. 5.3, each VM corresponds to a different entity and two different paths are present - one for HTTP requests and another for CCNx requests. The first VM corresponds to a user asking for content, either using CCNx or using the Linux command WGET (for HTTP requests). The second VM is the eNB, and here, to have the most realistic scenario possible, OAI (c.f. Section 3.2) was used. OAI is an open-source platform developed by Eurecom as an emulator for the 3GPP LTE Radio Access Network. It combines emulation of the physical layer with emulation of Medium Access Control (MAC) and higher layers, providing a full stack to perform emulation-only experiments but also the possibility to use an external physical layer (hardware) to interconnect real devices. In this evaluation, only emulation of eNB components was considered due to the higher restrictions of the hardware, namely the requirement to have licenses to operate within LTE frequencies. At OAI, depending on the traffic (HTTP or CCNx), the request either goes to the HTTP Proxy/Cache or to the co-located CCNx Router according to the IP Tables rules described in Section 5.2. If the second option is valid, redirecting traffic from the eNB stack relay point (previously described) to the CCNx daemon (CCNd), will either return the content back to the user (if cached) or forward it to the CCNx router in the third VM, which has a CCNx storage repository (CCNr).

For evaluation, the architecture described in the previous paragraph was implemented in a simple test bed environment. This test bed consists of three VMs, each mapping to two 2.5 GHz Xeon CPU physical cores, 8 GB of RAM, solid state drives storage and a 10 Gbps network interface. This test bed is intentionally of small scale, because it aims at evaluating feasibility (not scalability) and single-component performance.

Experiments consist of evaluating two different metrics by calculating the averages of the measured values. The first is processing time of a given LTE module for each received/transmitted frame (in microseconds). From previously conducted experiments, three modules were considered the heaviest in terms of processing at the eNB, and hence they were selected to be evaluated. These are Orthogonal Frequency-Division Multiple Access (OFDMA) modulation, Single-Carrier Frequency-Division Multiple Access (SC-FDMA) demodulation and Receiving

Physical Random Access Channel (RX PRACH). Processing of these modules is rather important, as it gets affected by the delays of HTTP/CCNx traffic filtering and by the competition for processing resources at the VM. In particular, process switching overhead (governed by the kernel scheduler), kernel interruptions and other factors will have an impact on such processing. If processing time values are too high, they can be beyond the LTE requirements for processing delays and frames will be lost. One may argue that a different VM or a different CPU core can be used to isolate processing and thus prevent the issue of processes competing for shared resources, but other issues would appear. Using another VM would mean extra delay in intercepted traffic (it would have to flow between VMs), meaning that LTE processing time requirements would not be met. A second CPU core would mean that a real-time Operating System (OS) can dedicate that core to a single process, and such option is not a possibility with a public cloud, rendering the C-RAN concept useless.

The second metric is the time it takes to get a certain quantity of data (in seconds), from the time the request has been issued until the full content data is received. This metric is the key indicator for performance, while the first one is the key indicator for feasibility in terms of processing impact.

In all the experiments, a mix of content objects adding up to 1 GB of data had to be transferred either using ICN or HTTP (with and without the edge HTTP Proxy/-Cache), with 10 user threads requesting content objects simultaneously. This mix contains objects with one of the following randomly decided sizes: 5 MB, 10MB and 20MB. Also, for testing purposes only, we considered as a reference value that 20% of these are cached when ICN or HTTP caching are used (1 out of 5). Additionally, for each of the evaluations to be performed (processing time of one of the three modules and performance of content download), multiple parameters were changed. A very important parameter is the LTE modulation, which can either be Quadrature Phase Shift Keying (QPSK), 16 Quadrature Amplitude Modulation (16QAM) or also 64 Quadrature Amplitude Modulation (64QAM). In our case, in order to get the higher processing loads possible, we considered the maximum possible Modulation Coding Scheme (MCS) values for each of them, which are respectively 9, 16 and 28. Also, the used channel bandwidth was varied. We considered 5 MHz, 10 MHz and 20 MHz at the 2.6 GHz band, as these are the typical channel bandwidth values used in real LTE deployments.

5.3.2 Results

In Fig. 5.4, we may observe the processing time for OFDMA modulation (execution measured time it takes to perform OFDMA modulation for each LTE frame, on average) with impact from concurrent processing of HTTP/ICN mechanisms. The first noticeable point concerns the high processing times for 20 MHz when comparing to other channel bandwidth values. This growth of processing times between channel bandwidth values is in fact not proportional, and thus 20 MHz can be considered a challenging channel bandwidth value in terms of processing resources. Such fact explains partially the bigger gap between ICN or HTTP with caching and regular HTTP content transfers, as less processing resources are available at the system and the overhead caused by caching mechanisms is much higher. The other reason for 20 MHz to be a challenging channel bandwidth value is the quantity of traffic/time unit (e.g. Mbps), which increases with greater LTE channel bandwidth values (and better modulations) and also leads to more processing resources usage by either the CCNx daemon or the HTTP Proxy/Cache (more traffic/time means more processing requirements). At the same time, and surprisingly, the complexity of the HTTP Proxy/Cache and its requirement to maintain sessions (as per HTTP 2.0 protocol) deeply affects the processing times, leading to higher values than ICN with its internal caching mechanisms. This difference becomes important for LTE processing, and although optimizations can be explored to all the components, they will still have a non-negligible impact. We can also observe that using ICN instead of regular HTTP (no caching) adds extra processing time due to protocol complexity and caching mechanisms, which averages at around 7% more and in this case is not too relevant when the tradeoff between performance and feasibility is evaluated. Indeed, considering all the advantages brought by caching and by the ICN protocol, this result technically proves the feasibility of the proposed solution when OFDMA modulation is considered as the key LTE processing module for downlink communication.

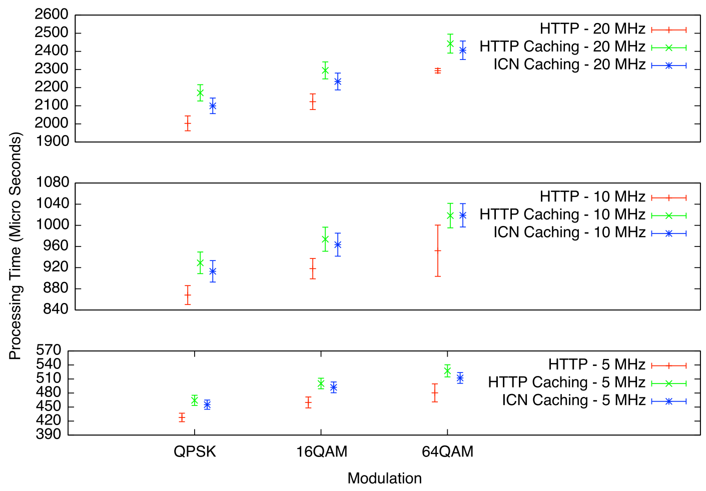


Figure 5.4: Test Results - OFDMA Modulation

In Fig. 5.5, SC-FDMA demodulation processing time (execution measured time it takes to perform SC-FDMA demodulation for each LTE frame, on average) with impact from concurrent processing of HTTP/ICN mechanisms is analyzed. Here, the first aspects one may notice are processing times overall. These are clearly higher than for OFDMA modulation in all the scenarios tested. Since uplink traffic is low, this can only be explained by the greater complexity of the algorithm or possible implementation issues at the LTE emulator. Such fact is, however, not

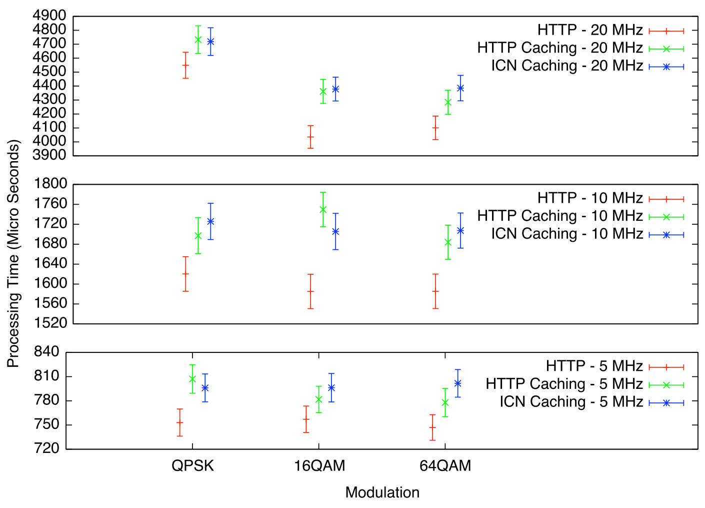


Figure 5.5: Test Results - SC-FDMA Demodulation

very important for the evaluation being performed here as downlink processing is the most important for content downloads. In this figure, we can also observe that in general SC-FDMA demodulation has smaller or no processing time gaps between ICN and HTTP with caching content transfers. Existing differences are, however, explained by the number of messages flowing upstream, which is higher for ICN due to the method used to implement pipelining. At the same time, Interest messages and HTTP GET messages over the proxy (for caching purposes) are quite comparable to regular HTTP GET messages (no caching), although the higher quantity of these messages (due to message forwarding from the eNB) and local processing may explain the existing difference that can be observed between protocols with and without caching. For SC-FDMA demodulation, the average of 5% processing increase from regular HTTP to ICN content transfers validates our proposal in terms of tradeoff between advantages/drawbacks and also justifies the previous statements. It is, however, not possible to conclude that SC-FDMA demodulation validates the adoption of ICN over HTTP with caching if performance impact is the only criterion.

In Fig. 5.6, RX PRACH processing time (execution measured time it takes to perform RX PRACH for each LTE frame, on average) with impact from concurrent processing of HTTP/ICN mechanisms is considered as the metric to be evaluated. Unlike OFDMA modulation and SC-FDMA demodulation, RX PRACH has less factors leading to processing times difference between ICN and both HTTP content transfers (caching and no caching) because of its nature (only used before a scheduled channel is assigned). This leads to a visibly smaller gap among all the scenarios with the three content transfer types, in this case with a difference of only 2% on average between ICN and regular HTTP. Also, results are very comparable between ICN and HTTP with caching, almost without deviations. Despite some observed variations, values are roughly similar within the defined error margins.

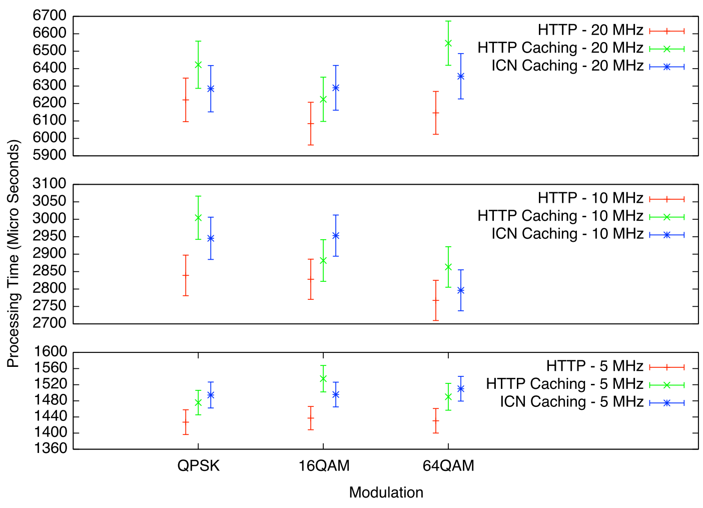


Figure 5.6: Test Results - RX PRACH

Therefore, the smaller influence of RX PRACH is always present and conclusions about the impact of ICN in RX PRACH can be considered valid.

Finally, in Fig. 5.7, a performance comparison is done in terms of content transfer latency (time between content request and download completion). In all the evaluated scenarios ICN brought advantages rather comparable to the ones of HTTP with caching, as any form of caching mitigates the drawbacks resulting from the higher overhead of traffic redirection and extra processing. The performance difference between the two is mostly visible with lower rates (derived from lower channel bandwidth values and worse modulations), a fact that can easily be explained by the availability of content at the eNB when cached by ICN or HTTP. With content available and ready to be delivered back to the user, there is no bottleneck resulting from packets coming in at a high rate (from the wired connection to the core) that have to be buffered and re-sent at a much lower rate (LTE radio). This factor is even more visible if looking at 16QAM and 64QAM modulations, as rates with these modulations are much higher and differences become almost negligible. In this case, only the latency between the second and the third VM, together with the service processing time, is creating an impact difference between usage of caching or not. Core traffic is still highly reduced (about 20% less in this case) and cost savings are high, ensuring that caching is still worth to be used.

From the results presented above, we may assume that the tradeoff between performance and processing time delay is in favor of ICN integration with its caching mechanisms. Although HTTP with caching may deliver slightly better results in terms of performance, it is also using more resources due to its requirement for sessions. Additionally, it does not bring extra advantages as ICN does. ICN is a value-added technology with important improvements on caching itself and with extra benefits such as improved mobility support, greater scalability and better security. And yet, even considering the additional processing of ICN in terms of re-

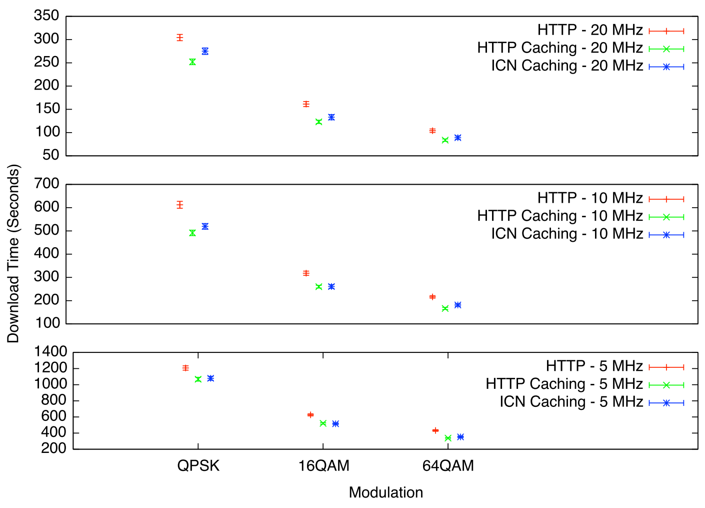


Figure 5.7: Test Results - Content Transfer Performance

quests forwarding and content delivery, the content transfer performance increase was on average 16% higher when comparing to regular HTTP, a number which is very comparable to the one of HTTP with caching (19%). At the same time, the impact in terms of processing resources by ICN (in comparison with HTTP with caching) was, on average, 5% lower in the worst case scenario (OFDMA modulation). We may also assume that these results could possibly be improved, as currently OAI has a caveat in terms of processing. Its physical layer is not multithreaded (highly dependent on CPU core frequency), and hence the possibility of compensating processing time delay with a larger number of CPU cores does not exist at the moment.

5.4 Summary

In this chapter, concepts for the integration of ICN into mobile networks were introduced, enabling many valuable benefits such as network edge caching and a novel paradigm for content requests. Caching is in fact the most valuable benefit, as referred studies demonstrate that its usage brings massive cost savings while improving performance.

A number of proposals exist to introduce caching within LTE mobile networks, but most look at it from a theoretical perspective, while others that go deeper into technical challenges only consider caching at the EPC level. Caching at this level is undoubtedly beneficial, but the biggest gains come from caching at the eNB level. To address this last topic, a proposal was presented using ICN as an enabler for caching and as a technology with greater benefits than HTTP, and challenges resulting from it were tackled and evaluated.

This proposal was evaluated in terms of feasibility and performance gains, considering multiple LTE modulations and channel bandwidth values while comparing to a non-ICN environment with the same content being transferred and also with the option of caching. Results demonstrate that while LTE processing times are generally higher and slightly more resources are consumed with an ICN-enabled environment, performance in terms of content access latency due to the effects of caching and the different paradigm are much better than with regular HTTP and comparable to the ones of HTTP with caching. In fact, most of the times ICN delivers better efficiency of resource usage due to session abstraction and greater scalability potential, resulting in both bandwidth usage and OPEX reductions. Also, the proposed architecture is compatible with legacy devices and specifications, ensuring that it can be adopted without disrupting existing devices and systems.

In Chapter 6 below, this integration is explored at the level of using multiple radio technologies together to ensure an efficient load balancing/offloading between them and leveraging the presence of ICN to perform it independently of the underlying protocols.

Chapter 6

Offloading and Load Balancing using Information-Centric Networking

Mobile networks usage rapidly increased over the years, with great consequences in terms of performance requirements. As cellular mobile networks are increasingly being used due to their wide coverage and availability, and may even be combined with content delivery concepts such as ICN (see Chapter 5), other networks such as Wi-Fi networks are typically only used when the users are at home or at work. This means that despite the coverage of Wi-Fi networks is quite big and access points surface at every corner of the streets, they are not fully explored. Projects from the operators and standalone companies attempt to foster the sharing of Wi-Fi networks, by creating a second network within the user's base station equipments that is completely independent and provides access to service subscribers. However, in most cases the user has to select either the cellular mobile network or the Wi-Fi, and even when mechanisms exist to offload to Wi-Fi, it is binary offloading (all or nothing). As it makes sense that any integrated approach to improve content delivery in mobile networks addresses this issue, in this chapter the proposal is to use ICN as an aggregating technology that can also provide offloading but at the same time introduces seamless and technology-agnostic load balancing. In Section 6.1 a brief motivation for this proposal is given. Section 6.2 describes its integrated architecture and the load balancing mechanisms. Section 6.3 presents the evaluation of the proposed mechanisms in a simulated environment. Finally, Section 6.4 concludes the chapter and highlights the key achievements of this work.

6.1 Motivation

Mobile networks evolution has been quite intense in the last few years, with major increase of throughput performance and radio resource efficiency. Such evolution is mostly driven by tremendous demand of bandwidth, on the one hand because smartphones and other mobile devices play a major role as content demanders, and on the other hand because traffic-heavy applications are part of the daily life of millions of people. However, satisfying the content requirements of the current number of users while ensuring a great perceived quality for all of them is still a big challenge. Combination of LTE and ICN, as proposed in the previous chapter, already provides a way of extending newly researched benefits to mobile networks.

Although this combination already has its own benefits by default, a particular benefit with a very promising outcome arises as an opportunity – seamless load balancing between radio technologies. This benefit can be explored due to ICN's inherent characteristics, which overcome the big challenge of having load balancing between a number of different network stacks. This occurs because ICN does not depend on the underlying protocol, and abstracts networks stacks that may not expose IP directly. Moreover, it also abstracts the transport protocols, and competes with solutions such as Multipath TCP [Han et al., 2006] by working regardless of TCP usage (ICN approaches typically use UDP to reduce overhead and increase performance). At the same time, it addresses the challenge of dealing with user mobility by not requiring sessions as TCP does. Still, ICN unrelated challenges also exist. The most obvious challenge is how to combine the different network links, which typically will have different characteristics and hence different throughputs, but common objectives such as saving costs by reducing the usage of LTE bandwidth, reducing energy consumption of mobile devices and ensuring a fair usage of resources also pose challenges requiring solutions.

Today, one form of load balancing, which is being explored and already deployed in some infrastructures, is Wi-Fi offloading [Lundström and Hall, 2011]. With Wi-Fi offloading, mobile operators are able to migrate users from overloaded LTE networks to existing Wi-Fi networks and still deliver an acceptable perceived network quality to their customers. This is still not the most efficient way of load balancing because both technologies are not used simultaneously, and, therefore, this chapter proposes an alternative for load balancing between technologies using ICN as an enabler and as a value added technology.

6.2 Architecture and Mechanisms

Any sort of load balancing in LTE mobile networks has challenges to overcome, either if using ICN or not. Using ICN already addresses some challenges by default, but other challenges remain. The proposal described hereafter [Gomes and Braun, 2015b] intends to address mainly the combination of two interfaces (links), the ratio to use when splitting content Interest messages (and corresponding segments) among those interfaces and how to make an ICN implementation compatible with such solutions. However, it also addresses other challenges indirectly, such as cost savings (LTE bandwidth savings), fair usage of resources (usage of a given technology according to network policies) and lower energy consumption. To lower energy consumption, we assume two factors: 1) Wi-Fi energy usage is typically lower [Huang et al., 2012], and when Wi-Fi throughput is good, Wi-Fi will be selected as the technology to deliver most (or all) of the data. 2) When Wi-Fi throughput is low, energy consumption is higher than if using other technologies such as 3G and LTE [Ding et al., 2013], and with our proposed strategy the usage of Wi-Fi will be low or null.

To serve as basis for this work, once again CCNx [PARC, 2015] was selected as the implementation of ICN, because it is a reference in the research community and its code is open source. This code was modified to handle load balancing, as depicted in Fig. 6.1.

By default, CCNx already handles the process of sending Interest messages over multiple interfaces (faces), with the possibility to use a number of different forwarding strategies. With our load balancing strategy, multiple faces for the same content prefix will be used when sending Interest messages that match that content prefix, which means that multiple faces will be simultaneously used to deliver the segments that are part of the content object.

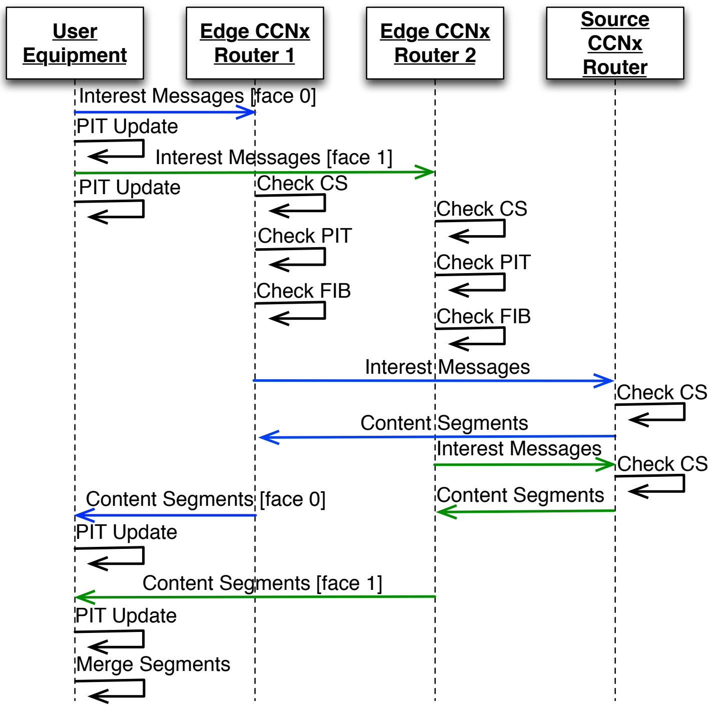


Figure 6.1: CCNx Load Balancing

With two radio technologies such as Wi-Fi and LTE, the UE will then send different Interest messages for specific segments of the content object over the two available faces. For each message, a record is inserted in a PIT to keep track of requested content. Interests will eventually reach the edge routers of each specific technology (close to the eNB in case of 3GPP LTE and to the ePDG in case of Wi-Fi). These routers will check their CS to check if they already have the content object segments cached and the PIT to see if the content object segments already have been requested. In this example, we assume that they will not have the content object segments and that the content object segments have also not been requested. Therefore, they will check their FIB for routes to request the content object segments and forward Interest messages to other CCNx routers in order to get them. As in this example the next CCNx router (Source CCNx Router) has the content object segments in its CS, the requested content object segments are then delivered to the edge CCNx routers and afterwards to the UE, with both updating their PIT table to mark the Interest messages as being satisfied. The UE will also merge the content object segments to present the requested file to the user.

Although this strategy enables the content object segments to be delivered over two interfaces and thus performs load balancing by aggregating the links, it is still neither efficient nor reliable because of the different link conditions. For instance, the 3GPP LTE link may have poor quality and hence performance will be possibly worse than without load balancing. Therefore, the amount of data to be requested over it should be less than the amount requested over a good quality Wi-Fi link. To decide on this ratio, a very lightweight score-based equation was developed:

$$S_i = \alpha * \frac{R_i}{R_{max}} + \beta * (1 - \frac{RTT_i}{RTT_{max}}) + \gamma * (1 - E_i) \tag{6.1}$$

where:

S_i is the score of link i, ranging from 0 to 3.

R_i is the peak transmission data rate of the link obtained using a certain Modulation and Coding Scheme (MCS), a predefined channel bandwidth and a given Multiple Input Multiple Output (MIMO) setting for the specific radio technology. For Wi-Fi, it is derived from the standard [Kraemer et al., 2009]. For LTE, it is calculated according to the 3GPP specifications [3GPP, 2015].

R_{max} is the maximum transmission data rate among the radio technologies to be used, based on the maximum MCS and under ideal conditions.

RTT_i is the round-trip time of the previously requested segment over link i.

RTT_{max} is the maximum round-trip time to be considered, which can be set to a high value such as the one for the timeout of the Interest messages.

E_i is the percentage of transmission errors obtained from the face (interface of link i) statistics at CCNx, which is represented as a value between 0 and 1. In real scenarios, Packet Error Rate (PER) [Lampe et al., 2003][Blankenship et al., 2004] can be used instead and thus offer a physical layer estimation for this value.

α, β, γ are weights for the parameters described above and $\alpha + \beta + \gamma = 1$.

This equation then assumes that three factors will affect how a link performs: transmission data rate, round-trip time and transmission error rate. Such factors are key factors for content transferring over any network, and with ICN this is not an exception. While round-trip time will mostly affect the experienced latency of the end user when dealing with live multimedia streaming (voice or video), the other two factors will always be crucial for any content transfer performance and hence user experience improvement. Also, data rate and overall throughput (also determined by the need of retransmissions caused by transmission errors) of a given link will play a major role in energy consumption of mobile devices. Therefore, they were given more weight than round-trip time to always give priority to the link, which delivers better performance with lower energy consumption.

Every network link will then get a score that can be used to calculate the ratio of Interest messages to be requested over each of those links at a particular time. As mobile networks are quite dynamic, every time a new segment is requested the score should also be re-calculated using Exponential Moving Average (EMA), ensuring that the best possible ratio is used. Getting back to our example, the ratios to split the Interest messages over LTE and Wi-Fi would be given by:

$$Ratio_{LTE} = \frac{S_{LTE}}{S_{LTE} + S_{WiFi}} \tag{6.2}$$

$$Ratio_{WiFi} = \frac{S_{WIFI}}{S_{WIFI} + S_{LTE}} \tag{6.3}$$

These ratios should be partially negotiated between the core network entities and UEs, meaning that the core network entities may dynamically change them by taking into account policies to decide on whether load balancing can be used or another single technology should be preferred. Factors such as a big score difference between links and user-defined policies at the UE may trigger the usage of a single technology. In this particular case, as there is no load balancing, performance may only be better if offloading to a link with a much better score is achieved. It is also worth mentioning that original scores are calculated directly at the UE based on the mentioned negotiations with the core network entities, and therefore each UE will have a score for each particular link.

6.3 Evaluation

To test the feasibility and the performance of the proposed solution, an experimentation scenario was defined and will be described below.

6.3.1 Experimentation Scenario and Parameters

The architecture for the evaluation of the proposed system is depicted in Fig. 6.2, with the main purpose of replicating a possible integration of CCNx routers with the EPS architecture. Hence, 100 users were considered with UEs attached to two different CCNx routers, one representing a CCNx router attached to LTE eNBs and another close to an ePDG for Wi-Fi integration, i.e. 2 hops away. These 100 users are each connected with up to two interfaces, either using LTE or using Wi-Fi.

However, these links are shaped to assume particular values of bandwidth (Mbps), delay (ms) and transmission errors (%).

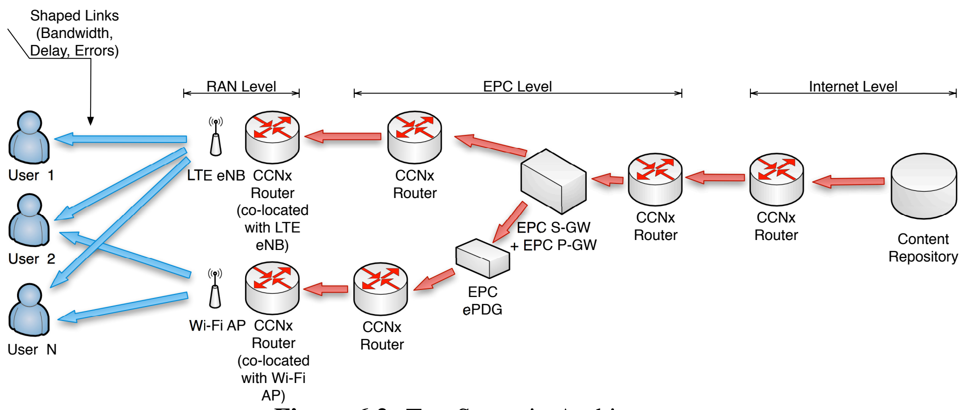


Figure 6.2: Test Scenario Architecture

In addition to the first layer of CCNx routers at RAN level, three more layers exist. The following two are CCNx routers at the EPC level, with one positioning in front of the S-GWs and another behind the P-GWs. With such positioning, GTP-U is not broken and traffic may flow normally. The last layer is at the Internet level, and in this case (for simplification purposes) has direct access to the content repository, which is the content source for the content object that will be retrieved by the users. It is also worth mentioning that in other scenarios, to improve scalability, the number of CCNx routers may be higher and more connections between them may exist, which means load balancing may occur at yet another level.

The architecture was implemented using the simulation environment described in Section 3.3 because the necessary evaluations could not be performed at this scale using a testbed environment such as the one from MCN. These simulations consists of 100 ns-3 user nodes, directly connected to CCNx making usage of ns-3 DCE. The links between the nodes and CCNx routers were shaped to assume particular values of bandwidth, delay and transmission error rate (fed to ns-3 stochastic error model and setting error units to packets). To test different shaping environments, particular values were chosen for both Wi-Fi and LTE links. With these values, three different shaping environments were created as shown in Table 6.1:

Table 6.1: Shaping Environments

Environment	Technology	Rate (Mbps)	Latency (ms)	Errors (%)
Proximity	LTE	100.8	10 ± 5	0
	Wi-Fi	300	5 ± 2	0
Urban Area	LTE	50.4	20 ± 5	1
	Wi-Fi	28.9	25 ± 2	0.5
Rural Area	LTE	33.6	35 ± 5	2
	Wi-Fi	86.7	15 ± 2	1

A first shaping environment reproduces close geographical proximity to both the LTE eNB and the Wi-Fi Access Point (AP), also considering that low load is present. A second shaping environment reproduces a typical urban area, where the LTE eNB may still be close to the users but with high load and low resource availability. As for Wi-Fi, it is available with a typical bandwidth but shared by a large number of users (lower throughput). The third shaping environment reproduces a typical rural area environment in which users are very far away from the base stations, with both LTE and Wi-Fi bandwidth availability being low.

Although these shaping environments define different radio profiles for the UEs, it is also worth to evaluate how the system performs with different numbers of UEs in each shaping environment. Therefore, it was considered that 30% of the UEs are in the Proximity shaping environment, 50% are in the Urban Area shaping environment and finally the remaining 20% are in the Rural Area shaping environment. From here, different experimentation scenarios were derived. A first set of three experimentation scenarios intend to evaluate one of the weight factors of the score equation with all the shaping environments:

- **Scenario 1:** Evaluate α by making it vary from 0 to 1 while keeping the other weights with the preset value.
- **Scenario 2:** Evaluate β by making it vary from 0 to 1 while keeping the other weights with the preset value.
- **Scenario 3:** Evaluate γ by making it vary from 0 to 1 while keeping the other weights with the preset value.

Additionally, other experimentation scenarios were evaluated. One aims at showing the evolution of the strategies' performance for growing content object sizes. The other evaluates the usage of LTE resources considering different strategies for UEs with each of the shaping environments, as from the mobile network operator point of view this must be minimized in order to save costs.

For each of the experimentation scenarios defined above, and for a set of strategies, multiple tests were conducted. These tests ran for 8 hours of simulated time, and were repeated 50 times each. The strategies being considered are: LTE only, Wi-Fi offloading and ICN Load Balancing. In the first strategy, all the UEs are connected only via LTE. In the second, UEs are offloaded to Wi-Fi whenever the conditions allow it: Wi-Fi resource load is not very high (below 50%), UEs are within range of Wi-Fi base stations (Received Signal Strength Indicator (RSSI) above -95dBm), ePDG infrastructure is available and policies set for the users permit offloading. In the third, UEs use a combination of LTE and Wi-Fi with our load balancing strategy. An additional strategy is also plotted only for comparison purposes, reflecting the performance of what would be the theoretically optimal strategy. These values were obtained by considering that content would be retrieved combining the full rates of both technologies and not considering transmission errors, delays or any sort of protocol overhead. For each particular scenario, the theoretic optimum is then given by:

$$T_i = \frac{Size_i}{\sum_{j=0}^{n} Throughput_j} \tag{6.4}$$

where:

T_i is the time in seconds to download content object i.

$Size_i$ is the size of content object i.

$\sum_{j=0}^{n} Throughput_j$ is the sum of throughputs of all the access links to be used.

During each of the tests, popularity and caching impacts were purposely minimized, with the requested content objects being selected randomly from a set with content sizes following a normal distribution $N(50, 15)$ MB and caching enabled by default with a CS size of 10 MB. Each UE also requested a content object at a random interval between 5 to 30 seconds. Using content objects of multiple sizes aims at finding how load balancing evolves with content object size, as overhead of CCNx is predicted to have less impact when content sizes get bigger.

To evaluate each strategy's performance, the time it took for the user from the initial Interest message until it got the full amount of data of the particular content object was selected as the metric for the first four experimentation scenarios. The value considered for the graphs was the average of all the values collected during the simulation period and for all the connected UEs. As for the last experimentation scenario, the metric was the percentage of LTE usage within the previous scenarios (on average).

6.3.2 Results

In Fig. 6.3, a noticeable difference can be observed when comparing the different strategies. In fact, due to the average higher data rates of Wi-Fi, offloading users and performing load balancing can easily introduce a quite high performance gain.

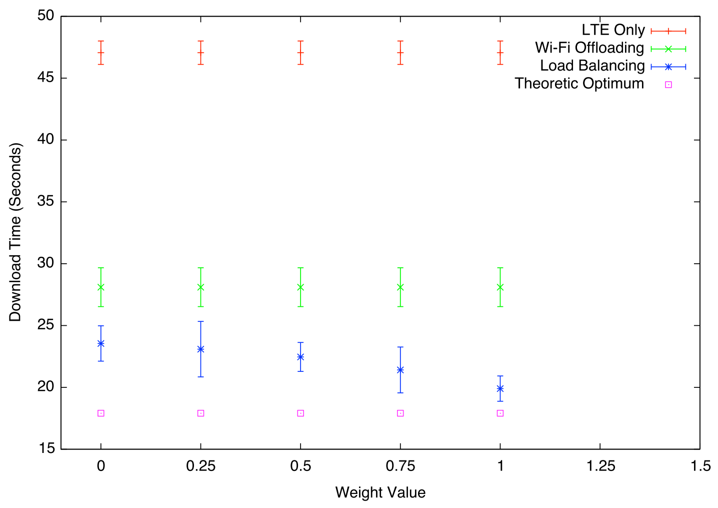


Figure 6.3: Test Results - Scenario 1

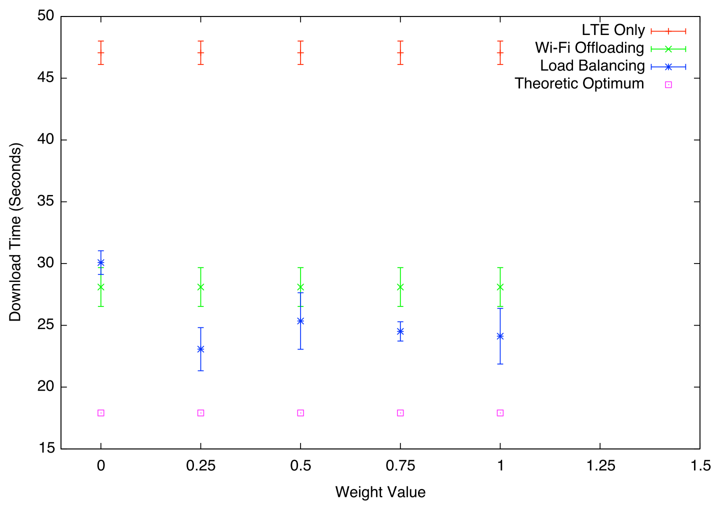


Figure 6.4: Test Results - Scenario 2

The performance gain of the load balancing strategy is about 70% in terms of content object download time on average, and it reaches a performance closer to what would be the theoretic optimum. The Wi-Fi offloading strategy is on average 40% better than using LTE only in this scenario, which is still a good improvement over the traditional setup but not efficient enough to improve shaping environments overall network performance. At the same time, it is possible to notice that performance also shows an increase when α becomes higher. From that, we may conclude that giving more weight to data rates when calculating scores is the right decision to improve overall performance.

In Fig. 6.4, with all the settings of the previous scenario but now evaluating the impact of β, once again a noticeable difference can be observed when comparing the performance of the different strategies. However, as this time the impact of the varying weight factor is lower, the performance is also slightly lower than before. It is now on average 66% higher than using LTE only. Performance is also further away from the theoretic optimum, which translates into less efficient resources usage. Finally, it is also possible to conclude that some weight factors for β negatively impact the load balancing strategy in such a way that only with Wi-Fi offloading the performance is better. Therefore, only β values of 0.25 and 1 could be considered as valid factors, with preference to 0.25 as with this factor the load balancing strategy's performance is much closer to the theoretic optimum.

In Fig. 6.5, which refers to scenario 3 and evaluates the impact of γ, performance increases are again very noticeable. However, performance values of Wi-Fi offloading and ICN Load Balancing are very similar. This is caused by a low impact of the weight factor, also due to the reduced values of transmission errors defined in the shaping environments. Nevertheless, it is possible to identify a factor for γ that still brings a noticeable performance improvement for the ICN Load Balancing strategy. This value is 0.5, and will be considered as the optimal value for γ in the remaining experiments. As $\alpha + \beta + \gamma = 1$ is a condition, factors were normalized so that their sum is 1, and for the next experiments $\alpha = 0.57$, $\beta = 0.14$ and $\gamma = 0.29$.

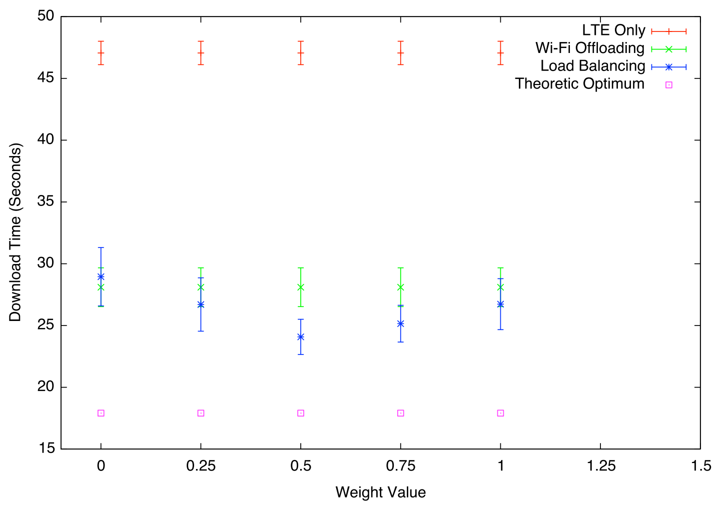


Figure 6.5: Test Results - Scenario 3

In Fig. 6.6, which refers to the scenario where the impact of content sizes is evaluated, performance improvements are present in similarity with the previous scenarios. These performance improvements can be seen in both the Wi-Fi offloading and ICN Load Balancing strategies. We may also observe a slight increase in performance when the content object size increases due to lower impact of the protocol overhead, although the ICN Load Balancing strategy's performance is always very similar to what would be the theoretic optimum.

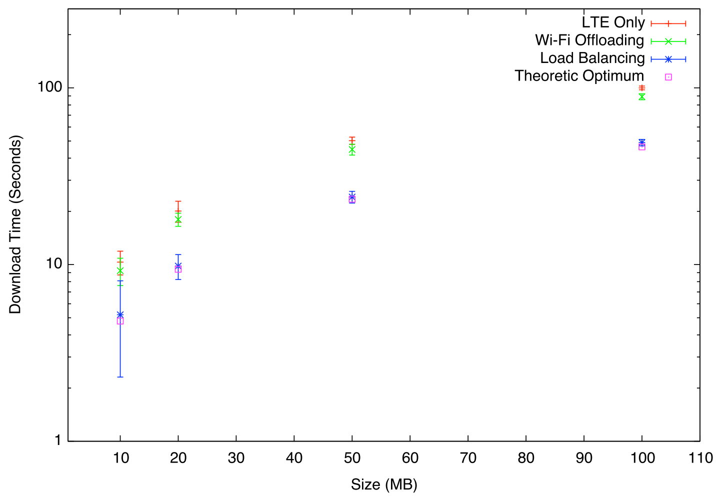


Figure 6.6: Test Results - Content Sizes Impact

Finally, Fig. 6.7 presents a comparison of LTE overall usage for the ICN load balancing and Wi-Fi offloading strategies. It is based on the percentage of LTE bandwidth (from the total bandwidth required by the UEs) required by the connected UEs in each of the shaping environments. Minimizing this percentage is a key objective for the operator to save costs and to maintain the overall performance of the cellular network.

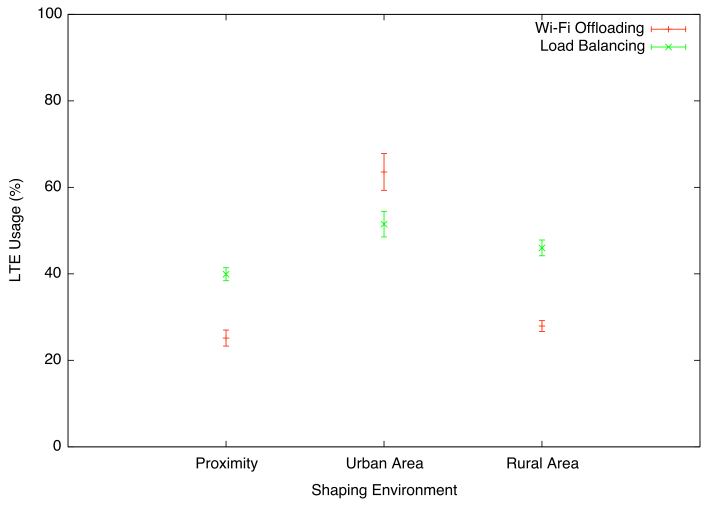


Figure 6.7: Test Results - LTE Usage

When considering the Wi-Fi offloading strategy, which is a typical strategy to be used nowadays due to the widely available Wi-Fi hotspots, we observe that in our shaping environments the usage of LTE resources varies between about 25% and 63%, with the higher value justified by the lower Wi-Fi data rate in the Urban Area shaping environment. Such results can be considered quite good in terms of LTE resources savings, but ideally they would be obtained with the ICN Load Balancing strategy to bring additional advantages (in terms of performance and energy consumption, for instance). This is not the case in two of the shaping environments, with about 15% of increase in LTE resource usage. By the contrary, again with the Urban Area shaping environment and due to the same factor, LTE resources are less used by the ICN Load Balancing strategy in that particular case. Overall, the difference between Wi-Fi offloading and ICN Load Balancing on lowering LTE resources usage it not very high, and the latter still delivers great LTE resource savings. Depending on the scenario, it may even behave better than Wi-Fi offloading and is a better option if all the advantages and disadvantages are taken into account.

6.4 Summary

This chapter introduced concepts for the combination of ICN and mobile networks, putting different technologies together and gathering many valuable benefits from the combination.

The proposed load balancing strategy was evaluated and compared to other strategies by simulation of multiple environments and scenarios. This evaluation shows that performance gains for content delivery are high and resources are used more efficiently from the network point of view. It was then demonstrated that cost savings for MNOs in terms of Capital Expenditures (CAPEX) and OPEX are possible due to less usage of expensive LTE resources, while delivering an improved performance for end users, ensuring also fairness of resource allocation from the mobile operator point of view. At the same time, energy consumption reductions can be achieved by giving priority to radio technologies with lower energy footprints. Therefore, such strategy could be beneficial for all the stakeholders involved, from the MNOs to the end users and also considering content providers, which aim at delivering high quality content without major constraints.

In the next chapter (Chapter 7), strategies to complement this work and the work of Chapter 5 are presented. These aim at improving edge caching with ICN in LTE mobile networks, thus ensuring that content will be available in a nearby cache as much as possible. Having the most requested content towards the edge then allows for load balancing to be effective, as it is likely that content will be at all the routers available for load balancing.

Chapter 7

Edge Caching Strategies

Nowadays CDNs are a big part of the Internet, and deliver most of the content across the world. However, CDNs and their caches are usually placed at the borders of networks, i.e. between networks belonging to different operators. Therefore, benefits from caching closer to the users and with higher granularity are not explored. Rather, CDNs have huge storage capacities and tend to group content by large geographical locations. This approach is highly ineffective and raises costs for mobile operators to have enough backhaul to support all the traffic generated by content within their networks. Moreover, users sometimes get frustrated by the lack of performance when downloading content, in particular when using mobile networks that are more prone to this issue. Hence, approaches such as the ones from Chapters 5 and 6 can be used to have a highly optimized content delivery system at the edge of LTE mobile networks, enabling caching of the most popular content and, as described Chapter 6 above, load balancing between different edge caches using different radio technologies. Hence, it makes sense to consider enhancements for these edge caches with strategies that optimize their usage and increase cache hit ratios.

In this chapter, edge caching strategies based on persistent caching are proposed to extend content delivery and caching to the edge of mobile networks and in close proximity to mobile users. It is also important to mention that, although this work fits directly with other contributions of this thesis, it was developed in close collaboration with my colleague Carlos Anastasiades [Anastasiades, 2016] and a bachelor's student (René Gadow) supervised by both of us [Gadow, 2015]. Hence, credit for the contributions hereby described and their impact is equally shared among us. In Section 7.1 a brief motivation for the work is given. Section 7.2 describes the internal architecture of the persistent caching mechanisms. Section 7.3 presents the evaluation of the proposed mechanisms in a realistic environment using a cluster and CCNx [PARC, 2015]. Finally, Section 7.4 concludes the chapter and provides an overview of our findings.

7.1 Motivation

With the vast proliferation of mobile devices in recent years, mobile data traffic has increased drastically and is expected to increase even more in the following years. According to Cisco's Global Mobile Data Traffic Forecast report [Cisco Systems, 2015], LTE will be more than half of the total mobile traffic by 2017 and the average traffic amount per smart phone will increase fivefold by 2019. Traffic studies, e.g., [Cha et al., 2007], show that many downloads are performed in the same popular files. To reduce traffic and increase performance, ICN caching can be integrated into LTE mobile networks as described in Chapter 5 and edge caching becomes a reality. However, caches need to operate at line-speed, thus, current memory technologies impose limitations. Fast memory is expensive, power hungry and only available in small capacities [Perino and Varvello, 2011]. Furthermore, caches are implemented in volatile storage, which is cleared, i.e., data loss, in case of power outages.

Therefore, in some scenarios, short-term caching may not be enough and content needs to be persistently stored (at the expense of slightly slower access times). This is particularly true if one wants to enable high availability and performance similar to CDNs by dynamically storing content in regions of high demand. Hence, persistent edge caching should be investigated for optimized content distribution and delivery.

We envision to extend hierarchical networks such as LTE mobile networks by adding persistent caches as shown in Fig. 7.1.

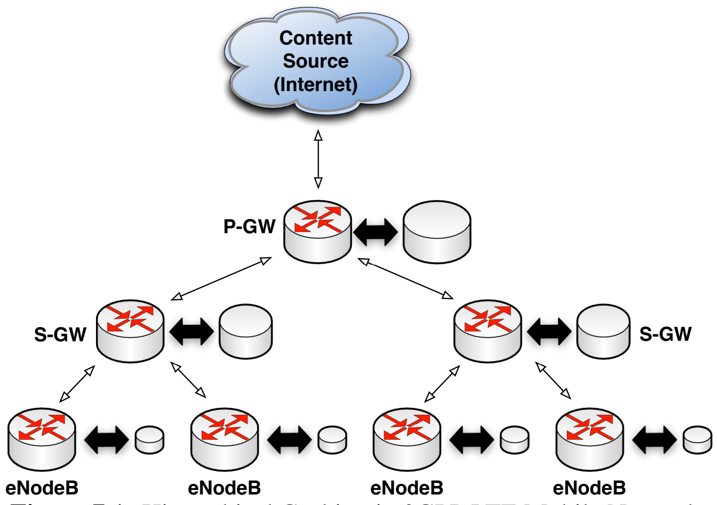


Figure 7.1: Hierarchical Caching in 3GPP LTE Mobile Network

Traffic from users is forwarded in a hierarchical way from eNBs S-GWs and from there to a P-GW. The P-GW provides connectivity to external networks such as the Internet. Persistent storage may be deployed alongside content routers attached to eNBs, S-GWs and P-GWs, and provides a repository to store a subset of content forwarded through these routers (similar to the CS). This enables storing very popular content of the day, e.g., electronic newspapers or popular videos, at the edge of the network to improve network performance. It also means that many requests of popular content may be satisfied already at edge routers, while requests for less popular content may be forwarded further to the next content router, which may hold a cached copy of the content. Therefore, only unpopular content, for which caching would not yield any benefits, would need to be retrieved all the way from the content source. Such an approach brings multiple advantages from the perspective of both end users and MNOs. For the former, perceived performance significantly improves due to lower content access latency, either delivered directly by caching or by one of its side effects, i.e. backhaul traffic experiences a major reduction, allowing faster content downloads from more distant sources. For the latter, OPEX can be reduced up to 36% [Sarkissian, 2013] due to the lower load of the network infrastructure.

7.2 Architecture and Mechanisms

In CCN, persistent storage is provided by repositories. The repository implementation in CCNx 0.8.2 [PARC, 2015] stores all content in the *repofile* and maintains references to the content in a B-tree.

Content sources publish content in repositories to make them available to other nodes. To use repositories for caching, content deletion needs to be introduced in an automatic way, e.g., based on popularity [Anastasiades et al., 2015]. However, we do not maintain popularity counters for two reasons. First, popularity counters would need aging mechanisms, introducing significant additional complexity [Podlipnig and Böszörmenyi, 2003]. For example, content that has been requested extensively one year ago may be less popular in the near future than content that has been frequently requested in the last hours, although the absolute number of requests would be lower. Aging timers would require last access timestamps in case multiple objects have expired (to know what to delete first), and would create a massive overhead and processing requirements in the already complex data structures. We prefer simplicity over complex solutions to minimize the processing overhead in content routers. Second, content requests that reach the repository (persistent cache) would not reflect the effective number of requests due to regular caching in the content store. In case of multiple concurrent requests, only the first request would be forwarded to the persistent cache while the others may be satisfied from the content store until the content is replaced.

Therefore, in this work, we maintain access information and delete content that has not been requested recently. There are two main differences to LRU strategies. First, deletion operations are performed based on content and not individual chunk popularities. Second, multiple content objects may be deleted at the same time to free space if a certain storage utilization threshold is reached because deletions in the filesystem take more time than in main memory (ms vs. sec).

7.2.1 Data Structures

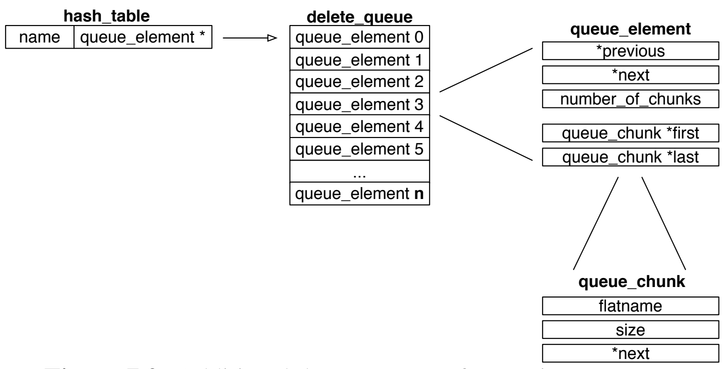


Figure 7.2: Additional data structures for persistent storage

Fig. 7.2 illustrates data structures required to enable content deletion for persistent storage. The *delete_queue* maintains an element for every content object in the *repofile*. The basic idea is to move requested content to the tail (bottom) of the queue such that unpopular content can be found at the head (top) of the queue. Therefore, if content needs to be deleted, it can be removed from the head.

Fig. 7.2 shows that the *delete_queue* is implemented as doubly linked list, on which every *queue_element* has a pointer to the previous and next *queue_element*. In addition, every *queue_element* has a pointer to another linked list of *queue_chunks*, i.e., the individual chunks of the content. Besides a pointer to the next element, we also maintain a pointer to the last chunk in the list to avoid long list traversals when including chunks of large content. The *queue_chunk* contains the flat name of a chunk to find the content (and its reference to the *repofile*) in the B-tree. When we need to find a *queue_element* quickly, we use a *hash table* to get its reference in the *delete_queue* based on a lookup of the base name, i.e., content name without chunk numbers.

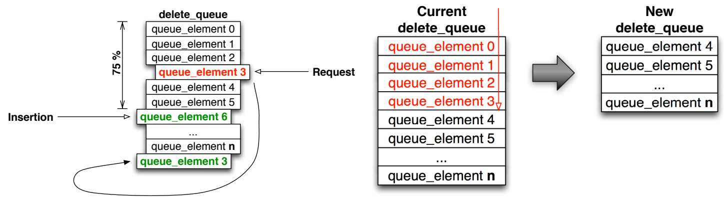
In contrast to related work on CCN caching, we keep content based on object granularity and do not make individual caching decisions for every chunk. Because content in CCN is requested sequentially based on the pipeline size, high variability in chunk downloads would degrade overall download performance.

7.2.2 Delete Queue Processing

In this section, we describe processing procedures in the *delete_queue*.

Inclusion

Content information is stored based on object granularity. When a chunk is received, the content name, i.e., base name without chunk number, can be extracted. Based on a hash table lookup, *delete_queue* entries of existing content can be found quickly. In this case, only a new *queue_chunk* needs to be added to the



(a) Insertion and Update (b) Deletion

Figure 7.3: Delete Queue Processing

delete_queue entry. If it is new content, the content is included in the *delete_queue*. As Fig. 7.3(a) shows, we include new entries in the middle of the lower half, i.e., at 75% of the queue. If content would be included in the upper 50% of the delete queue, new popular content could be deleted almost instantly, e.g., if the inclusion is just before a content deletion, since up to 50% of the repofile is deleted during a deletion operation. In addition, it is not appended to the tail such that unpopular content can quickly reach the head of the *delete_queue*, while popular content can go to the tail.

Queue Update

Fig. 7.3(a) illustrates also update operations on the *delete_queue*. Every time content is requested, the corresponding element in the *delete_queue* is pushed to the tail of the queue. This push operation can be performed for every requested chunk, every n-th chunk or only the first chunk. If every (or every n-th) chunk would be processed, there would be a tendency of larger files at the tail of the queue, since they have more chunks and, thus, more pushing operations. Therefore, we decided to consider only the first chunk of a content object as trigger for pushing operations.

Deletion

A deletion operation is initiated if the *repofile* has reached a certain size, i.e., the *repofile threshold*. Then, a deletion operation is performed by deleting a configurable percentage of the *repofile*, i.e., the *deletion ratio*. A deletion operation is initiated after a file inclusion, if the *repofile threshold* has been exceeded. In CCNx, file sizes are only known when the last chunk has been received with the *final bit* set. Therefore, the *repofile threshold* is a soft threshold and the *repofile* size can become slightly larger than the threshold depending on the size of included files, i.e., we do not perform deletion operations during file inclusions but rather afterwards.

Fig. 7.3(b) shows modifications on the *delete_queue* due to deletion operations. A deletion operation is performed by the following steps.

1. Prevent the repository daemon from accepting new content while the deletion operation is being performed. If new content arrives during local dele-

tion operations, it will only be stored temporarily in the content store. However, other repositories on the path to the requester will store it persistently.

2. Start at the head of the *delete_queue* and iterate through the elements until the *deletion ratio* is reached. All content entries up to this point will be deleted (red part in Fig. 7.3(b)) and the lower part becomes the new *delete_queue*.

3. Every *queue_element* contains multiple *queue_chunks*. The *queue_chunks* of all deleted content objects need to be sorted based on their position in the *repofile* such that the repofile can be sequentially processed (see next step) and every B-tree entry needs to be processed only once. In our current implementation, we use the $\mathcal{O}(n \log n)$ merge-sort algorithm for sorting.

4. All content from the *repofile* (except deleted chunks) are copied to a new *repofile*. This is required because file systems do not support selective deletions inside files. Due to deletions, content is copied to other positions in the new *repofile*, thus, reference values in the B-tree need to be updated.

5. All B-tree entries of deleted content are removed.

6. Instruct the repository daemon to start accepting new content again.

To limit service interruptions from deletions, a (read-only) repository can be started to provide content from the old *repofile*. Otherwise, Interests may just be forwarded and satisfied by persistent caching at the next router level. Thus, only in the worst case Interests would be forwarded all the way to the content source.

7.3 Evaluation

Persistent caching has been implemented by extending the repository implementation in CCNx 0.8.2 [PARC, 2015], and extensive evaluations have been performed in different scenarios on physical servers of a Linux cluster (http://www.ubelix.unibe.ch). Evaluations have been performed on Intel Xeon E5-2665 and Intel Xeon E5-2650-v2 processors, i.e., each evaluation ran on a single core with 25GB of RAM and enough disk space to store the repofile. This scenario was select because the repository implementation was being evaluated as a single component for performance, but also because a single repository was enough to validate the impact of deletion operations in cache hit ratios and our modified caching policies.

7.3.1 Experimentation Scenarios and Parameters

Fig. 7.4 shows our evaluation topology. We evaluate the performance of an edge router, e.g., at an eNB, that continuously receives requests from the network according to YouTube and web server traffic models. The edge router is connected to a local repository, which is responsible for persistent caching. Independent of the network topology, an edge router has a *downstream face* from which file requests are received and content is returned and an *upstream face* where received Interests are forwarded and new content is received, i.e., file inclusions at the persistent cache of the edge router. The evaluation parameters are listed in Table 7.1.

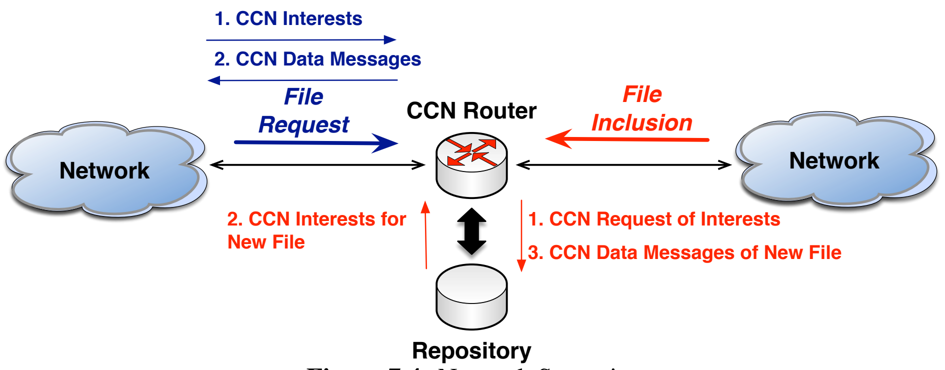


Figure 7.4: Network Scenario.

Parameter	**YouTube**	**Web server**
Requests	every 5s	
Request Popularity	Zipf distribution with $\alpha = 2$	$\alpha = 1$
File distribution per popularity class	Zipf distribution, $\alpha = 1$ mapped to inverse classes	
New Files	every 10s	
File sizes per popularity class	Gamma distribution, $\alpha = 1.8$, $\beta = 5500$ min. 500KB max. 100MB	Gamma distribution, $\alpha = 1.8$, $\beta = 1200$ min. 50KB max. 50MB
Repofile thresholds	2GB, 4GB, 8GB	8GB, 12GB, 16GB
Deletion ratios	50%, 25%	
Effective duration	86400s (1 day)	

Table 7.1: Evaluation parameters.

Similar to existing ICN literature [Rossi and Rossini, 2011], we assume that content popularity follows a Zipf distribution. We use 20 popularity classes and perform evaluations with parameters α set to 1 and 2. A parameter of $\alpha = 1$ is considered realistic for web server traffic and $\alpha = 2$ is used for YouTube traffic [Rossi and Rossini, 2011]. Several studies have shown [Cha et al., 2007; Yu et al., 2011; TubeMogul, 2009] that most files are unpopular and only a few files are very popular. Therefore, we map the number of files in all popularity classes to a Zipf distribution $\alpha = 1$ with inverted classes, i.e., most files are included in class 19 and fewest files in class 0.

The file sizes in each popularity class vary as well. Based on existing YouTube models [Abhari and Soraya, 2010], we set the file size distribution for our YouTube scenario to a gamma distribution with $\alpha = 1.8$ and $\beta = 5500$. Our YouTube files are between 500KB and 100MB, while most files are between 2 and 10MB (9.9MB mean). The file sizes for web server traffic are considerably smaller [Williams et al., 2005]. File sizes have increased in the last years and we believe that file sizes will increase even more in future information-centric networks. Transmitted ICN packets need to have a certain minimum size to be efficient, e.g., chunk size of 4096 bytes or more, to avoid too large overhead for content headers including names and signatures. Therefore, we believe that for future ICN traffic, many small files may be aggregated to larger data packets or ICN would only be applied to large static files, e.g., pictures or embedded videos, and not small text files that may change frequently. Therefore, we use a gamma distribution with $\alpha = 1.8$ and $\beta = 1200$ for the web server scenario. Our web server files are between 50KB and 50MB. However, most files are between 750KB and 1250KB (2MB mean).

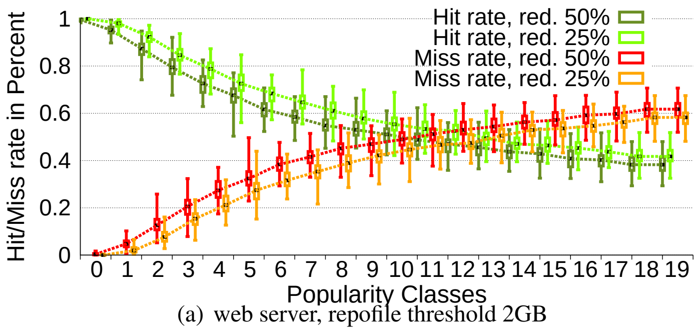
In our scenarios, requests are performed periodically, i.e., 1 new content request every 5 seconds. The requested content from the popularity class (Zipf distribution) is selected randomly among all created content objects in that popularity class. To simulate a dynamically growing file catalog and to evaluate the performance of persistent caching with regular deletion operations, we create and request new files every 10s. These files are included into the repository, i.e., file inclusions, as mentioned above.

For every scenario, we evaluate various *repofile thresholds* and *deletion ratios* of 50% (DR50) and 25% (DR25) of the *repofile*. We measure the performance of persistent caching during operation, i.e., the repository is filled initially with content and we collect statistics after a first deletion operation has been performed. The effective evaluation starts after the first deletion and lasts 86400 seconds (1 day). Thus, in one day we create approximately 85.54 GB of data in the YouTube scenario and 18.67GB of data in the web server scenario. Every configuration has been evaluated and repeated 100 times on the cluster servers that run a CCN router with persistent caching.

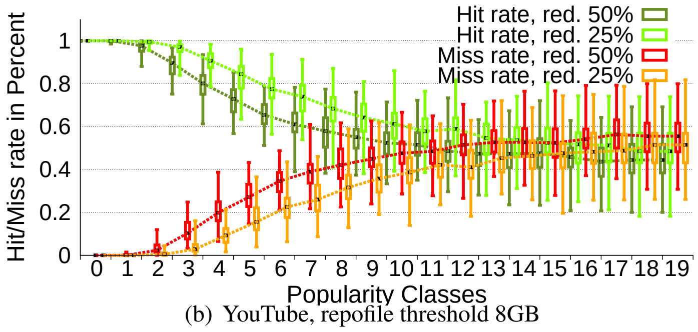
7.3.2 Cache Hit and Miss Rates

In this subsection, we evaluate the cache hit rates of our repository implementation in the YouTube and web server scenario. Fig. 7.5 and Fig. 7.6 show the cache hit and miss rates for all popularity classes in different configurations. The y-axis shows the hit/miss rates and the x-axis indicates the popularity class. The figures on the left side are obtained for our web server scenario, i.e., requests with Zipf distribution $\alpha = 1$, and the figures on the right side show the YouTube scenario with Zipf distribution $\alpha = 2$.

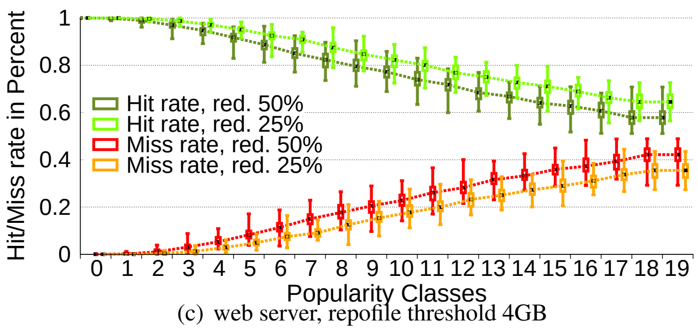
Fig. 7.5(a) shows the hit and miss rates in the web server scenario with a repofile threshold of 2GB. The dark green boxplots show the hit rates for a deletion ratio



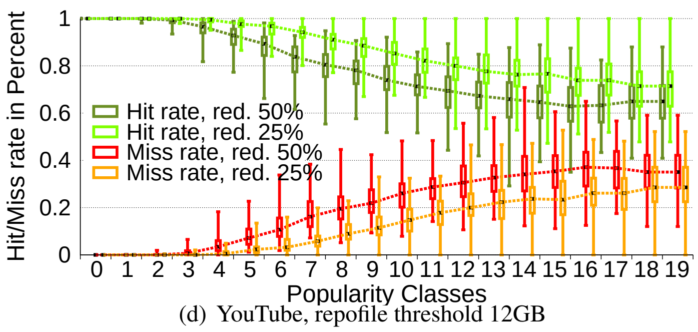
(a) web server, repofile threshold 2GB



(b) YouTube, repofile threshold 8GB

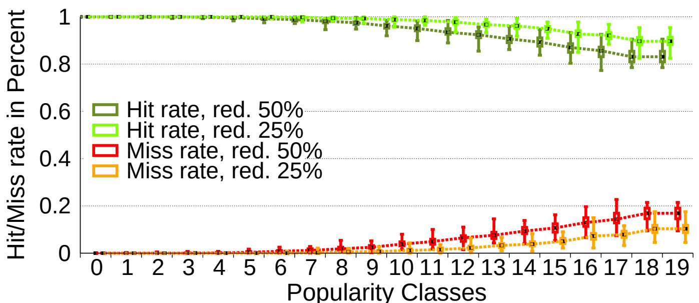


(c) web server, repofile threshold 4GB

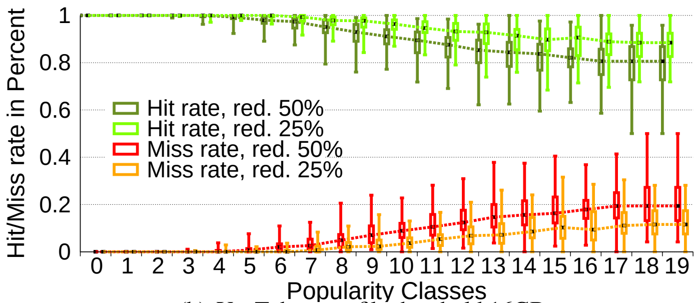


(d) YouTube, repofile threshold 12GB

Figure 7.5: Hit and Miss Rates for Web Server and YouTube Scenarios



(a) web server, repofile threshold 8GB



(b) YouTube, repofile threshold 16GB

Figure 7.6: Hit and Miss Rates for Web Server and YouTube Scenarios (cont.)

of 50% (DR50) and the light green boxplots for a deletion ratio of 25% (DR25). The overall hit rate of DR25 is slightly higher, i.e., 81%, compared to 77.5% with DR50. For high popularity classes, such as classes 0 and 1, the difference between DR50 and DR25 is smaller because files from these classes are barely deleted in both cases. However, for classes 3-17, the difference between DR25 and DR50 is larger by up to 6.3% because these files are kept more likely with DR25, while they are deleted with DR50. The red boxplots show the miss rates for DR50 and the orange boxplots for DR25. For DR50, hit rates are higher than miss rates up to files from popularity class 11, while for DR25, hit rates are better for more files, i.e., up to class 13. Even for the most unpopular content in class 19, DR50 results in a slightly higher miss rate of 61.5% compared to 58.1% with DR25. Therefore, freeing space too aggressively, i.e., DR50, has a noticeable impact on cache hit rates in the web server scenario.

Fig. 7.5(b) shows the hit and miss rates for our YouTube scenario with a repofile threshold of 8GB. Because file sizes are larger compared to the web server scenario, we use larger repofile thresholds for YouTube scenarios than for web server scenarios. With DR50, the overall hit rate is 95.3% and with DR25 it is 96.9%. The relative differences between DR50 and DR25 are smaller compared to the web server scenario. This is due to the fact that the probability for requests in most popular files in classes 0 and 1 are larger for a Zipf distribution with $\alpha = 2$ (YouTube) than $\alpha = 1$ (web server), i.e., 62.7% and 15.7% instead of 27.8% and

13.9%. Therefore, more than 78% of all requests in the YouTube scenarios request content from popularity classes 0 and 1. Since our implementation keeps the most popular files, there is no difference for DR50 and DR25 for most of the requests. However, for class numbers larger than 2, DR25 results in 4.9% to 12% higher hit rates than DR50. We notice a large variability in performance for popularity class numbers larger than 3 in Fig. 7.5(b). The variability is much larger than for the web server scenario in Fig. 7.5(a). This can be explained by two reasons. First, the request frequency of class numbers larger than 3 is higher with Zipf distribution $\alpha = 1$ compared to $\alpha = 2$ due to larger request probabilities. Second, the file ranges that we selected in the YouTube scenario are larger than for the web server scenario resulting in higher variability. When considering the average values, DR25 results in higher hit rates than miss rates up to popularity class 17, while for DR50 the miss rates become higher already at popularity class 11.

Fig. 7.5(c) and Fig. 7.5(d) show the hit and miss rates for the web server scenario with a repofile threshold of 4GB and the YouTube scenario with a repofile threshold of 12GB. Similarly as above, DR25 results in superior performance compared to DR50. In the web server scenario, DR25 results in an overall hit ratio of 93%, while for DR50 it is 90.7%. In the YouTube scenario, DR25 results in an overall hit rate of 99% and with DR50 it still reaches 98.1%. Although these values are much higher than in the web server scenario, miss rates for popularity classes numbers larger than 12 may become larger than the hit rates in the YouTube scenario (worst case) due to a large variability.

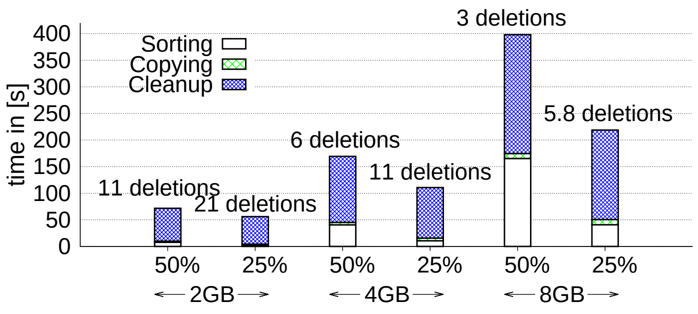
Fig. 7.6(a) and Fig. 7.6(b) show that if we further increase the repofile threshold to 8GB in the web server scenario and 16GB in the YouTube scenario, the average hit rates do not go below 80%. Even in the worst case in the YouTube scenario, the hit rates are always higher than miss rates.

7.3.3 Deletion Times

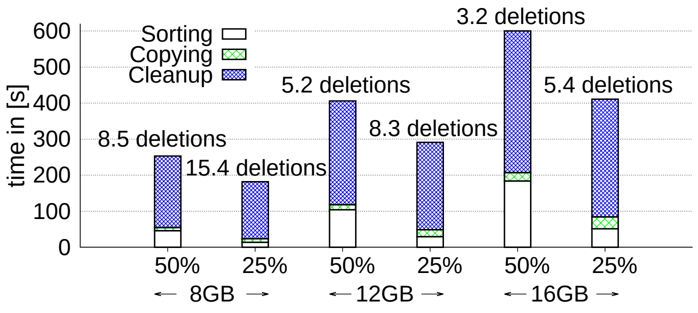
In this section, we evaluate the time durations to perform deletion operations in the web server and YouTube scenario.

Fig. 7.7 illustrates the overall times for deletions and the number of deletions in each evaluated scenario. The deletion time is split into subparts for sorting chunks of deleted content, copying files from the repofile and cleanup of the B-tree. The number on top of each bar denotes the average number of deletions in the corresponding configuration.

Fig. 7.7(a) illustrates the deletion times as well as the number of deletion operations (see Section 7.2.2) during 1 day in web server scenarios with repofile thresholds of 2GB, 4GB and 8GB. For every repofile threshold, the percentage on the x-axis denotes deletion ratios of 50% (DR50) and 25% (DR25). For small repofile thresholds of 2GB, deletions with DR25 take only 16.2% less time than for DR50. This is because most of the time is required for updating file references and deleting entries in the B-tree (cleanup). For DR50, B-tree cleanup requires 86%



(a) web server scenarios



(b) YouTube scenarios

Figure 7.7: Deletion times and number of deletions

of the total deletion time and for DR25, it requires even 92%. With increasing repofile threshold, the time for sorting increases. For a repofile threshold of 8GB and DR50, sorting is responsible for 41.5% of the deletion time and for DR25 it is a smaller fraction of 18.6%. DR50 requires more than 300% additional time for sorting and 24.8% more time for cleanup compared to DR25, while their difference for copying is insignificant. For one deletion operation with repofile threshold 8GB, DR25 requires 45% less time for the deletion than DR50. Considering that DR25 requires nearly twice as many deletions over one day, DR25 results only in 6% longer deletion times than DR50. However, due to the increased hit rate for DR25 (see Section 7.3.2), it may be worth investing 6% more time for deletion.

Fig. 7.7(b) shows deletion times and number of deletion operations for the YouTube scenario. A deletion operation with a repofile threshold of 8GB takes considerably less time than for the web server scenario: for DR50, it is 36.4% less time and for DR25 16.9% less time. Because files are larger in the YouTube scenario, i.e., have more chunks, fewer files are stored in the repository for the same repofile threshold. Due to the sequential request strategy in CCN, chunks of the same file are already (more or less) ordered. However, because popular files are continuously pushed down in the *delete_queue*, the sorting overhead increases with the number of deleted files, i.e., no First-In-First-Out (FIFO) deletion strategy. Fewer files that contain more ordered chunks (YouTube scenario) result in a less fragmented repository file than many files with fewer ordered chunks (web server scenario) and can, therefore, be sorted faster. As a result, sorting requires only 18.1% for DR50 and only 7.5% for DR25. However, the larger the repository file becomes, the higher is the overhead for sorting and cleanup. For a repofile threshold of 16GB, sorting is responsible for 30.6% of the deletion time for DR50 and 12.5% for DR25. Although sorting takes 3.5 times more time with DR50 compared to DR25, a deletion operation with DR25 only requires 31.6% less time compared to DR50. This is because the overhead from copying is not negligible anymore, i.e., 41.6% more time for DR25, and cleanup becomes more expensive for DR25, i.e., only 16.9% less time compared to DR50. When taking into account the number of deletions, DR25 results in 15.6% longer deletion times. Considering that the overall hit rate for DR50 and DR25 is almost the same (less than 1% difference), it may be a better strategy to use DR50 instead of DR25 in the YouTube scenario.

7.4 Summary

Persistent caching is required to support delay-tolerant networking or provide high content availability similar to CDNs. In this chapter, the repository implementation of CCNx was extended to support deletion and hence enable persistent caching at the edge of mobile networks. The followed approach for content deletion is based on a *delete_queue*, which keeps the most popular files at the tail of the queue. If disk space needs to be released, content can be removed from the head of the queue.

Extensive experimental evaluations were performed for different configurations in a web server and YouTube scenario. In every scenario, new content has been generated periodically such that deletion operations in the repository were necessary due to limited space. Evaluations have shown that this design can maintain high cache hit rates in both scenarios, but performance depends on the reserved repofile size for caching.

High cache hit rates at the edge are beneficial for both users and network operators. While network operators can reduce network traffic at the core network to improve network availability and reduce operational costs, users may benefit from faster content downloads (e.g. shorter delays, less round-trip time variability) as well as partial service and content availability if the core network is overloaded.

Nevertheless, more advanced strategies can be used at the edge to make the caching network converge quicker to its optimized state towards the requests of the users. These strategies, based on user mobility, request patterns, storage availability and other factors are described and evaluated in Chapter 8 below.

Chapter 8

Follow-Me Cloud Content Distribution Strategies

With a boom in the usage of mobile devices for traffic-heavy applications [Cisco Systems, 2015], mobile networks struggle to deliver good performance while saving resources to support more users and save on costs. Although edge caching strategies described in Chapter 7 already deliver very good performance when comparing to normal strategies and provide savings for mobile operators in terms of core consumed bandwidth, they are still far from being close to optimal in terms of efficiency and enhancement of user experienced performance. In fact, they only explore the proximity factor to keep popular content cached towards the edge and do not take into account other factors of the highly dynamic networks where they are used. Mobility, events, users' interests and other factors can be considered to go further beyond edge caching and thus optimize the way edge caches are used. In this chapter, enhanced follow-me cloud content distribution strategies are proposed for the preemptive migration of content stored in ICN caches at the edge of LTE mobile networks. With such strategies, the concept of content following the users interested in it becomes a reality and content within caches is more optimized towards the requests of nearby users. In Section 8.1 a brief motivation for this proposal is given. Section 8.2 describes its architecture and required changes to make it compatible with MCN and use ICNaaS (see Chapter 4). Section 8.3 describes mechanisms used for different decisions and requiring different decision techniques, score-based and distance-based, depending on the type of decision to make. Section 8.4 presents the evaluation of the proposed mechanisms for implementation purposes, in a testbed environment and also a simulated environment. Finally, Section 8.5 concludes the chapter and highlights the benefits of these contributions.

8.1 Motivation

Mobile networks evolution in the last few years has been quite intense, with major increase of throughput performance and resources usage efficiency. Such evolution is mostly driven by a tremendous demand for bandwidth at the edge to retrieve content, and mainly multimedia content such as photos, videos and audio. However, satisfying the requirements of the current number of users to deliver such massive amounts of content with very dynamic networks is still an open challenge, which is currently being addressed by a number of emerging concepts and technologies such as MEC [Ahmed and Ahmed, 2016] (see Section 2.4.2).

Using MEC as the starting point and leveraging the contributions from the previous chapters, the necessity for a model to optimize content delivery in mobile networks becomes a reality. Despite the demonstrated improvements in terms of performance and reduction of traffic at the core network by simply having edge caching strategies (see Chapter 7), such enhancement is not straightforward. Indeed, enhanced ICN caching mechanisms must efficiently populate caches and maintain content where it will yield the most benefit. Such benefit is intrinsically related with users' mobility patterns, as cached content should be available at users' new locations (e.g. associated to different LTE eNBs). This support requires preemptive actions (i.e. preventive measures to handle future requests) resulting, for instance, from mobility prediction algorithms [Li and Ascheid, 2012; Rajagopal et al., 2002; Chon et al., 2012], which may be employed for triggering the migration of content. Nevertheless, determining the optimal set and subset of content to be migrated, for a given amount of available resources (e.g. cache sizes, number of routers) and user mobility patterns, falls into an NP-complete optimization problem that has to be tackled. Possible solutions for this problem are introduced in the next section.

8.2 Architecture

In-line with the idea of FMC described in Section 2.5, the proposal for enhancing edge caches can be summarized into making decisions and perform preemptive migrations of mobile network's edge-cached content based on user mobility, i.e. migrations (copies) ahead of future user requests at a new location [Gomes et al., 2016a; Gomes et al., 2016b; Gomes et al., 2016c]. The following key objectives are assumed: mobility prediction must be used to take actions before users move from one location to another, content may only be migrated if it is likely that it will yield benefit at the destinations. Multiple destinations may be considered at the same time to improve accuracy and migration cost should be minimized as much as possible. This approach mainly differs from related FMC approaches by re-using the approach from Chapter 7 and allowing the migration of content at different cache levels in the hierarchy of LTE mobile networks, according to the proximity of users, additional metrics (e.g. popularity by groups, availability of resources, etc.) and higher granularity, fulfilling the goals of MEC.

We also assume that content migrations should happen among caches with modified policies based on popularity and similar meaningful metrics, as they typically implement a LRU policy that can delete recently added content in a matter of seconds if the load of received Interests is high. In this case we are interested in maintaining our own policy by re-using the one from Chapter 7, i.e. popularity based queue, with the most popular and recently accessed content at the head of the queue and deletions happening at the tail.

In order to achieve the previously mentioned goals, a generic architecture was defined as illustrated in Fig. 8.1.

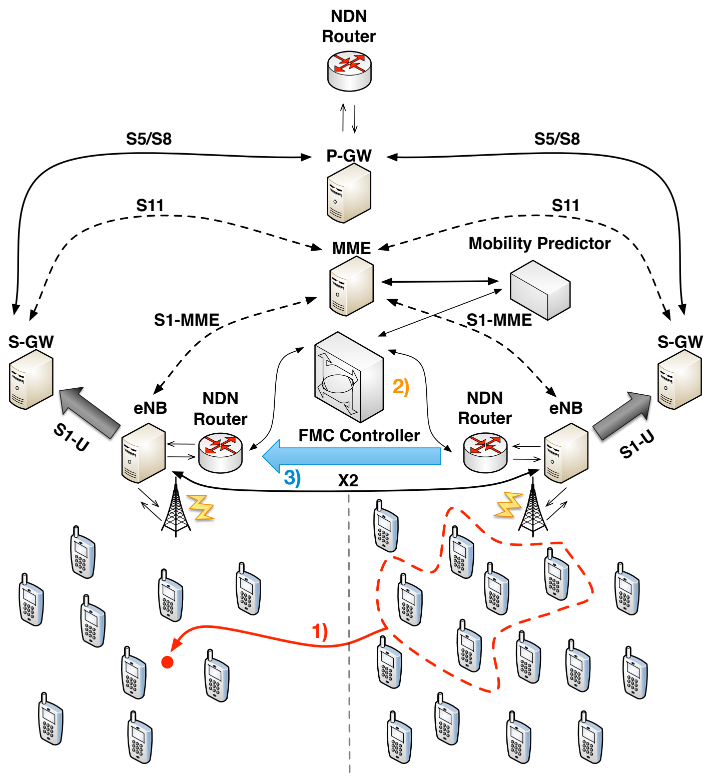


Figure 8.1: Follow-Me Cloud Architecture

In this architecture, we assume as base architecture the 3GPP EPS, namely the existence of its main components (eNB, S-GW, P-GW, MME) and its main interfaces (X2, S1-U, S1-MME, S11, S5/S8). At the same time, a recent and increasingly popular approach for ICN is selected - NDN (c.f. Section 2.2.8). It is selected as the base approach considering it brings advantages in terms of performance, but also due to its widely available simulation framework (ndnSim, see Section 3.3) that enables the required simulation experiments that validate the proposal to run.

We consider that NDN routers are co-located with LTE eNBs, serving the subset of users present at each network cell. At the same time, information about the network and its users is gathered at the FMC Controller. This information includes mobility prediction data, local content popularity and availability at a given cell, number of users per cell and availability of resources such as storage. With that information, a multi-step process for content migration is triggered every time mobility is predicted or detected by the Mobility Predictor using input from the LTE MME, based on the defined time period between predictions:

1. FMC Controller is notified about user mobility (e.g. user ID n is moving from cell ID x to cell ID y with probability z) and decides, upon policies such as the number of users moving to a destination cell, if other steps should be taken or the process should stop.

2. If decisions have to be made regarding content migration, the FMC Controller has to decide: where to migrate content (cell ID and corresponding NDN router), what subset of the local content should be migrated (content object's prefixes), when to do it (according to other scheduled migrations) and finally how to do it (routing and load balancing for content requests).

3. After a decision is made, the FMC Controller issues a NDN-standard message called Request of Interests, which instructs the destinations' NDN router(s) to fetch the subset of content to be migrated from the closest source and place it at its cache.

While the above aforementioned architecture is generic and can be applied to a number of environments, it requires small changes in its internals and components towards the integration in some scenarios. One of them is the MCN project, with its end-to-end cloud orchestration architecture. In this case, the base approach to consider for the adaption of this FMC architecture to be MCN-compatible is taken from Chapter 4, making usage of ICNaaS and introducing the possibility to be evaluated in the more realistic testbed environment described in Section 3.1.

One of the key benefits of having an MCN-compatible architecture is that it may interact with services that provide the virtualization of the entire mobile network's infrastructure (except the radio antennas). With that virtualization and subsequent cloudification of functionalities as services, many enhancements and new features can be explored at the edge of the mobile networks in a more realistic and non-simulated setting. Additionally, extra flexibility is attained by deploying resources on-demand where they are needed to fulfill the requests and adapting the number of resources to the load created by those requests. This requires integration with other MCN services, such as RANaaS, EPCaaS and MOBaaS (see Section 2.3.2).

Besides a simple integration with MCN services, to have the FMC architecture ready for MCN a small list of changes has to be considered:

- NDN Routers are replaced by CCN Routers, as ICNaaS was setup to use CCN Routers from its initial design in the project.
- The FMC Controller has exactly the same functionalities as before, but its entity gets renamed. It is named FMC Manager to comply with the naming scheme of ICNaaS' components.
- Mobility information is not retrieved from the MME, but rather from an external source: a mobility trace file obtained from existing mobility datasets. This

is due to the limitations in terms of number of cells for the MCN experimentation scenarios.

Also, the following implementation options (not visible in the architecture) are taken:

- RANaaS and EPCaaS are now used to support the whole 3GPP LTE architecture and have a running mobile network.
- The Mobility Predictor becomes MOBaaS, a service that actually implements mobility prediction algorithms and provides predictions to the FMC Manager.

As first described in Chapter 4, the support for the FMC architecture in MCN uses CCN instead of NDN and has two key components within ICNaaS: the CCN Server, which runs at every CCN router, and the FMC Manager, which is the same entity as the previously described FMC Controller. Both of them are orchestrated (i.e. deployed, provisioned, scaled and disposed) by the SO of ICNaaS.

8.3 Decision Mechanisms

As far as decision mechanisms are concerned, two types of decisions must be carefully analyzed and made in the proposed system: **where** to migrate content and **what** content to migrate. Other decisions (described below) may exist, but they only come as a complement and do not pose the same importance in the system. Also, the order for the decisions is rather important. While one may argue that, logically, first we ought to decide **what** content to migrate and only afterwards **where** to put it, a careful analysis leads exactly into the opposite. This is because the trigger for migrations is mobility or mobility prediction, and the output is a set of destinations. With these, the **where** has to be immediately selected. Moreover, the **what** decision, as it will be discussed below, involves comparing existing content objects at origin and target destination. So that this assessment can be performed, one target destination or a set of target destinations already needs to be decided.

Depending on the decision, different mechanisms can be used to handle it and better adapt to the underlying problem. These are described in the subsections below.

8.3.1 "Where" Decisions

As previously justified, the first decision is **where** to migrate content when users are moving. Although mobility prediction will output a list of candidate destinations and respective probabilities, such set of destinations may not be complete and still needs to be reduced and ranked according to other important factors.

To increase the list of candidates, one may assume that neighbor cells in-between the returned destination candidates and the origin should also be considered. After all, the user will need to travel through those cells and, depending on the delta time, i.e. amount of time in the future being predicted, it may even stop and stay there longer than at the final destination. With this full list of destination candidates, the problem is now how to rank these in order to select only a few that satisfy the defined criteria and will yield the highest trade-off between benefit and cost.

To handle decisions that involve ranking of candidates, MCDM algorithms such as the ones mentioned in Section 2.5 can be used. The most common methods are score methods where a score is given to each target destination and only the ones with the highest score are considered. Within these, perhaps the most well-known and used method is Analytical Hierarchy Process (AHP) [Saaty, 1980; Saaty, 1987; Saaty, 1990]. This method starts by summarizing the problem, deciding the hierarchical list of criteria to be considered for the decision and listing the alternatives to be ranked. In the case of the **where** decision, the problem is already defined: a destination or destinations need to be selected for content migration. Considering that not only the destination of users is important but also to maximize the efficiency of cache usage within the network edge, the following criteria were defined:

- Mobility prediction information containing probabilities for destinations.
- Percentage of non-intersecting content between origin and target destination.
- Percentage of free storage space at target destination.
- Relative size of mobile group, defined as the ratio between number of users moving and users present at target destination.
- Cost of migration, estimated as the network transfer delay to copy the expected data size from its origin to a target destination.

Afterwards, a $N \times M$ matrix is created, where N is the number of alternatives and M the number of criteria. For each cell of the matrix, a score value is calculated to reflect how good the alternative is in terms of the criteria being considered.

As each of the criteria may have a different significance for the decision, each of them should also have a weight value to be considered. In order to rank criteria, judgment is used by creating a $M \times M$ matrix where each criteria is compared against the others using pair-wise comparisons, i.e. each compared to all the others in terms of importance. For instance, we may define that mobility prediction information is three times more important than the relative size of the mobility group. In that case, the cell that compares mobility prediction with relative size of the mobility group will have a value of 3/1, and the opposite comparison the value of 1/3. This matrix, however, cannot be used directly. An eigenvector with the final weights has to be calculated following the procedure:

1. Convert fractions to decimals.
2. Square the resulting matrix.

3. Sum up the rows of the matrix and get a vector. Each of the rows of the vector must be divided by the sum of all its rows to normalize the values.

4. Repeat the previous steps until the resulting vector is not different from the previously obtained vector.

With the scores of each alternative for each criterion and the weights, the score of alternative i is given by:

$$S_i = \sum_{\substack{j=1 \\ \forall i \in [1,N]}}^{M} w_j r_{ij} \tag{8.1}$$

where:

S_i is the score of the i^{th} alternative;
r_{ij} is the normalized rating of the i^{th} alternative for the j^{th} criterion, which is calculated as $r_{ij} = x_{ij}/(\max_i x_{ij})$ for benefits and $r_{ij} = \frac{1}{x_{ij}}/(\max_i \frac{1}{x_{ij}})$ for costs;
x_{ij} is an element of the decision matrix, which represents the original value of the j^{th} criterion of the i^{th} alternative;
w_j is the weight of the j^{th} criterion;
M is the number of criteria;
N is the number of alternatives.

With the list of destination alternatives ranked and sorted in descending order by their score, hereafter just called "ranking", the decision about where to migrate content may be made based on the defined policies. For instance, the first three alternatives (destinations) of the ranking can be selected and content will be migrated to all of them. Or, depending on the problem and assuming that the score is normalized, alternatives with scores above 0.75 are to be selected.

8.3.2 "What" Decisions

After the decision about the target destinations, the decision of **what** content objects (complete files) should be migrated still remains. This decision has to be made considering that currently the association between LTE users and NDN users is not known, and therefore content cannot be related to a particular moving user. Therefore, it takes the local popularity of content as key criterion (together with its size) and considers the following steps. First, if the content object being considered is available nearby (1 hop distance) or already at the destination, it will not be migrated. Second, if it is not available, and if free space is available at the destination, content objects are just migrated until the cache is filled. Third, if no free space is available, both popular content at the origin and destination should be considered together and ranked to fill the destination cache with the content that will deliver the greatest benefit for all the users (existing and new ones).

The problem of filling up the target destination's cache with the closest to optimal set of content can be modeled as a Knapsack problem. This problem can be described as a combinatorial optimization problem: given a set of items, each with a certain weight and value, determine the subset of items so that the total weight is less than or equal to a given limit (cache storage space) and the total value is as large as possible.

To solve the Knapsack problem, Dynamic Programming [Andonov et al., 2000] can be used. However, it is a NP-complete problem and, even if a solution is found, it may take too long to calculate. Therefore, other algorithms can be used. MCDM algorithms (see Section 2.5) are usually adopted to reach near-optimal solutions for complex decisions problems such as this one. As described for the **where** problem, score-based algorithms could be used. These, however, will attempt to select the highest ranking content objects and fill up the caches without any combinatorial optimization. For combinatorial optimization such as the one of Knapsack, a different set of MDCM algorithms should be used - distance-based decision algorithms [Méndez et al., 2009; Chiou et al., 2009]. These do not simply calculate a score and rank it, they calculate the distance between each alternative and the ideal alternative, which is the best score in each criterion. Hence, for the **what** decision, they will deliver the best performance and go further beyond score-based algorithms.

Different algorithms based on distance could be considered, such as TOPSIS [Hwang et al., 1993], MeTHODICAL [Sousa et al., 2014] and DiA [Tran and Boukhatem, 2008]. Algorithm 1 details MeTHODICAL steps for B-benefits, K-costs and multiple criteria organized in a $M_{n,m}$ matrix for m-criteria and n alternatives. Step 2 allows the weighting of normalized $\widehat{\mathbf{B}}_{i,b} = \mathbf{b}_b \times \overline{\mathbf{B}_{i,b}}$ benefits and $\widehat{\mathbf{K}}_{i,c} = \mathbf{k}_c \times \overline{\mathbf{K}_{i,c}}$ costs, with $i = 1, 2, \cdots, n$, $b = 1, 2, \cdots, B$ and $c = 1, 2, \cdots, K$.

The difference among the MCDM algorithms is mostly related with the methods used to determine the optimal solution, which employ distance functions to determine the distance to ideal values, c.f. Algorithm 1 steps 5 and 6. TOPSIS employs the Euclidean distance, $D_i = \sqrt{Id_j - v_{i,j}}$, while DiA employs the Manhattan distance, $D_i = |Id_j - v_{i,j}|$, where v is the value of the solution being compared and Id is the value of the current near-optimal solution. The employment of such distances leads to non-optimal results due to the missing correlation between the values of the different criteria [Lahby et al., 2012]. The distance of MeTHODICAL, $A(\widehat{\mathbf{K}}_j) = m(\widehat{\mathbf{K}}_j) \pm v(\widehat{\mathbf{K}}_j)$, considers a range that is determined by the m-mean and v-variance functions, therefore supporting correlation. To determine the popularity of each content object k, the following equation was considered:

Algorithm 1 – MeTHODICAL optimization steps (as per [Sousa et al., 2014])

```
Require: $\sum_j^B \mathbf{b}_j = 1$ #Benefits weights vector
Require: $\sum_j^K \mathbf{k}_j = 1$ #Costs weights vector
Require: $\sum_i^m \sum_j^B \mathbf{B}_{i,j} \geq 0$ #Benefits matrix
Require: $\sum_i^m \sum_j^K \mathbf{K}_{i,j} \geq 0$ #Costs matrix
Require: $\mathbf{s}_{i,(t-1)} = 0$ #Initialize Score vector for $(t)ime - 1$
 1: $\overline{\mathbf{N}_{ij}} = \frac{\mathbf{M}_{i,j} - min(\mathbf{M}_{n,m})}{Max(\mathbf{M}_{n,m}) - min(\mathbf{M}_{n,m})}, i = 1, \cdots, n$ #Normalization
 2: $\widehat{\mathbf{G}}_{i,j} = \mathbf{n}_j \times \overline{\mathbf{N}_{ij}}$ with $i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, m$
 3: $I(\widehat{\mathbf{B}}_j) = max\{\widehat{\mathbf{B}}_{i,j} | i = 1, 2, \cdots, n\}$ #Ideal Benefits solution
 4: $I(\widehat{\mathbf{K}}_j) = min\{\widehat{\mathbf{K}}_{i,j} | i = 1, 2, \cdots, n\}$ #Ideal Costs solution
 5: $\Delta(\widehat{\mathbf{B}}_i) = \sum_{j=1}^{B} \left[ \frac{[I(\widehat{\mathbf{B}}_j) - \widehat{\mathbf{B}}_{i,j}]^2}{[I(\widehat{\mathbf{B}}_j) - A(\widehat{\mathbf{B}}_j)] + 0.01} \right] A(\widehat{\mathbf{B}}_j) = m(\widehat{\mathbf{B}}_j) + v(\widehat{\mathbf{B}}_j)$
 6: $\Delta(\widehat{\mathbf{K}}_i) = \sum_{j=1}^{K} \left[ \frac{[I(\widehat{\mathbf{K}}_j) - \widehat{\mathbf{K}}_{i,j}]^2}{[I(\widehat{\mathbf{K}}_j) - A(\widehat{\mathbf{K}}_j)] + 0.01} \right] A(\widehat{\mathbf{K}}_j) = m(\widehat{\mathbf{K}}_j) - v(\widehat{\mathbf{K}}_j)$
 7: $\mathbf{s}_i = \sqrt{\alpha \times \Delta(\widehat{\mathbf{B}}_i) + (1 - \alpha) \times \Delta(\widehat{\mathbf{K}}_i)},\ i = 1, 2, \cdots, n$
 8: $\mathbf{s}_{i,t} = \mathbf{s}_i + v\big(\mathbf{s}_i, \mathbf{s}_{i,(t-1)}\big),\ i = 1, \cdots, n$ #Set current score
 9: $\mathbf{r}_i = order\big(\mathbf{s}_{i,t}\big)$ #Vector in crescent order
```

$$P_k = \begin{cases} p_k * \dfrac{n_{mgt}}{n_{mgt} + n_{dst}}, \text{if the } k^{th} \text{content object is} \\ \text{considered for migration.} \\ \\ p_k * \dfrac{n_{dst}}{n_{mgt} + n_{dst}}, \text{otherwise.} \end{cases} \tag{8.2}$$

where:

P_k is the popularity of the k^{th} content object;
p_k is the local popularity of the k^{th} content object;
n_{mgt} is the number of users migrating from origin to the selected destination;
n_{dst} is the number of users at the selected destination for content.

8.3.3 Other Decisions

When content to be present at a given location is not already available locally, decisions are made towards deciding **how**, i.e. using which route, to copy it from its nearest replica and **when** that operation should be performed to avoid major load peaks in the network. The "how" decision is based on a simple load balancing strategy, considering the available links' status information and giving priority to direct links, e.g. 3GPP-defined X2 interfaces between eNBs. The "when" decision is derived from the available time for migration (given by mobility prediction) and the existing schedule for other migrations using the same components. Based on the time it will take to copy the content subset and the deadline to have it copied, a slot is picked in the migration schedule and the FMC Controller instructs the NDN router at the destination to fetch the content accordingly.

8.4 Evaluation

This section describes the evaluation methodology and scenarios we adopted to validate the architectures and mechanisms proposed in the previous sections.

8.4.1 Standalone Implementation Experiments

To decide which algorithms and which parameters to use for the decision of the subset of content to migrate, a first standalone evaluation was performed for the distance-based algorithms.

Experimentation Scenarios and Parameters

It includes two steps in different scenarios: a first step with a small testbed to fill databases and a second step at a Linux cluster to run large scale experiments in parallel.

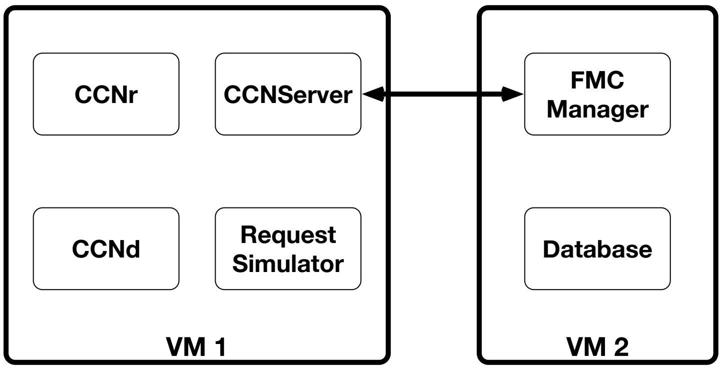


Figure 8.2: Evaluation testbed

In the first step, based on the MCN-compatible architecture, only the FMC Manager and the Request Simulator components are included, both implemented in the Python programming language, as depicted in Fig. 8.2. The Request Simulator component was configured to process the requests according to the configurations of each defined scenario. The CCN Server receives the commands to migrate content according to the decisions of optimization algorithms running in the FMC Manager. These components, as shown in Fig. 8.2, run in two VMs inter-connected with 10 Gigabits per second (10 Gbps) links, each with 2 vCPUs, 4GB of RAM and 40GB of disk space.

The MCDM algorithms validation compares Knapsack [Garey and Johnson, 1979], DiA [Tran and Boukhatem, 2008], TOPSIS [Hwang et al., 1993] and MeTHODICAL [Sousa et al., 2014] algorithms. Knapsack, as an optimization

Table 8.1: Parameters for Content and Requests

Parameter	**Normal**	**YouTube**	**Web-Server**
Request Popularity	Zipf Distribution $\alpha = 1$	$\alpha = 2$	$\alpha = 1$
Number of Popularity Classes	10	20	20
Content Object sizes per class	Normal $\mu = 30.60$ $\sigma^2 = 15.72$ min 150Kb max 70Mb	Gamma $\alpha = 1.8$ $\beta = 500$ min 500Kb max 100Mb	Gamma $\alpha = 1.8$ $\beta = 1200$ min 50Kb max 50Mb
Content Object distribution per class	Zipf Distribution with reversed classes $\alpha = 2$	$\alpha = 1$	$\alpha = 1$
Total number of content objects	2000	2000	2000

technique providing optimal solutions for NP-complete problems, has associated a fully polynomial time approximation scheme, which leads us to the usage of the UBELIX cluster for a timely determination of optimal solutions. Moreover, this validation did not consider mobility predictions. We assumed that users connected to one source cell move towards an arbitrary destination cell, as only the content subset migration is being analyzed (not location) when simulating file requests following the different distributions being evaluated.

Table 8.1 summarizes the three content production and request scenarios that are defined according to three typical usage profiles: Normal, YouTube and Web-Server.

Content popularity was defined according to the Zipf distribution [Rossi and Rossini, 2011], considering the characteristics of each scenario (e.g. number of files). The number of files present in each popularity class was determined by performing a reverse mapping of the Zipf distribution, as performed in the related work [TubeMogul, 2009; Yu et al., 2011]. Such reverse mapping is used because it has been demonstrated that the majority of the files are unpopular while only a few files are extremely popular. Finally, the total number of content objects and the content object size distribution per class follow the settings that are characteristic for each scenario. In the Normal scenario, the size of content objects follows a Normal Distribution, with mean 30.6MB and variance of 15.72MB. This distribution

Table 8.2: Configuration Criteria in MCDM Algorithms

Item	Criteria description
Benefits:	Popularity contents as per Eq. 8.2
Costs:	Size of Files
Cache Size:	256, 512MB, 1, 2 and 4GB

is based on a survey of current Internet statistics, such as the average size of a web page (2MB) [HTTP Archive, 2015] and the average size of 60 seconds YouTube videos with a resolution of 720p (40MB) [Google, 2015]. The YouTube scenario follows a model already defined in a previous work [Abhari and Soraya, 2010], using a gamma distribution with $\alpha = 1.8$ and $\beta = 500$. For the WebServer scenario we defined a model based on the observed growth of the size of files available at file servers [Williams et al., 2005], following a gamma distribution with $\alpha = 1.8$ and $\beta = 1200$.

Having content request scenarios defined, different parameters can be configured for the distance-based "what" decision mechanisms. MeTHODICAL enables weighting the distance of benefits criteria and the distance of costs criteria, both using α in Algorithm 1 step 7. In this evaluation, we considered $\alpha = 0.5$ for a balanced importance. In addition, step 8 was not considered since it aims to avoid fluctuations in handover decisions. It should also be noticed that both TOPSIS and DiA do not have such configuration parameters.

As depicted in Table 8.2, the MCDM algorithms have considered the popularity and the file size criteria metrics. In the initial steps of Algorithm 1, no weights have been considered, since the popularity is a composite criterion as it includes the relation between requests and popularity at source and destination cells. Since different cache sizes may produce different results, we consider five possible cache sizes: 256MB and 512MB, 1GB, 2GB and 4GB. The selection of these cache sizes is explained by the fact that we are considering the first level of caching (CS in CCN), which is stored in RAM or other similar high performance memory. While the usage of such type of caching significantly reduces access latencies, its size cannot be expanded arbitrarily to accommodate more files due to power and cost limitations [Perino and Varvello, 2011]. Therefore, a second level of caching is typically considered (hard-drive based, as the CCN repositories in Chapter 7) [Anastasiades et al., 2015] and our model can easily be extrapolated to both of these levels. However, as the selection of content stays unchanged by the type of memory in use, multiple levels of caching are out of scope in this evaluation.

To perform the first step of the validation, the small testbed depicted in Fig. 8.2 was run for several days to inject requests into the database of the FMC Manager for the files of each scenario, following the specified popularity distributions. With the database populated with the simulated requests of two different cells, the second step could start. Data was hence processed with multiple parallel jobs at the UBELIX cluster using Python and R [R Foundation, 2014] to apply the MCDM and knapsack algorithms.

This validation considers accuracy and efficiency. For assessing accuracy we consider (i) the number of files correctly selected for migration by the MCDM mechanism, when compared with the optimal solution provided by knapsack [Garey and Johnson, 1979] (the optimal set of files that can be moved given cache constraints); and (ii) the accumulated ratio of content requests corresponding to cache hits. 100% indicates that all content requests were available on cache and 0% indicates that all content requests were not available in the cache. Efficiency is assessed in terms of the processing time required to determine the content to be migrated.

Regarding the evaluation of file selection correctness, we considered two distinct metrics. The first is the percentage of files that each algorithm correctly identifies as migration candidates, when compared to the reference results produced by knapsack (compared to the number of files). The second metric is the relative proportion of "correct" content (to be requested by users at new location) included in the migration (e.g. a 65% value means 35% of the migrated content was wrongly selected for migration and 65% correctly selected).

Results

Table 8.3 presents the results obtained with the Knapsack algorithm for each scenario and for each cache size considered, depicting the optimal solution determined by Knapsack in terms of number of content objects and ratios of content requests corresponding to cache hits. These results highlight the impact of the popularity distribution in the different scenarios. For instance, 47 files in the YouTube scenario account for ≈71% of the requests received, while for an equivalent value in the WebServer scenario 366 files are required. Recall that the main difference between these scenarios relies on the size of the content objects per class.

It should be noticed that the Knapsack algorithm takes the cache size as a parameter, and determines a different set of files for each cache size. The MCDM algorithms do not take the cache size as a parameter, and being deterministic, the solution for the different cache sizes is always the same. After obtaining the ordered set of files from the MCDM algorithms, the files to have in cache are selected until the cache size limit is reached. By taking into consideration the results obtained, it is possible to see that the results from the MeTHODICAL algorithm are very close to the specific results of the Knapsack algorithm for each cache size.

For the results that are not deterministic, the confidence interval is presented over the average and shows what results can be expected from further repetitions. It was determined with a confidence level of 95% and allows us to not only show the consistency of the good results while using MeTHODICAL, but also that the performance of both DiA and TOPSIS is poor across the Normal and YouTube

Table 8.3: Knapsack Results per Scenario

Cache Size	**Normal**		**YouTube**		**WebServer**	
	Content obj.	Cache Hits (%)	Content obj.	Cache Hits (%)	Content obj.	Cache Hits (%)
256MB	15	29.71	47	70.69	233	63.06
512MB	29	43.10	83	82.07	366	75.74
1GB	47	56.31	160	90.50	648	87.75
2GB	78	69.67	300	95.28	1202	96.32
4GB	156	80.60	558	98.12	1980	99.95

scenarios, and highly variable in the WebServer scenario.

Fig. 8.3 shows the ratio of files correctly selected by the different MCDM algorithms, compared to the optimal solution determined by the Knapsack algorithm for each scenario. MeTHODICAL, for instance, selected around 90% of the files identified as the optimal solution for the Normal scenario (i.e. determined by knapsack), while DiA and TOPSIS only selected around 5% and 10%, respectively.

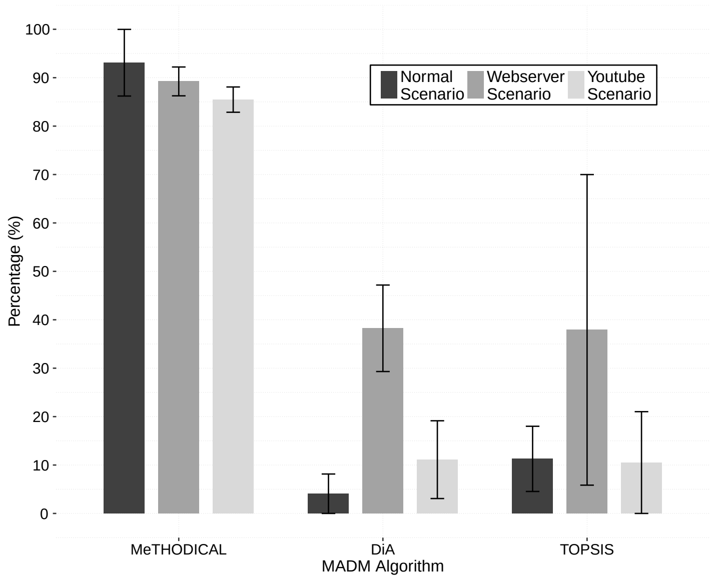


Figure 8.3: Number of Files Correctly Selected for Migration

Fig. 8.4 illustrates the relative volume of content correctly selected for migration, compared to the total volume of content actually migrated (including files which were wrongly selected for migration). The higher these values are, the more accurate the results provided by the MCDM algorithm will be.

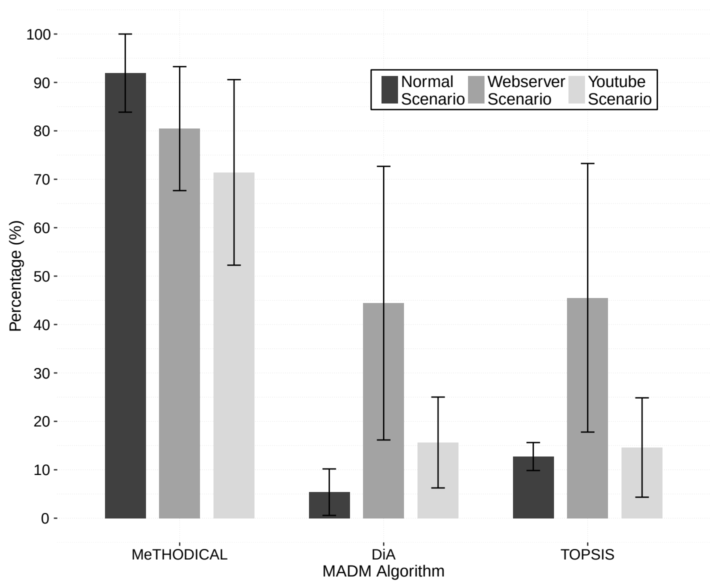


Figure 8.4: Volume of Content Correctly Selected for Migration

Considering the depicted results shown in Fig. 8.3 and Fig. 8.4, it is clear that MeTHODICAL is the best performing algorithm independently of the cache size. On the other hand, the performance of DiA and TOPSIS is not consistent and is impacted with the cache size of routers, as they present a high standard deviation. The low performance of DiA and TOPSIS is due to the fact that both of these techniques do not correlate values to determine optimal solutions. MeTHODICAL supports correlation of values through the distance function, by correlating the values through the mean and variance functions, as depicted in Algorithm 1 in steps 5 and 6.

Next, we look at the cache hits in the form of content requests percentage that is served from the cache of routers. Cache hits are considered for the various algorithms, scenarios and cache sizes. In the Normal scenario (c.f. Table 8.1), MeTHODICAL produces results almost as good as knapsack, as it can be observed in Fig. 8.5. As the cache size increases, the percentage of cache-hits also increases, since the cache can accommodate more files. Both MeTHODICAL and Knapsack achieve cache hit ratios of around 80% with 4GB caches. On the other hand, both DiA and TOPSIS perform unsatisfactorily, with ratios below 10%. In the Normal scenario the number of files to migrate is lower and files are larger (c.f. Tables 8.1 and 8.3).

Results observed for the YouTube scenario follow the same trend (c.f. Fig. 8.6), with Knapsack and MeTHODICAL achieving top cache hit ratios around 90%. The increased performance is explained by the characteristics of the files in the

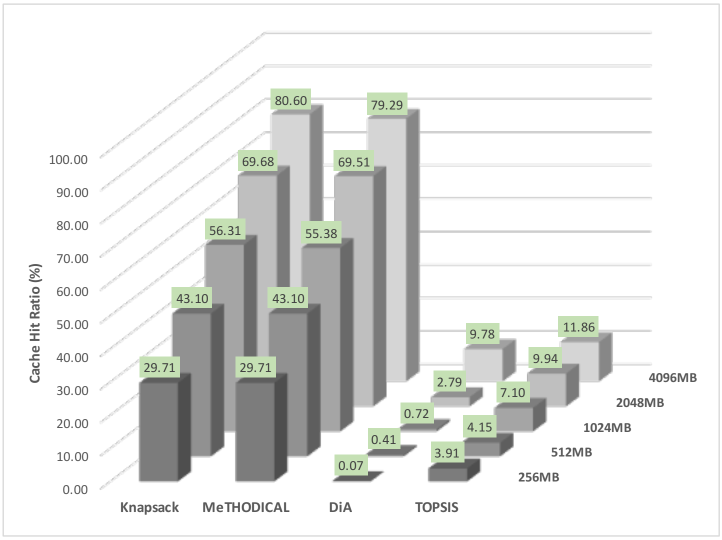


Figure 8.5: Percentage of cache-hits in the Normal Scenario

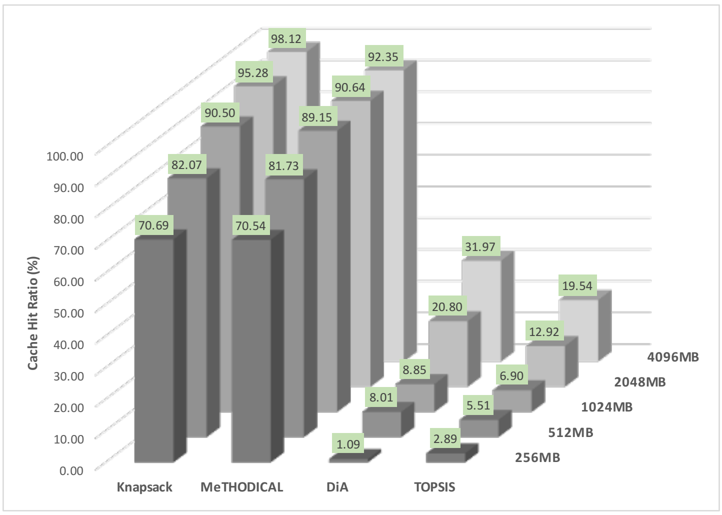


Figure 8.6: Percentage of cache-hits in the YouTube Scenario

YouTube scenario, which are smaller than in the Normal scenario. It should be noted that DiA and TOPSIS also improve their performance, mainly due to the increased number of cached files.

In the WebServer scenario many more files can be cached with a cache size of 4 GB, as depicted in Fig. 8.7. This fact explains the cache hit ratios around 99% for Knapsack and MeTHODICAL and 96% for DiA and TOPSIS. Indeed, with 2 GB cache size the cache hit ratio for DiA and TOPSIS also increases to values of around 50%. The size of files in this scenario is the lowest leading to the case where the cache can accommodate almost all the files.

Experiments for assessing accuracy suggest the selection of MeTHODICAL and

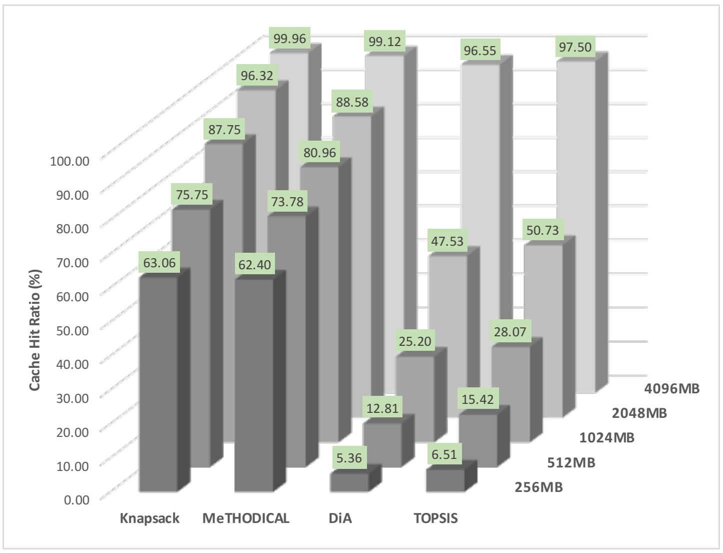


Figure 8.7: Percentage of cache-hits in the WebServer Scenario

Table 8.4: Processing Time (s) of FMC Decision Algorithms in the Normal Scenario

Algorithm / Cache Size	256MB	512MB	1GB	2GB	4GB
Knapsack	612.8	1398.9	2845.6	5705.5	13094.2
MeTHOD.	0.003	0.003	0.003	0.003	0.003
DiA	0.102	0.102	0.102	0.102	0.102
TOPSIS	0.104	0.104	0.104	0.104	0.104

Knapsack algorithms to solve the NP-complete problem of content migration in the FMC model, as these algorithms consistently produce the best results for all scenarios, also being less affected by cache size, file size or number of files. However, as discussed next, it is still necessary to determine the computational costs of both options to assess their practical viability.

As for efficiency, results are presented in the next paragraphs. Table 8.4 presents the processing time for the different algorithms in the Normal scenario. The processing time of the Knapsack algorithm increases with the size of the cache, which is not the case of MeTHODICAL (MeTH) and related algorithms. The same behavior can be observed in the YouTube and WebServer scenarios regarding the processing time, as depicted in Table 8.5 and Table 8.6, respectively.

The reason for the fact that MCDM algorithms do not present different processing

Table 8.5: Processing Time (s) of FMC Decision Algorithms in the YouTube Scenario

Algorithm / Cache Size	256MB	512MB	1GB	2GB	4GB
Knapsack	518.4	1054.6	2124.9	4262.1	8547.5
MeTHOD.	0.023	0.023	0.023	0.023	0.023
DiA	0.075	0.075	0.075	0.075	0.075
TOPSIS	0.075	0.075	0.075	0.075	0.075

Table 8.6: Processing Time (s) of FMC Decision Algorithms in the WebServer Scenario

Algorithm / Cache Size	256MB	512MB	1GB	2GB	4GB
Knapsack	645.2	1293.8	2590.4	6371.2	12004.5
MeTHOD.	0.031	0.031	0.031	0.031	0.031
DiA	0.096	0.096	0.096	0.096	0.096
TOPSIS	0.094	0.094	0.094	0.094	0.094

times for each cache size is that they determine the same optimal solution for each cache size. The difference only relies on the number of files that can be selected for migration.

Hence, these algorithms are deterministic. As long as the input is the same, they always output the same result. In this case, they output an ordered list of content objects to be moved from a given origin to a target destination calculated by the algorithms used for the **where** decision, being the first element the most important to be moved, i.e. the one that should be migrated first. Later on, the list is iterated to add the files that can be moved until the cache size limit is reached.

As stated before, the FMC model, which operates in the same fashion as a real-time application, requires high-efficiency in terms of performance (not using unnecessary resources) and processing time. The results depicted so far demonstrate that MeTHODICAL is the most accurate and efficient MCDM algorithm, fulfilling the performance requirements of the FMC model.

It is also demonstrated that in-network caching with cache sizes as small as 512MB can provide improvements for users and operators, since it becomes possible to achieve cache-hit ratios around 50%, which meets the requirements of high performance caching hardware implementing first-level of caching mechanisms.

8.4.2 Testbed Experiments

In order to evaluate the proposed mechanisms in a more realistic environment (testbed described in Section 3.1), the FMC architecture described in Section 8.2 with the list of changes that make it MCN-compatible was considered. It comprises FMC specific components and a deployment of ICNaaS is performed to evaluate the end-to-end integrated model from the users experience perspective in the context of the MCN project.

Experimentation Scenario

To enable the full context of MCN, ICNaaS, MaaS and MOBaaS were deployed as a service combination, i.e. simultaneously deployed and aware of each other. Due to existing constraints the EPCaaS data containing user mobility information was received from a mobility trace (described below), while RANaaS was left out to simplify the setup and allow a higher number of cells. This deployment's setup is shown in Fig. 8.8 at the MCN end-to-end orchestration framework's level, depicting AAAaaS authentication where applicable.

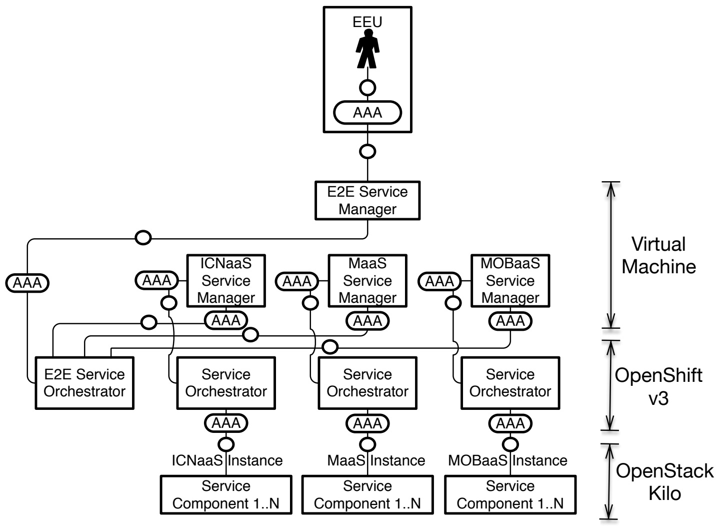


Figure 8.8: End-to-End Orchestration

The first step for deployment is the authenticated request from an EEU to the End-to-End SM. This component creates an End-to-End SO to manage this deployment. After the E2E SO has been created, it uses a manifest file to list the services to be deployed and their dependencies. Then, endpoints are queried from a catalog and services are requested by contacting their SMs, which have the descriptions of their services and information on how to create their SOs. Hence, these SMs create SOs at OpenShift, which will instantiate the required components at OpenStack Kilo [OpenStack Foundation, 2016]. When all the components have been deployed, endpoints of dependencies are provided to the dependent services, e.g. ICNaaS requires MOBaaS and needs its endpoint. That triggers the provisioning phase, which concludes the deployment and provides information to the EEU that the E2E deployment is ready to be used.

As for the components deployed by the orchestration framework at OpenStack Kilo for the required services, they are summarized in the following list (vCPU is a virtual CPU unit, mapped to a thread/core by the hypervisor:

- One ICN Manager with 1 vCPU, 2GB of RAM and 20GB of disk space.
- Five CCN Routers, each with 2 vCPUs, 4GB of RAM and 40GB of disk space. Four are assigned to a given LTE cell ID, while the other is in a different layer (at EPC P-GW level). Hence, it is a 1-4 tree topology.
- One FMC Manager with 2 vCPUs, 4GB of RAM and 40GB of disk space.
- One MP Middleware with 1 vCPU, 2GB of RAM and 20GB of disk space.
- One MOBaaS Predictor with 2 vCPUs, 4GB of RAM and 40GB of disk space.
- One MaaS Zabbix with 1 vCPUs, 2GB of RAM and 20GB of disk space.

Regarding the ICNaaS clients (end users), they were deployed manually as CCN Routers without repositories. After setting up content prefixes' routes, network connections at 100 Megabits per second (100 Mbps) to approximate LTE speeds and content request generators, they were ready to start the proposed experiments.

Mobility prediction dataset

In order to make location prediction, historical user traces should be provided to MOBaaS. However, due to the fact that in this prototype implementation actual user traces are not available, MOBaaS includes a mobility data trace provided by Nokia for academic research (collected in a different initiative from the one considered in the simulation environment below). The dataset was collected during the Nokia Mobile Data Challenge [Laurila et al., 2012], which is a large-scale research initiative aimed at generating innovations around smartphone-based research, as well as community-based evaluation of related mobile data analytics methodologies. This dataset includes rich context information running at the mobile phone for around 200 users for 2 years, of which MOBaaS could only select 100 ranging from 2-6 months due to trace quality issues in regard to the useful location information.

Evaluation metrics

This end-to-end users experience validation attempts to assess the experience perceived by end users with and without FMC (edge caching only), thus depicting the benefits that can be obtained when using this model. It consists of two different metrics: 1.) average download time of a file within a certain group size (determined according to the settings in the YouTube and WebServer scenarios, the same as used in simulation and described in subsection 8.4.3) and 2.) users' satisfaction. While the first metric is straightforward, the second requires further explanation. Users' satisfaction can be evaluated in multiple different ways and with numerous methods, and it is often very subjective depending on the users themselves. As we do not have real users in our evaluation or even content transfer types that can easily match common Quality of Experience (QoE) metrics, we propose that a Sigmoid function [Seggern, 2006] is used to approximate the users' satisfaction with respect to perceived QoS [Stamoulis et al., 1999; Pal et al., 2005]. Therefore, the users' satisfaction can be given by:

$$U(x) = \frac{1}{1 + e^{-\alpha(x-\beta)}} \tag{8.3}$$

where $U(X) \in [0, 1]$ is the satisfaction factor, x is the variable representing calculated download bandwidth in Mbps and α and β are constants that need to be defined to set the steepness and center of the curve.

For the definition of α and β, common sense from a user's perspective was the strategy. Starting with β, the center of the curve corresponds to a satisfaction factor of 0.5. We can assume that the user will have an average satisfaction if the bandwidth is more or less what it is expecting to be from its daily usage. Therefore, it makes sense that an average download bandwidth value is considered for β. From Akamai's latest report on the state of the Internet [Belson, 2016], a value of 5.6 Mbps is assumed to be the average download bandwidth in the Internet. Thus, $\beta = 5.6$. As for α, the steepness of the curve, we decided to use a value of $\alpha = 0.4$ because 0.9 satisfaction is achieved when download bandwidth is around twice the average and 0.99 satisfaction is achieved when download bandwidth is around triple the average. The resulting function is depicted below in Fig. 8.9.

Results

The end-to-end user experience evaluation includes download times and users' satisfaction in the WebServer and YouTube scenarios, as depicted in Fig. 8.10 and Fig. 8.11. With the number of files being considered (2000) and their respective size depending on the distribution associated with the specific scenario, files are grouped according to their size, where 1MB includes files with a size below 1.5MB, while 2MB includes files with size in the range $[1.5, 2.5[$, etc. In addition, download time is measured for each file, with and without FMC (edge caching only). This value is also used to calculate bandwidth in Mbps and thus the satisfaction

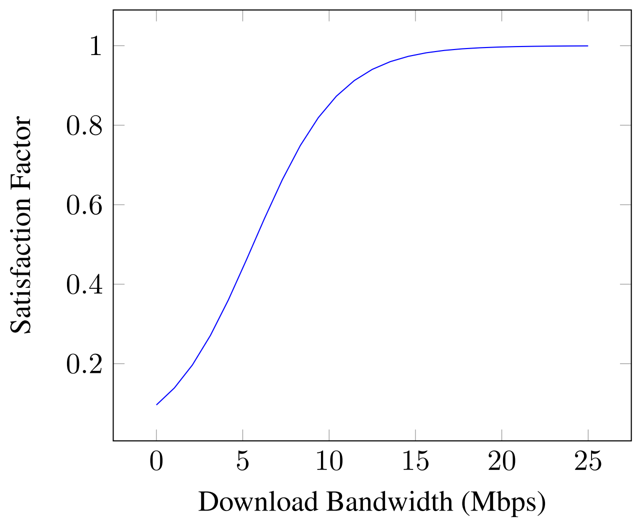


Figure 8.9: User's Satisfaction Plot

factor. The aim of such evaluation is to demonstrate the performance benefits of the FMC model and associated services perceived by end users.

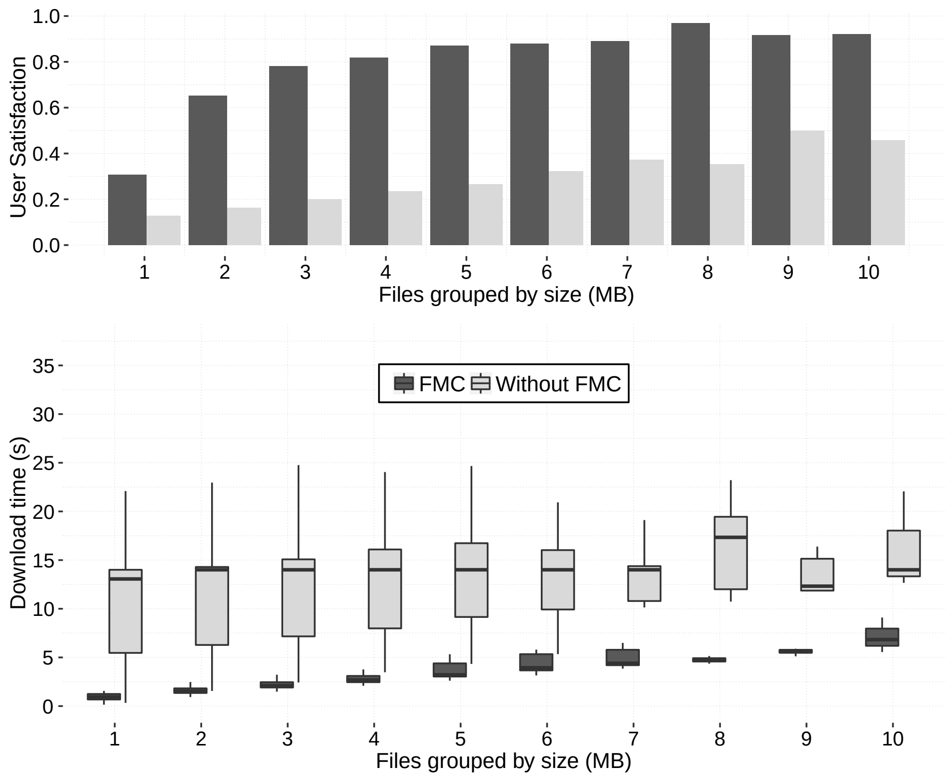


Figure 8.10: Download Time in seconds for the WebServer Scenario

Independently of the size of files, download times are higher when FMC is not employed. Indeed, FMC is able to reduce the download times by a factor of 2-5 (as perceived by end users). It is also noticed that FMC is able to provide consistent results for files with approximately the same size. The download time increases in a linear fashion as the size of grouped files increases. This is an expected result, as bigger files required more time to download. The variation in the case without FMC leads to dissatisfaction of users, who may experience inconsistent download times. For instance, for a file with a size below 1MB, it can take a minimum of 6 and a maximum of 13 seconds. This is further validated by the users' satisfaction analysis. Even with small file sizes that have greater overhead in ICN transfers, satisfaction with FMC is much higher and in fact a size of 5MB is already enough to obtain a satisfaction factor close to 0.9. It also shows that users' satisfaction is much more consistent with FMC, and hence that users will have a greater experience overall.

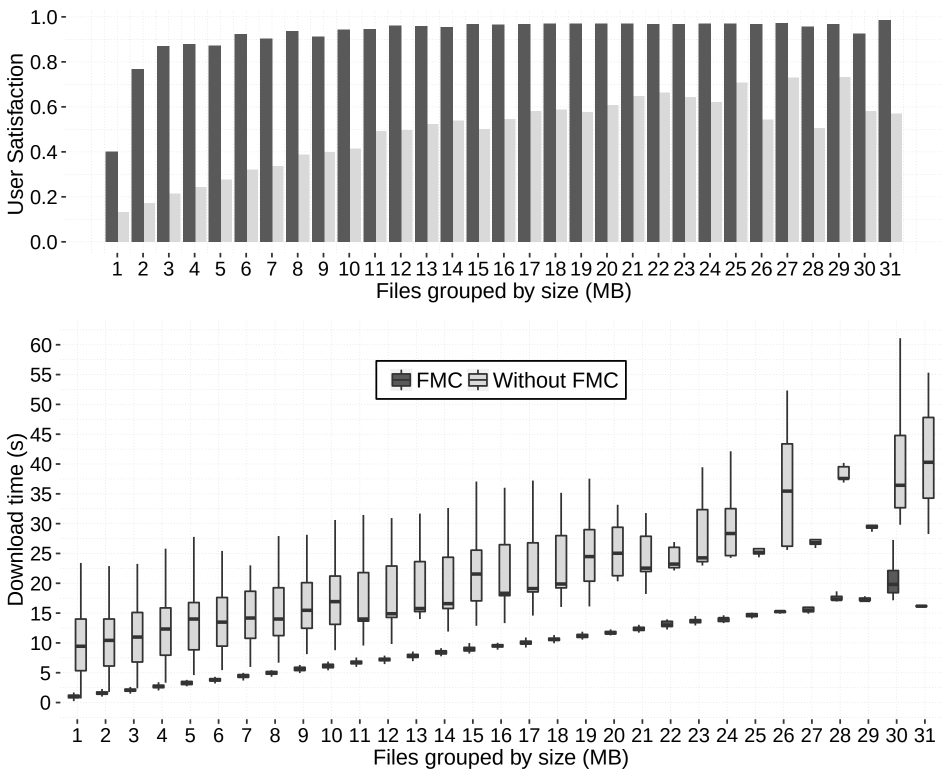


Figure 8.11: Download Time in seconds for the YouTube Scenario

The YouTube scenario is characterized by files with bigger sizes, i.e. sizes above 30MB. In the plot of Fig. 8.11 they are grouped as 31MB, which considers file sizes in the range $[30.5, +\infty[$. In this scenario the difference between the file size and the performance achieved with FMC is more evident, in comparison to the WebServer scenario. The download time increases linearly, for the cases with and without FMC (edge caching only), but has high variations when FMC is not employed. FMC is able to support stable download times irrespective of the file sizes. In contrast, higher variations and inconsistencies are observed with bigger file sizes in the case where FMC is not employed. As in the WebServer case, this behavior leads to the dissatisfaction of users as the downloads vary a lot in time, without considering the impact that content requests may have in the core network. Moreover, we can observe again that users' satisfaction with FMC is very consistent and much higher, and that with a small size of 3MB the satisfaction is already achieving values close to 0.9. In fact, even when file sizes are much higher (30MB) the satisfaction with FMC is about 50% higher than without it.

Considering the scenarios for which the FMC model is intended, the perceived experience for users that follow popularity trends is improved, decreasing latency when obtaining popular files while not decreasing the perceived experience of other users. In addition, the FMC model also enhances core networks by introducing bandwidth savings and allowing a better usage of resources, i.e. cache of edge routers.

8.4.3 Simulation Experiments

In order to evaluate the proposed strategies with the mechanisms introduced in Section 8.3, a set of simulation experiments was defined and is described in detail in the next subsections.

Mobility Data Input

A realistic mobility trace was selected [Eagle and (Sandy) Pentland, 2006]. This trace includes data from one hundred human subjects over the course of nine months, and it was collected by MIT students using Nokia 6600 smart phones in the academic year of 2004/2005. Although the information collected includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status, for this evaluation only the information on mobility was considered: Object Identifier (OID), endtime, starttime, person_OID, celltower_OID. Therefore, it is possible to know at every given time to which cell a given person is connected. Such trace can thus be used to assess how the system behaves in realistic conditions, as if mobility was generated in any other way (e.g. mobility model), it would probably create biased results and render the conclusions invalid for real-world scenarios.

Concerning mobility prediction, it is not the main focus of this work and there was no implemented framework to be used in simulation. Therefore, 15 minutes delta time predictions (15 minutes in the future) were generated at every 1 hour of simulation time according to the results obtained by mobility prediction works [Samaan and Karmouch, 2005; Amirrudin et al., 2013]. It is still useful and worth to consider such long time frames for content transmission as the prediction also takes time, and results such as the one from MOBaaS [Karimzadeh et al., 2015] demonstrate that this time becomes long when there are many predictions for many users. As concluded in the mentioned works, a 50% user movement randomness corresponds to an accuracy of about 50%. Thus, the generated predictions for these experiments had an accuracy following a normal distribution $N(0.5, 0.1)$.

Basic Setup

The setup for this evaluation is depicted in Fig. 8.12. It consists of the proposed architecture implemented in the ns3 simulator using its LTE framework together with ndnSIM 2.1 (c.f. Section 3.3). First, the simulator creates a Content Producer attached to a NDN Router, which is a node with NDN capabilities such as caching and forwarding. The latter is by itself attached using IP and 10 Gbps links to the EPC of the LTE module. Afterwards, a pair of eNB + NDN Router (including a NDN Content Store, i.e. cache of 2 GB stored in RAM) is created for each cell of the trace mobility file, attaching randomly positioned (within the cell's coverage) UEs + NDN Consumers to the LTE network according to the trace mobility inputs. These attachments are changed over the simulation time, thus emulating user mobility and triggering an handover using the X2 interface. That handover is managed by the MME, which is modified to feed information to the Mobility Predictor. The Mobility Predictor feeds mobility information, while NDN Routers provide the remaining relevant information (criteria) to the FMC Controller, which makes decisions and therefore instructs content to be copied between NDN Routers.

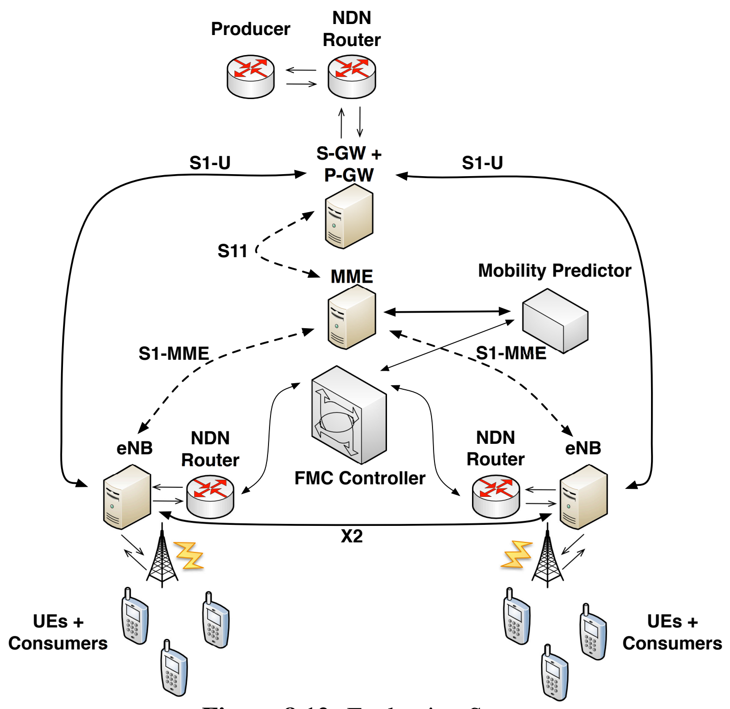


Figure 8.12: Evaluation Setup

As for the simulation itself, it runs for 12 hours in a daytime period of the trace mobility file, when it is more likely for users to be active. Simulations were repeated 30 times using different day periods with at least 60 active users (limitation imposed by size and quality of available traces) and considering the 100 most visited cells, using a Linux cluster to parallelize the work (http://www.ubelix.unibe.ch). Additionally, for parameters not mentioned here, the default values were used (e.g. LTE radio parameters).

Where Decision Criteria's Weights

Assuming that different weights for the criteria used in score-based algorithms for where decisions may return different results, four different weight sets were considered for evaluation. These are highlighted in the tables below, which each contain the AHP judgment matrix for all the criteria and the resulting eigenvector with weights calculated using the process described in Section 8.3.

(a) Matrix

	M.P.	**Diff**	**F.S.**	**G.S.**
M.P.	1/1	2/1	4/1	3/1
N.C.	1/2	1/1	2/1	3/1
F.S.	1/4	1/2	1/1	1/2
G.S.	1/3	1/3	2/1	1/1

(b) Vector

	Weight
M.P.	0.46124
N.C.	0.28450
F.S.	0.10633
G.S.	0.14793

Table 8.7: Weight Set #1

(a) Matrix

	M.P.	**Diff**	**F.S.**	**G.S.**	**Cost**
M.P.	1/1	2/1	4/1	3/1	3/1
N.C.	1/2	1/1	2/1	3/1	2/1
F.S.	1/4	1/2	1/1	1/2	1/1
G.S.	1/3	1/3	2/1	1/1	2/1
Cost	1/3	1/2	1/1	1/2	1/1

(b) Vector

	Weight
M.P.	0.39778
N.C.	0.25232
F.S.	0.09725
G.S.	0.14897
Cost	0.10368

Table 8.8: Weight Set #1 with Cost

(a) Matrix

	M.P.	**Diff**	**F.S.**	**G.S.**
M.P.	1/1	1/2	3/1	2/1
N.C.	2/1	1/1	3/1	4/1
F.S.	1/3	1/3	1/1	2/1
G.S.	1/2	1/4	1/2	1/1

(b) Vector

	Weight
M.P.	0.28450
N.C.	0.46124
F.S.	0.14793
G.S.	0.10633

Table 8.9: Weight Set #2

(a) Matrix

	M.P.	Diff	F.S.	G.S.	Cost
M.P.	1/1	1/2	3/1	2/1	2/1
N.C.	2/1	1/1	3/1	4/1	3/1
F.S.	1/3	1/3	1/1	2/1	1/1
G.S.	1/2	1/4	1/2	1/1	2/1
Cost	1/2	1/3	1/1	1/2	1/1

(b) Vector

	Weight
M.P.	0.31739
N.C.	0.36398
F.S.	0.11526
G.S.	0.10714
Cost	0.09623

Table 8.10: Weight Set #2 with Cost

In Table 8.7, the importance given to mobility prediction (M.P.) is higher than for any other criterion. At the same time, group size (G.S.) is considered more important than free space (F.S.) and cost is not considered.

In Table 8.8, everything is similar to Table 8.7 besides the fact that migration cost is now considered and the resulting weights are different.

In Table 8.9, the greatest importance is given to the amount of non-intersecting content between the caches (N.C.) and, unlike in Table 8.7, F.S. is considered more important than G.S. Also here, cost is not considered.

In Table 8.10, everything is similar to Table 8.9 besides the fact that migration cost is now considered and the resulting weights are different.

Content and Requests

The Content Producer consists of a file generator, which generates 100 000 files according to the defined scenario: either a YouTube scenario or a web server scenario. The first scenario intends to mimic video streaming traffic using conditions from the well-known YouTube video portal, which is the type of traffic that dominates Internet nowadays. The second scenario attempts to mimic traffic of users accessing modern Web 2.0 pages with plenty of multimedia content such as high-resolution images. As shown in Table 8.11, it is assumed that content popularity of both of them follows a Zipf distribution [Rossi and Rossini, 2011]. For this setup, 20 popularity classes are taken into account. As several studies have shown [Yu et al., 2011; TubeMogul, 2009] that most content objects are unpopular and only a few content objects are very popular, the number of content objects to be included in each popularity class is mapped to a Zipf distribution with $\alpha = 1$ and with inverted classes, i.e., most content objects are included in class 19 and fewest files in class 0.

As also described in Table 8.11, file sizes within each popularity class are different. Based on existing YouTube models [Abhari and Soraya, 2010], file size distribution for a YouTube scenario is set to a gamma distribution with $\alpha = 1.8$ and $\beta = 5500$. The file sizes for web server traffic are considerably smaller [Williams et al., 2005]. However, these file sizes have increased during the last years, and it is safe to assume that they keep increasing in the future. Transmitted NDN packets

Parameter	**Web Server**	**YouTube**
Requests	Every 5 seconds	
Request Popularity	Zipf distribution with $\alpha = 1$	$\alpha = 2$
File Distribution per Popularity Class	Zipf distribution, $\alpha = 1$ mapped to inverse classes	
File Sizes per Popularity Class	Gamma distribution, $\alpha = 1.8, \beta = 1200$ min. 50KB max. 50MB	Gamma distribution, $\alpha = 1.8, \beta = 5500$ min. 500KB max. 100MB

Table 8.11: Evaluation Parameters

need to have a certain minimum size to be efficient, e.g., segment size of 4096 bytes or more, to avoid too large overhead for content headers including names and signatures. Therefore, it is assumed that for future NDN traffic, many small files may be aggregated to larger data packets in order to use up the entire segment or NDN would only be applied to large static files, e.g., pictures or embedded videos, and not small text files that may change frequently. Therefore, a gamma distribution with $\alpha = 1.8$ and $\beta = 1200$ was selected for the web server scenario.

As for the content requests to be performed by users (Consumers) during the simulation, a parameter of $\alpha = 1$ is considered realistic for web server traffic and $\alpha = 2$ is used for YouTube traffic.

Evaluation Metrics

Finally, five different metrics were evaluated to assess accuracy of the strategies, performance improvements for end users and potential savings for operators:

1. The position of an optimal solution (highest profit destination) in the score-sorted ranking of destination alternatives is derived from the output of the AHP where decision. The optimal solution is the location where the group of users was on which the requested volume of content recently migrated was the highest. If it is in the first positions (1, 2, 3, etc.) of the aforementioned ranking, it means that the decisions were good and will yield benefits for the users.

2. The number of cache hits, which enables the comparison of strategies and the benefit to be quantified in terms of end users perspective and possible network bandwidth savings.

3. The average download latency experienced by users, which considers the best weight set from previous metrics evaluation.

4. The aggregated usage of bandwidth at the core interfaces (S1 and X2), which evaluates the overhead caused by different strategies and how the load becomes distributed.

5. The comparison of timings for FMC in the different scenarios to evaluate if migrations are made on time when they are reactive (no predictions) or proactive (mobility predicted).

Results

In the graphs below, results from the experiments defined in the previous section can be observed. To assess the first evaluation metric a comparison is made between the different weight sets, and a Cumulative Distribution Function (CDF) is generated for each ranking position. With the CDF, it is possible to obtain the cumulative percentage of occurrences of the optimal solution within the n first positions of the ranking. For example, if the CDF is 0.5 for an x axis value of 3, it means that in 50% of the decisions the optimal solution was within the 3 first positions of the ranking.

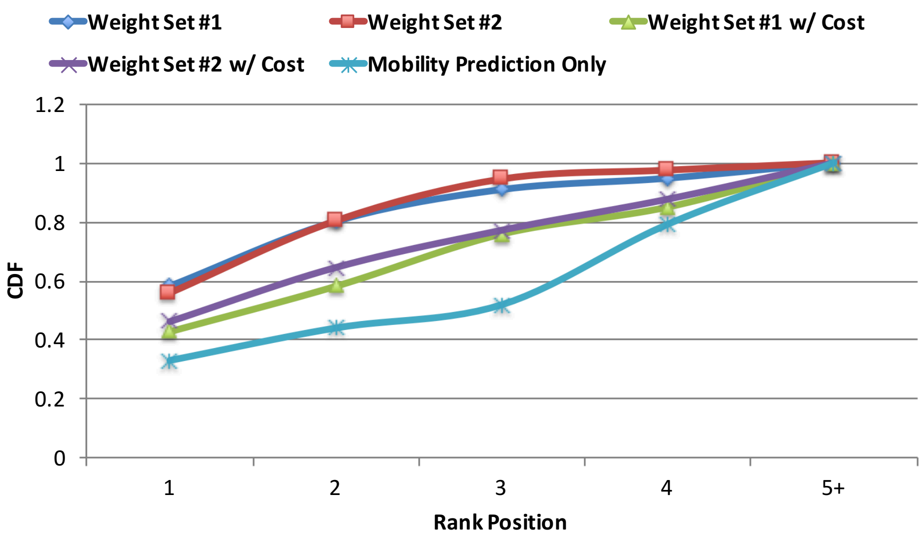


Figure 8.13: Optimal Solution in Ranking

In Fig. 8.13, results show that weight set #1 has a higher percentage of optimal solutions at the first position of the ranking, meaning that selections were perfect in almost 60% of the cases. However, the results of weight set #2 converge quicker to 100% of the cases, surpassing weight set #1 after (and including) rank position number 2. Overall, one may assume that weight set #2 selects better options than any other weight set, especially considering that the optimal solution was within the first three positions of the ranking in more than 90% of the cases. This can be easily explained by the accuracy of mobility prediction, which can have a great variance and does not account for the time users spend at the predicted locations. At the same time, giving priority to destinations where most of cached content is not the same as in the origin has a big inherent potential to be explored from the beginning.

When looking at the weight sets but considering cost, the trend is slightly different. Weight set #2 with cost outperforms weight set #1 with cost from the beginning, with the optimal solution being in the first position of the ranking almost 50% of the cases and in more than 80% of the cases the optimal solution being in the first 4 positions of the ranking. These results are according to what was expected, as cost limits the performance but considering it still delivers a good trade-off for both end users and network operators.

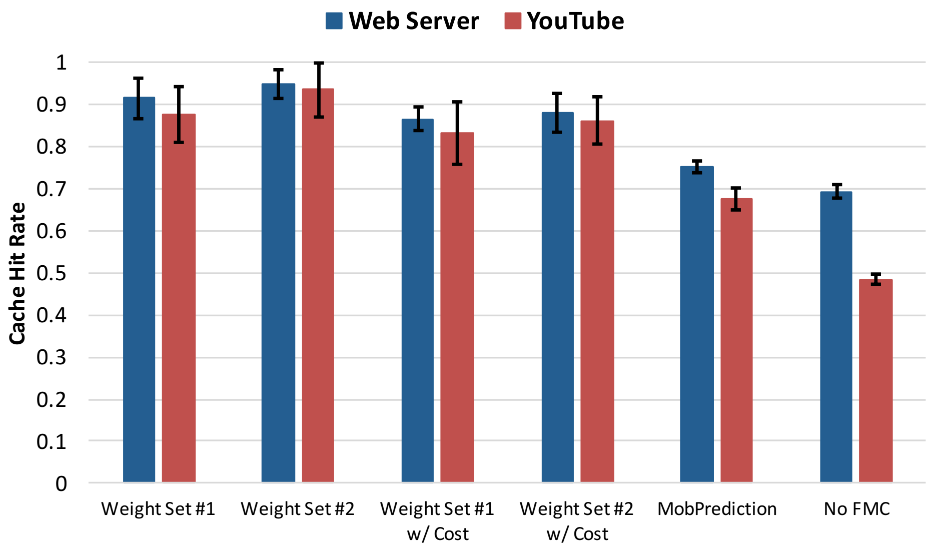


Figure 8.14: Cache Hit Rates

As for the second evaluation metric, a comparison between our strategies with different weight sets, mobility prediction only [Vasilakos et al., 2012b] and no use of FMC (default edge caching strategy) is shown in Fig. 8.14, evaluating cache hits in both the YouTube and Web Server scenarios.

From the depicted results, one may observe that cache hit rates tend to be lower for the YouTube scenario because of bigger file sizes and a different Zipf distribution. However, in this particular case our FMC strategies show the biggest difference towards simple Mobility Prediction and No FMC. For instance, the cache hit rate using weight set #2 is up to 40% higher than with the default strategy without FMC, and over 20% higher than relying solely on Mobility Prediction.

As for the Web Server scenario, the benefit is not so high (up to 20% less). Such fact is explained by the characteristics of web server traffic, which has a lot of small objects that are easily cached even if the cache storage space is low. Therefore, users may find most of the content already distributed over the network, and migration strategies do not copy a large amount of content that can yield benefits. However, multimedia content now accounts for the most traffic in mobile networks

[Cisco Systems, 2015], and we can easily conclude that FMC content migration strategies deliver their biggest performance for the biggest part of the traffic.

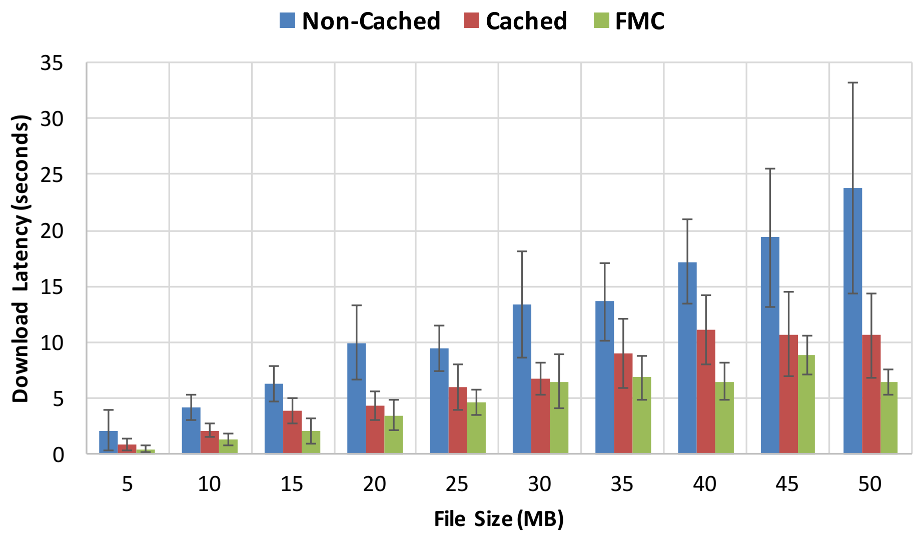


Figure 8.15: Average Content Download Latency - Web Server

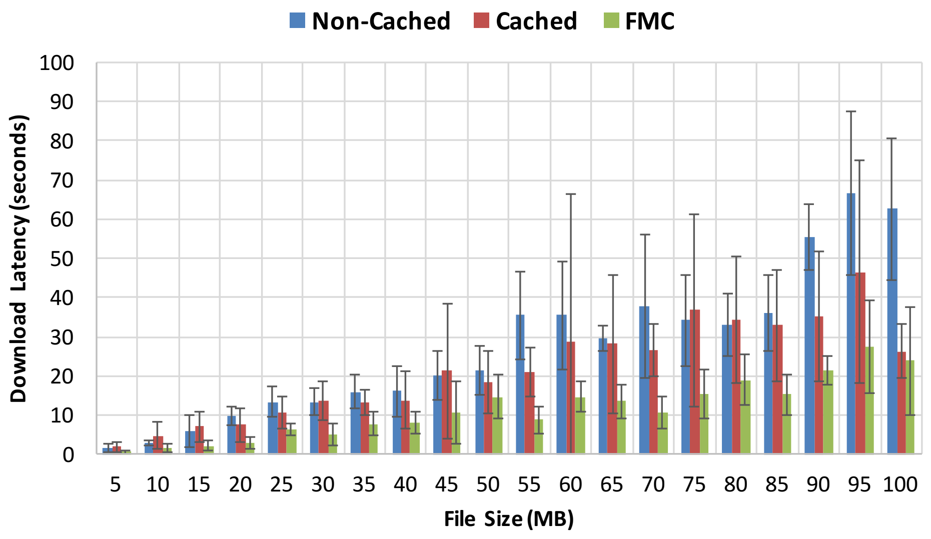


Figure 8.16: Average Content Download Latency - YouTube

In Fig. 8.15 and Fig. 8.16, a comparison between the different strategies (no edge caching, edge caching as described in Chapter 7 and FMC) is presented for the Web Server and YouTube scenarios. Here, the FMC strategy is considered to be Weight Set #2 with cost, thus having the highest cache hit rates together with possible savings in core bandwidth. First, we may observe that data points present high variance, caused by the method they were obtained with. As the number of files is too big to represent, sampling was performed and files were grouped in 5MB increments, i.e. $[0, 5[$, $[5, 10[$, etc. This sampling considers the sizes of all files generated in the multiple runs, and has the influence of random seed factors, network conditions, processing and others. Results confirm in an end-to-end user perspective what was visible when comparing cache hit rates: improvements experienced by end users are considerable and caches are used more efficiently, i.e. cached content corresponds mostly to content that will actually be requested by users. This is true for both scenarios, and again we may easily see that improvement towards regular edge caching is much bigger when multimedia traffic is considered and content sizes tend to increase. Moreover, the variance of results is much lower with FMC strategies. This ensures a more stable experience for end users.

As for aggregated core bandwidth usage, Fig. 8.17 depicts a comparison for the different strategies (no edge caching, edge caching and FMC), in the two scenarios and also in different LTE core interfaces (S1-U and X2). First, there is a high variance for the non-cached scenarios and even a slight variance for edge caching scenarios, explained by the different patterns of users requests and movements and the lack of convergence in content distribution towards the users interests. Second, we observe that, as expected, the aggregated usage of the S1-U interfaces is clearly reduced for both scenarios when caching at the edge. Third, we can also see that there is an overhead created by using FMC strategies when comparing to edge caching. This overhead becomes more clear over time, when caches start to be filled and more content is migrated, but it is compensated by the usage of the X2 interfaces between eNBs. This balances the load and at a certain point in time even adds more load to the X2 interface (prioritize), thus moving traffic away from the EPC and using available resources more efficiently. Finally, we conclude that there are differences between scenarios, especially because of the file sizes being bigger in YouTube traffic. This leads to a higher bandwidth consumption reduction with edge caching in the YouTube scenario, but also a slightly higher overhead for FMC strategies. Despite that, reductions in the usage of S1-U interfaces, and therefore in EPC, are still meaningful because of more traffic being offloaded using X2 interfaces.

Finally, in Fig. 8.18 we plot the combined average execution times of different components for the FMC strategies (in both traffic scenarios) together with the average available time brought by both X2 handover procedures and mobility prediction. All values have a confidence level of 95% and confidence intervals are present in the plot. We see that despite the accuracy of mobility prediction (about 50%), on average there is still plenty of available time for decisions and other cache operations (bear in mind that we are using a logarithmic scale). Using mobility prediction, in both scenarios the FMC components are able to execute within the available time frame. At the same time, if FMC operations are triggered reactively (no prediction), the handover delay is only enough for the decisions process. This means that the content transfer procedure will only start after the user is already

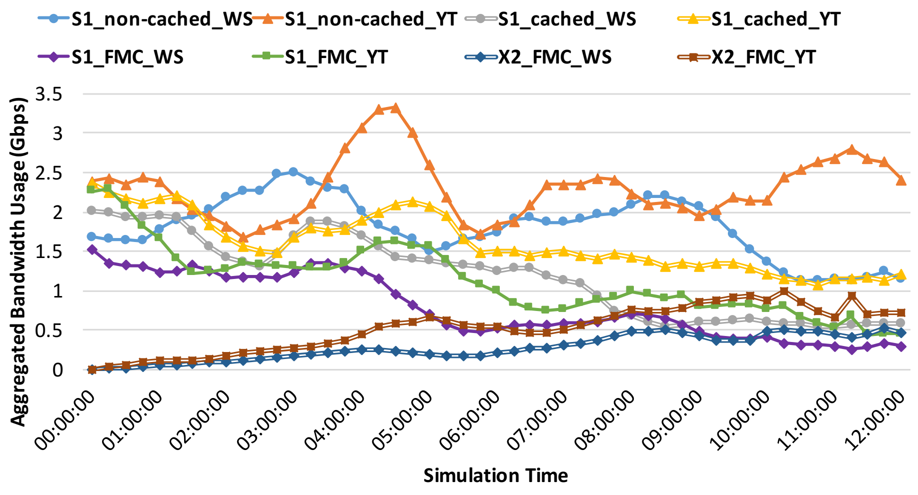


Figure 8.17: Aggregated Core Bandwidth Usage

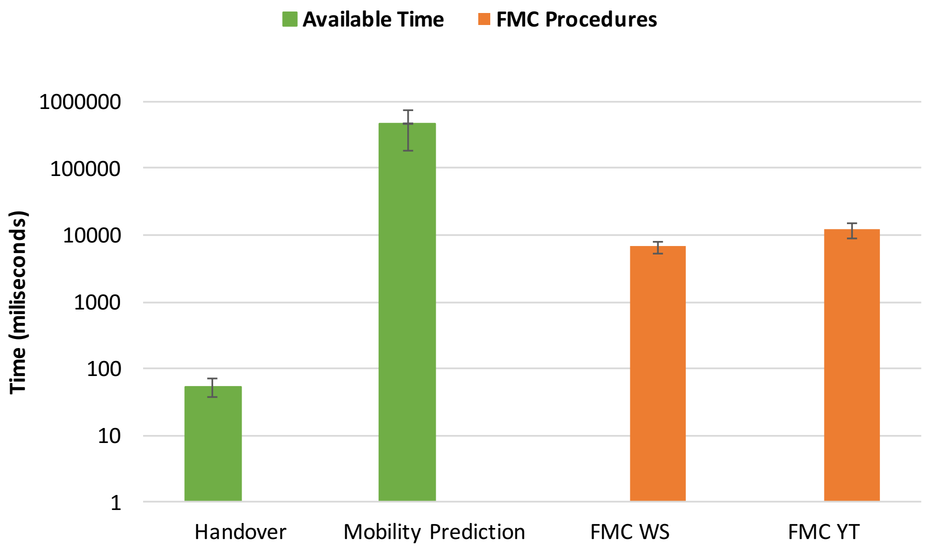


Figure 8.18: Execution Times of Components

at the new location, and in the worst case scenario it will have some initial cache misses while the content is still transferring. Overall, the impact of this behavior is not very high, as from previous figures we still observed very high cache hit ratios.

8.5 Summary

As users move to different locations, they still want to access content on which they are interested with low latency and without delays or breaks, especially if dealing with multimedia content. From the network perspective, this can only be granted if caches exist at the edge of mobile networks and the content kept in those caches (with limited resources) is the right content, i.e. popular content that local users will consume.

A number of related proposals already exist, but cannot be applied to content (only to services) or have other limitations such as assuming a very specific scenario or scope.Therefore, broader approaches, able to deal with content migration, have been proposed to handle multiple criteria decisions with different techniques and considering multiple factors that will trigger content migration.

A first evaluation was performed in a standalone manner, i.e. single components running in a preset scenario to select the most adequate and optimal algorithm for content subset selection. It proved that the selected algorithm (MeTHODICAL) is efficient while providing an accuracy always above 80% when compared to the optimal solutions determined by Knapsack. Also, it proved to be much better than other algorithms in the literature and more efficient to handle decisions with very strict timings.

A second evaluation was done in a realistic environment with the full MCN context and a testbed. The goal was to evaluate end-to-end user experience, measuring cache-hits enabled by FMC migrations and user satisfaction. Results show a smooth operation with real-time applications with the FMC model being able to deliver content with lower latency to end-users while, simultaneously, allowing them to be more satisfied. It is also clear that, as in the simulation experiments, operators may benefit from cache hit rates that allow huge savings of network backhaul.

A last evaluation was done in a simulation environment to enable large scale experiments and also to consider components and factors that were not available in more realistic environments. The evaluation in terms of performance, considering multiple weight values for decisions of the "where to mgirate" and different scenarios, achieved important results. When comparing to the case where default NDN caching strategies are used, clear benefits can be observed and quantified, leading to the conclusion that not only FMC enhanced caching strategies are the way to go when handling edge caches, but also that the architecture proposed together with all its decision mechanisms can achieve the goal of delivering content with lower latency to end users while efficiently using and saving well-valued network bandwidth.

Chapter 9

Conclusions and Outlook

Mobile networks already play a key role in the daily lives of millions of people worldwide, and this trend is expected to become even more important as a growing number of devices become connected to the Internet. Concepts such as Internet of Things (IoT), Smart Cities and M2M communication use mobile networks as their backhaul for information transmission. Moreover, multimedia content keeps growing as the major source of traffic in the Internet, with content providers like Netflix, Hulu and YouTube accounting for most of the traffic nowadays. These factors contribute to the need for fast and reliable networks, together with highly optimized content delivery strategies that can cope with the extreme demand. In this thesis, a set of contributions towards enhanced content delivery in mobile networks was presented and evaluated. These contributions are deeply related and integrated with each other (see Section 1.3.6), and together become a foundation for new strategies and concepts that can make a difference in the field. Section 9.1 briefly describes the main challenges tackled by those contributions, and Section 9.2 summarizes the contributions yet again. Finally, Section 9.3 outlines a set of topics and strategies that can be considered for future work and greatly improve the contributions of this thesis.

9.1 Challenges Addressed in the Thesis

The challenges and problems to which the contributions of this thesis attempted to provide solutions mainly concern the following domains:

- **Experimentation Methodologies:** The approach that many people follow in research is based on either quick prototyping or simulation to compare their findings with existing approaches as easily as possible. However, simulation results are not always replicated in real world, and most of the times cannot be even applied in real world scenarios. Thus, it becomes of utter importance to perform real-world prototyping in testbed environments. The main issue is that it is a very time consuming and tedious task. Many hours are spent attempting to setup the required platforms, have them running as expected and later on prepare the prototypes to be run in that environment.

- **Restrictions Imposed by Technology:** Although the pace of technological evolution is quite high nowadays, technology is still a barrier to achieve certain goals. For instance, radio spectrum and the way it is used creates limitations in terms of bandwidth and density of mobile network's cells. Also, computing resources, in particular general purpose computing resources as the ones present in typical cloud data centers, are still not capable of flawlessly running tasks that have been running in dedicated hardware for the past years. This seriously hinders efforts to virtualize and cloudify certain parts of the mobile networks.
- **Restrictions Imposed by Standards and Specifications:** Standards and specifications are not only important, but a necessity. However, most of them are designed and defined by consortiums such as 3GPP where most researchers have no voice. They are usually influenced by large corporations and targeted to serve their business interests. As mobile network operators and vendors are required to follow these standards and specifications, limitations to what can be done and how it can be done become a reality.
- **Lack of Integration:** A very large number of new technologies and concepts is being explored and used nowadays. Nevertheless, there is little integration between most of them and the benefits that they can provide us with have to be explored either in a standalone fashion or in a poorly integrated environment where frustration leads to low adoption by the users or performance benefits are so few that we may even believe that there is no advantage on having them integrated with each other.
- **Obsolete Protocols:** Most protocols in the Internet and mobile networks were developed decades ago, and were not designed to handle what the reality of today demands. This is true for a lot of services that nowadays are part of our daily life and we have learned how to deal with it, but becomes a significant issue when it starts affecting the core of all network communication. That is particularly real when discussing IP and its limitations, as recently the migration from IPv4 to IPv6 clearly shows that any changes are difficult although they are urgent to face the growing list of issues.

9.2 Thesis Summary

The challenges described above were faced during the work that lead to this thesis, and were addressed with a set of solutions, concepts, mechanisms and architectures presented in Chapters 3-8. While Chapter 3 describes the evaluation platforms used to perform the evaluations of this thesis, subsequent chapters present solutions that depend on each other and together create a base framework to achieve the common goal of improving content delivery in mobile networks.

In Chapter 3 evaluation platforms to evaluate the contributions of this thesis in the most realistic possible scenarios were introduced. First, a cloud computing testbed that includes state of the art software and hardware was created from scratch to support the work of this thesis and be part of a bigger testbed for the MCN project. This cloud computing testbed proved to be very reliable and have a very high performance, becoming even a key part of the MCN projects towards end-to-end evaluations. It also demonstrated to be very flexible to handle the evaluations performed for this thesis, always delivering consistent results and allowing for experiments that were important for the validation of implemented mechanisms. Then, a simulation environment was also briefly introduced. This complemented the testbed environment and allowed for experiments that could not run in a realistic environment due to the lack of dependencies and to the lack of resources to match the needed scale.

In Chapter 4 a framework for content delivery based on cloud computing principles was presented. This includes a service named ICNaaS, which integrates in an end-to-end prototype developed for MCN and implements all the key cloud principles while being ready for the particularities of ICN. Moreover, it serves as a base for the remaining contributions of this thesis as having a quick way to deploy ICN in a realistic scenario makes evaluations of high quality and in a reasonable amount of time. With its mechanisms for automated deployment in a virtualized infrastructure, it enables fast deployment of a content delivery system in any new network without much compromise in terms of overall performance. Also, results from the performed evaluation have shown that algorithms used for scaling decisions quickly react to ensure the load coming from user requests is well distributed and that their experienced download performance is not affected, while at the same time computing, storage and network resources are used in an efficient manner that enables cost savings for operators.

In Chapter 5 the integration of LTE mobile networks and ICN was explored to gather together the benefits of both and explore new paradigms and strategies for enhanced content delivery. It includes a modified LTE EPS and mechanisms to remain 3GPP compliant and maintain performance. This work was of extreme importance, as it serves as an enabler for the remaining contributions to be described below. Its evaluation demonstrated that it is feasible to integrate content delivery approaches such as ICN within LTE mobile networks, as the processing latencies are not much affected. Additionally, it also became evident that ICN has greater benefits that common protocols such as HTTP, allowing for greater performance on content delivery to the users, better support for mobility due to the lack of sessions and adding little overhead to the processing of LTE eNBs.

In Chapter 6 the combination of different radio technologies is exploited by using ICN and thus avoiding lower layer incompatibilities and complexities. By going one step further from typical offloading and allowing load balancing adapted to network conditions of each technology, results demonstrated that it can improve the performance for end users by using the resources of multiple technologies while saving energy on their devices by giving priority to less power hungry technologies and, overall, saving resources for mobile network operators. Moreover, it offers a way to exploit existing infrastructures that usually are not used up to its full potential such as Wi-Fi networks.

In Chapter 7 strategies for persistent caching at the edge of mobile networks were introduced. These focused on mechanisms to ensure that storage is efficiently used towards the interests of users in the vicinity of the caches and that cache hit rates are greatly increased, and also that content is properly distributed to avoid high traffic volumes at and near the root of the tree (content source). Moreover, these mechanisms proven to be fast and reliable although the implementation of repositories of CCN is not very efficient.

In Chapter 8 the main idea was to optimize the usage of edge caches by filling them up with the content most likely to be requested by nearby users. With that purpose, a number of strategies for content migration with different decisions mechanisms were described. By relying on fast and efficient decisions made according to a set of criteria, mechanisms ensure that edge caches converge more quickly to the interests of nearby users, becoming ready even when users are still not there. These strategies proven to significantly increase performance experienced by end users when downloading content while reducing traffic towards the core of the network.

9.3 Future Work

The work conducted in this thesis is in a very important field nowadays, and the topics that are covered are also very hot topics in the research community. In this section, a brief list of topics to explore and possible paths to follow is presented per chapter. These can be considered as future work to further enhanced the contributions already presented in this thesis.

First, the platforms described in Chapter 3 can be improved. Although simulation platforms can always be extended to deal with new protocols and concepts, new solutions are always surfacing online and more complete (higher quality) traces can always be used to feed the simulation and make it more realistic. As for the cloud computing testbed platform, it relies on a software framework that is growing exponentially. New features and improvements are being developed every single day, and these have to be implemented by upgrading the software and later on exploiting them to better handle the tasks to be performed. Moreover, and as expected, hardware becomes obsolete quickly. Upgrades to hardware both to extend the availability of resources and to improve performance are bound to happen at a point.

In Chapter 4 a service (ICNaaS) was presented and evaluated. Although it was developed in the context of MCN and the project has ended, its code is open source and is available online. This makes it widely available for other developers to further enhanced it and add relevant features. Namely, it can be expanded to have better and more automatic scalability using OpenStack Ceilometer, make usage of new concepts such as Software-Defined Networking (SDN) to improve the performance of ICN in the cloud and also implement better mechanisms to handle topology discovery and management.

Chapter 5 introduced a modified 3GPP LTE EPS architecture and mechanisms to integrate ICN with it and provide extra benefits inherent to ICN such as caching closer to the users. Although the results demonstrated that the system is feasible and actually has a positive tradeoff between gain and losses, it may still be improved. Recent ideas on turning the LTE eNBs into fully-IP devices are surfacing, and these together with SDN concepts may lead to more optimized systems that have less impact on performance, greater flexibility and more chances of being adopted by mobile network operators.

In Chapter 6 mechanisms for the usage of multiple radio technologies simultaneously were introduced. These rely on ICN and thus on its advantages, providing a seamless and high-level way of performing load balancing and profiting from the existence of coverage of multiple technologies at a given place. However, many more metrics could be considered for the calculated radios (e.g. bandwidth predictions, radio resource blocks usage, network energy, etc.) and a more integrated setup (with current solutions) can be considered. Moreover, simulations can be run in more complex and complete scenarios to evaluate them in a more realistic way.

In Chapter 7 strategies were presented to have persistent caching at the edge of mobile networks and also to distribute content in a tree-oriented topology in a way that optimizes storage usage. The results were quite good when comparing to existing solutions, but more possibilities can be explored. For example, better algorithms and mechanisms to maintain content at CCN repositories and thus optimize the deletion process. Also, deletion may happen with more criteria and be more attached to strategies as the ones discussed in Chapter 8.

Finally, in Chapter 8 follow-me caching strategies were introduced. These, as the ones described in Chapter 7, aim at efficiently using the storage space available at the edge of mobile networks and ensuring that at any given time the content in cache is as close as possible to the ideal content to be cached for the users in its vicinity. These strategies may also be considered at other levels, as it is not clear that in the future there will be smaller cells at the edge (the research community is divided). Additionally, considering more criteria and extra context information such as time of the day and events data may lead to more optimized copies and thus a more efficient system.

Acronyms

16QAM 16 Quadrature Amplitude Modulation.

3G 3rd Generation.

3GPP 3rd Generation Partnership Project.

4G 4th Generation.

5G 5th Generation.

64QAM 64 Quadrature Amplitude Modulation.

AAAaaS Authentication, Authorization and Accounting as a Service.

AHP Analytical Hierarchy Process.

AP Access Point.

API Application Programming Interface.

APNs Access Point Names.

BGP Border Gateway Protocol.

BSS Business Support System.

C-RAN Cloud Radio Access Network.

CAPEX Capital Expenditures.

CC Cloud Controller.

CCN Content-Centric Networking.

CDF Cumulative Distribution Function.

CDNs Content Delivery Networks.

CPU Central Processing Unit.

CS Content Store.

DCE Direct Code Execution.

DCs Data Centers.

DNS Domain Name System.

DNSaaS Domain Name System as a Service.

DONA Data-Oriented Network Architecture.

DoS Denial of Service.

DSSaaS Digital Signage System as a Service.

DTNs Delay-Tolerant Networks.

EDGE Enhanced Data rates for Global Evolution.

EEU Enterprise End User.

EMA Exponential Moving Average.

eNB Evolved Node B.

EPC Evolved Packet Core.

EPCaaS Evolved Packet Core as a Service.

ePDG Evolved Packet Data Gateway.

EPS Evolved Packet System.

ETSI European Telecommunications Standards Institute.

FI Future Internet.

FIB Forwarding Information Base.

FIFO First-In-First-Out.

FMC Follow-Me Cloud.

FP7 Seventh Framework Programme.

FPGAs Field Programmable Gate Arrays.

GPRS General Packet Radio Service.

GSM Global System for Mobile communications.

GTP-U General Packet Radio Service Tunneling Protocol - User plane.

HSPA High Speed Packet data Access.

HSS Home Subscriber Server.

HTTP Hyper Text Transfer Protocol.

IaaS Infrastructure as a Service.

ICN Information-Centric Networking.

ICNaaS Information-Centric Networking as a Service.

IMSaaS IP Multimedia Subsystem as a Service.

IoT Internet of Things.

IP Internet Protocol.

IPv4 IP Version 4.

iSCSI Internet Small Computer Systems Interface.

IT Information Technology.

ITG Infrastructure Template Graph.

LBaaS Load Balancing as a Service.

LDAP Lightweight Directory Access Protocol.

LRU Least Recently Used.

LTE Long Term Evolution.

M2M Machine-to-Machine.

MaaS Monitoring as a Service.

MAC Medium Access Control.

MCDM Multiple-Criteria Decision-Making.

MCN Mobile Cloud Networking.

MCS Modulation and Coding Scheme.

MEC Mobile Edge Computing.

MIMO Multiple Input Multiple Output.

MME Mobility Management Entity.

MNO Mobile Network Operator.

MOBaaS Mobility Prediction as a Service.

MTU Maximum Transfer Unit.

NAS Non-Access Stratum.

NASA National Aeronautics and Space Administration.

NC Enhanced Data rates for Global Evolution.

NDN Named Data Networking.

NDO Named Data Object.

NetInf Network of Information.

NFV Network Functions Virtualization.

NFVI Network Functions Virtualization Infrastructure.

NIST National Institute of Standards and Technology.

NRS Name Resolution Service.

NS Network Simulator.

OAI OpenAirInterface.

OCCI Open Cloud Computing Interface.

OFDMA Orthogonal Frequency-Division Multiple Access.

OID Object Identifier.

OPEX Operational Expenditures.

OS Operating System.

OSPF Open Shortest Path First.

OSS Operations Support System.

OTP Over The Top.

P-GW PDN Gateway.

P2P Peer-to-Peer.

PCRF	Policy and Charging Rules Function.
PDCP	Packet Data Convergence Protocol.
PDN	Packet Data Network.
PER	Packet Error Rate.
PIT	Pending Interest Table.
PKI	Public Key Infrastructure.
PSIRP	Publish-Subscribe Internet Routing Paradigm.
QoE	Quality of Experience.
QoS	Quality of Service.
QPSK	Quadrature Phase Shift Keying.
RAID	Redundant Array of Independent Disks.
RAM	Random Access Memory.
RAN	Radio Access Network.
RANaaS	Radio Access Network as a Service.
RCBaaS	Rating, Charging and Billing as a Service.
REST	Representational State Transfer.
ROHC	Robust Header Compression.
RPM	Rotations Per Minute.
RSSI	Received Signal Strength Indicator.
RX PRACH	Receiving Physical Random Access Channel.
S-GW	Serving Gateway.
SAN	Storage Area Network.
SAS	Serial Attached SCSI.
SC-FDMA	Single-Carrier Frequency-Division Multiple Access.
SDN	Software-Defined Networking.
SM	Service Manager.

SO	Service Orchestrator.
SSH	Secure Shell.
STG	Service Template Graph.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.
UE	User Equipment.
UMTS	Universal Mobile Telecommunications System.
URL	Uniform Resource Locator.
USIM	Universal Subscriber Identity Module.
VIM	Virtualized Infrastructure Manager.
VM	Virtual Machine.
VNFs	Virtual Network Functions.
VoD	Video On Demand.
WAN	Wide Area Network.
WiMAX	Worldwide Interoperability for Microwave Access.

Bibliography

[3GPP, 2015] 3GPP (2015). Evolved Universal Terrestrial Radio Access (E-UTRA). TS 36.211-213 and TS 36.300.

[Aad et al., 2013] Aad, I., Gomes, A., Crosta, P. S., and et al. (2013). Design of Mobile Platform Architecture, IMSaaS and DSN applications. Technical report, FP7 MCN Consortium.

[Abhari and Soraya, 2010] Abhari, A. and Soraya, M. (2010). Workload generation for youtube. *Multimedia Tools Appl.*, 46(1):91–118.

[Afanasyev et al., 2012] Afanasyev, A., Moiseenko, I., and Zhang, L. (2012). ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN.

[Agyapong et al., 2014] Agyapong, P., Iwamura, M., Staehle, D., Kiess, W., and Benjebbour, A. (2014). Design considerations for a 5g network architecture. *Communications Magazine, IEEE*, 52(11):65–75.

[Ahlgren et al., 2012] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. (2012). A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36.

[Ahmed and Ahmed, 2016] Ahmed, A. and Ahmed, E. (2016). A survey on mobile edge computing. In *IEEE International Conference on Intelligent Systems and Control.*

[Akamai, 2015] Akamai (2015). Akamai CDNs.

[Alcatel-Lucent, 2009] Alcatel-Lucent (2009). LTE Evolved Packet System Architecture.

[Ali, 2013] Ali, I. (2013). Architecture enhancements for non-3GPP accesses. TS 23.402.

[Amirrudin et al., 2013] Amirrudin, N., Ariffin, S., Malik, N., and Ghazali, N. (2013). User's mobility history-based mobility prediction in lte femtocells network. In *RF and Microwave Conference (RFM), 2013 IEEE International*, pages 105–110.

[Anastasiades, 2016] Anastasiades, C. (2016). *Information-Centric Communication in Mobile and Wireless Networks*. PhD thesis, University of Bern.

[Anastasiades et al., 2015] Anastasiades, C., Gomes, A., Gadow, R., and Braun, T. I. (2015). Persistent caching in Information-Centric networks. In *40th Annual IEEE Conference on Local Computer Networks (LCN 2015)*, Clearwater Beach, USA.

[Andonov et al., 2000] Andonov, R., Poirriez, V., and Rajopadhye, S. (2000). Unbounded knapsack problem: Dynamic programming revisited. *European Journal of Operational Research*, 123(2):394 – 407.

[Antonescu et al., 2013] Antonescu, A.-F., Gomes, A., Robinson, P., and Braun, T. (2013). Sla-driven predictive orchestration for distributed cloud-based mobile services. In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 738–743.

[Astely et al., 2009] Astely, D., Dahlman, E., Furuskar, A., Jading, Y., Lindstrom, M., and Parkvall, S. (2009). Lte: the evolution of mobile broadband. *Communications Magazine, IEEE*, 47(4):44–51.

[Bakiras, 2005] Bakiras, S. (2005). Approximate server selection algorithms in content distribution networks. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 3, pages 1490–1494 Vol. 3.

[Balamash and Krunz, 2004] Balamash, A. and Krunz, M. (2004). An overview of web caching replacement algorithms. *Communications Surveys Tutorials, IEEE*, 6(2):44–56.

[Belson, 2016] Belson, D. (2016). Akamai State of the Internet Report, Q4 2015. Technical Report 4, Akamai Technologies.

[Blankenship et al., 2004] Blankenship, Y., Sartori, P., Classon, B., Desai, V., and Baum, K. (2004). Link error prediction methods for multicarrier systems. In *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, volume 6, pages 4175–4179 Vol. 6.

[Bohnert et al., 2012] Bohnert, T. M., Edmonds, A., Braun, T., and et al. (2012). MCN Description of Work. Technical report, FP7 MCN Consortium.

[Carella et al., 2015] Carella, G., Gomes, A., Crosta, P. S., and et al. (2015). Evaluation of Mobile Platform, IMSaaS and DSN. Technical report, FP7 MCN Consortium.

[Cha et al., 2007] Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S. (2007). 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. ACM.

[Chen et al., 2012] Chen, F., Guo, K., Lin, J., and La Porta, T. (2012). Intra-cloud lightning: Building cdns in the cloud. In *INFOCOM, 2012 Proceedings IEEE*, pages 433–441.

[Chiou et al., 2009] Chiou, C. W., c. Chen, C., and c. Chiou, S. (2009). A decision-making model of budget allocation for the restoration of traditional settlement buildings. In *Management and Service Science, 2009. MASS '09. International Conference on*, pages 1–4.

[Cho et al., 2012] Cho, K., Lee, M., Park, K., Kwon, T., Choi, Y., and Pack, S. (2012). Wave: Popularity-based and collaborative in-network caching for content-oriented networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 316–321.

[Chon et al., 2012] Chon, Y., Shin, H., Talipov, E., and Cha, H. (2012). Evaluating mobility models for temporal prediction with high-granularity mobility data. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pages 206–212.

[Choudhury et al., 2010] Choudhury, M. D., Sundaram, H., John, A., Seligmann, D. D., and Kelliher, A. (2010). "birds of a feather": Does user homophily impact information diffusion in social media? *CoRR*, abs/1006.1702.

[Cisco Systems, 2015] Cisco Systems (2015). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf`.

[CloudFlare, 2015] CloudFlare (2015). CloudFlare.

[Corici et al., 2015] Corici, M., Gomes, A., Crosta, P. S., and et al. (2015). Final Implementation of IMSaaS, DSN, and Mobile Platform. Technical report, FP7 MCN Consortium.

[Dannewitz et al., 2013] Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B., and Karl, H. (2013). Network of information (netinf) - an information-centric networking architecture. *Comput. Commun.*, 36(7):721–735.

[De Vleeschauwer and Robinson, 2011] De Vleeschauwer, D. and Robinson, D. C. (2011). Optimum caching strategies for a telco cdn. *Bell Labs Technical Journal*, 16(2):115–132.

[Demestichas et al., 2013] Demestichas, P., Georgakopoulos, A., Karvounas, D., Tsagkaris, K., Stavroulaki, V., Lu, J., Xiong, C., and Yao, J. (2013). 5g on the horizon: Key challenges for the radio-access network. *Vehicular Technology Magazine, IEEE*, 8(3):47–53.

[Detti et al., 2012] Detti, A., Pomposini, M., Blefari-Melazzi, N., Salsano, S., and Bragagnini, A. (2012). Offloading cellular networks with Information-Centric Networking: The case of video streaming. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–3.

[Ding et al., 2013] Ding, A. Y., Han, B., Xiao, Y., Hui, P., Srinivasan, A., Kojo, M., and Tarkoma, S. (2013). Enabling energy-aware collaborative mobile data offloading for smartphones. In *2013 IEEE International Conference on Sensing, Communications and Networking (SECON)*, pages 487–495.

[DiVi Networks, 2012] DiVi Networks (2012). Global Data Flow Report - 2012.

[Dominguez et al., 2011] Dominguez, A., Novo, O., Wong, W., and Valladares, T. (2011). Publish/subscribe communication mechanisms over psirp. In *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*, pages 268–273.

[Eagle and (Sandy) Pentland, 2006] Eagle, N. and (Sandy) Pentland, A. (2006). Reality mining: Sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4):255–268.

[Edmonds et al., 2013] Edmonds, A., Bohnert, T. M., Gomes, A., and et al. (2013). Overall Architecture Definition, Release 1. Technical report, FP7 MCN Consortium.

[Edmonds et al., 2015] Edmonds, A., Bohnert, T. M., Gomes, A., and et al. (2015). Overall Architecture Definition, Release 2. Technical report, FP7 MCN Consortium.

[Edmonds et al., 2012] Edmonds, A., Metsch, T., Papaspyrou, A., and Richardson, A. (2012). Toward an open cloud standard. *IEEE Internet Computing*, 16(4):15–25.

[ETSI, 2016] ETSI (2016). ETSI NFV Standards.

[Eum et al., 2012] Eum, S., Nakauchi, K., Shoji, Y., Nishinaga, N., and Murata, M. (2012). Catt: Cache aware target identification for icn. *Communications Magazine, IEEE*, 50(12):60–67.

[Fall and Farrell, 2008] Fall, K. and Farrell, S. (2008). Dtn: an architectural retrospective. *Selected Areas in Communications, IEEE Journal on*, 26(5):828–836.

[FasterXML, 2015] FasterXML (2015). Project Jackson.

[Fonseca, 2015] Fonseca, V. (2015). Smart Content Relocation in Content-Centric Networks. Master's thesis, University of Coimbra.

[Foster et al., 2008] Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10.

[Furuskar et al., 1999] Furuskar, A., Mazur, S., Muller, F., and Olofsson, H. (1999). Edge: enhanced data rates for gsm and tdma/136 evolution. *IEEE Personal Communications*, 6(3):56–66.

[Gadow, 2015] Gadow, R. (2015). Persistent Caching in Information-Centric Networking. Bachelor's thesis, University of Bern.

[Gaito et al., 2013] Gaito, S., Maggiorini, D., Quadri, C., and Rossi, G. (2013). Selective offload and proactive caching of mobile data in lte-based urban networks. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 271–274.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[Ghosh et al., 2005] Ghosh, A., Wolter, D. R., Andrews, J. G., and Chen, R. (2005). Broadband wireless access with wimax/802.16: current performance benchmarks and future potential. *IEEE Communications Magazine*, 43(2):129–136.

[Gomes and Braun, 2015a] Gomes, A. and Braun, T. (2015a). Feasibility of information-centric networking integration into lte mobile networks. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 628–634. ACM.

[Gomes et al., 2016a] Gomes, A., Braun, T., and Monteiro, E. (2016a). Enhanced caching strategies at the edge of lte mobile networks. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 341–349.

[Gomes and Braun, 2015b] Gomes, A. and Braun, T. I. (2015b). Load balancing in LTE mobile networks with Information-Centric networking. In *IEEE ICC 2015 - Third International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA) (ICC'15 - Workshops 05)*, London, United Kingdom.

[Gomes et al., 2016b] Gomes, A. S., Fonseca, V., Sousa, B., Palma, D., Simoes, P., Monteiro, E., and Cordeiro, L. (2016b). A mobile follow-me cloud content caching model. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 763–766.

[Gomes et al., 2016c] Gomes, A. S., Sousa, B., Palma, D., Fonseca, V., Zhao, Z., Monteiro, E., Braun, T., Simoes, P., and Cordeiro, L. (2016c). Edge caching with mobility prediction in virtualized {LTE} mobile networks. *Future Generation Computer Systems*, pages –.

[Google, 2015] Google (2015). Recommended upload encoding settings. `https://support.google.com/youtube/answer/1722171?hl=en`.

[Griffin et al., 2002] Griffin, T., Shepherd, F., and Wilfong, G. (2002). The stable paths problem and interdomain routing. *Networking, IEEE/ACM Transactions on*, 10(2):232–243.

[Gupta and Tang, 2006] Gupta, H. and Tang, B. (2006). Data caching under number constraint. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 1, pages 435–440.

[Haberland et al., 2013] Haberland, B., Derakhshan, F., Grob-Lipski, H., Klotsche, R., Rehm, W., Schefczik, P., and Soellner, M. (2013). Radio Base Stations in the Cloud. *Bell Labs Technical Journal*, 18(1):129–152.

[Han et al., 2006] Han, H., Shakkottai, S., Hollot, C. V., Srikant, R., and Towsley, D. (2006). Multi-path tcp: A joint congestion control and routing scheme to exploit path diversity in the internet. *IEEE/ACM Transactions on Networking*, 14(6):1260–1271.

[HTTP Archive, 2015] HTTP Archive (2015). Interesting stats. `http://httparchive.org/interesting.php`.

[Huang et al., 2012] Huang, J., Qian, F., Gerber, A., Mao, Z. M., Sen, S., and Spatscheck, O. (2012). A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 225–238, New York, NY, USA. ACM.

[Hwang et al., 1993] Hwang, C.-L., Lai, Y.-J., and Liu, T.-Y. (1993). A new approach for multiple objective decision making. *Computers & Operations Research*, 20(8):889–899.

[Jacobson et al., 2009] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 1–12, New York, NY, USA. ACM.

[Karimzadeh et al., 2015] Karimzadeh, M., Zhao, Z., Hendriks, L., de O. Schmidt, R., la Fleur, S., van den Berg, H., Pras, A., Braun, T., and Corici, M. J. (2015). Mobility and bandwidth prediction as a service in virtualized lte systems. In *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pages 132–138.

[Kim et al., 2012] Kim, D.-h., Kim, J.-h., Kim, Y.-s., Yoon, H.-s., and Yeom, I. (2012). Mobility Support in Content Centric Networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pages 13–18, New York, NY, USA. ACM.

[Koponen et al., 2007] Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. (2007). A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pages 181–192, New York, NY, USA. ACM.

[Kraemer et al., 2009] Kraemer, B. P., Rosdahl, J. W., and Stephens, A. P. (2009). IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages 1–565.

[Ksentini et al., 2014] Ksentini, A., Taleb, T., and Chen, M. (2014). A markov decision process-based service migration procedure for follow me cloud. In *Communications (ICC), 2014 IEEE International Conference on*, pages 1350–1354.

[Lahby et al., 2012] Lahby, M., Cherkaoui, L., and Adib, A. (2012). Article: New Optimized Network Selection Decision in Heterogeneous Wireless Networks. *International Journal of Computer Applications*, 54(16):1–7. Published by Foundation of Computer Science.

[Lampe et al., 2003] Lampe, M., Giebel, T., Rohling, H., and Zirwas, W. (2003). Per-prediction for PHY mode selection in OFDM communication systems. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 1, pages 25–29 Vol.1.

[Laurila et al., 2012] Laurila, J. K., Gatica-Perez, D., Aad, I., J., B., Bornet, O., Do, T.-M.-T., Dousse, O., Eberle, J., and Miettinen, M. (2012). The Mobile Data Challenge: Big Data for Mobile Computing Research. In *Pervasive Computing*.

[Lee et al., 2010] Lee, U., Rimac, I., and Hilt, V. (2010). Greening the internet with content-centric networking. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 179–182, New York, NY, USA. ACM.

[Li et al., 1999] Li, B., Golin, M., Italiano, G., Deng, X., and Sohraby, K. (1999). On the optimal placement of web proxies in the internet. In *INFOCOM '99.*

Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 3, pages 1282–1290 vol.3.

[Li and Ascheid, 2012] Li, H. and Ascheid, G. (2012). Mobility prediction based on graphical model learning. In *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, pages 1–5.

[Li et al., 2012] Li, Y., Lin, T., Tang, H., and Sun, P. (2012). A chunk caching location and searching scheme in content centric networking. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2655–2659.

[Liebsch et al., 2012] Liebsch, M., Schmid, S., and Awano, J. (2012). Reducing backhaul costs for mobile content delivery - an analytical study. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2895–2900.

[Liebsch and Yousaf, 2013] Liebsch, M. and Yousaf, F. (2013). Runtime relocation of cdn serving points - enabler for low costs mobile content delivery. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 1464–1469.

[Liu et al., 2011] Liu, H., Sun, Y., and Kim, M. S. (2011). Provider-level content migration strategies in p2p-based media distribution networks. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 337–341.

[Liu, 2009] Liu, W. (2009). Research on dos attack and detection programming. In *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*, volume 1, pages 207–210.

[Lundström and Hall, 2011] Lundström, A. and Hall, G. (2011). Wi-Fi Integration. Technical report, Ericsson.

[Martiquet, 2012] Martiquet, N. (2012). 3GPP system to Wireless Local Area Network (WLAN) interworking; System description. TS 23.234.

[MCN Consortium, 2015] MCN Consortium (2015). EC FP7 Mobile Cloud Networking project.

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology (NIST).

[Méndez et al., 2009] Méndez, M., Galván, B., Salazar, D., and Greiner, D. (2009). *Multiple-Objective Genetic Algorithm Using the Multiple Criteria Decision Making Method TOPSIS*, pages 145–154. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Mezzavilla et al., 2012] Mezzavilla, M., Miozzo, M., Rossi, M., Baldo, N., and Zorzi, M. (2012). A lightweight and accurate link abstraction model for the simulation of lte networks in ns-3. In *Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '12, pages 55–60, New York, NY, USA. ACM.

[Mirantis, 2016] Mirantis (2016). Mirantis Fuel for OpenStack. `https://www.mirantis.com/products/mirantis-openstack-software/`.

[Montpetit et al., 2012] Montpetit, M.-J., Westphal, C., and Trossen, D. (2012). Network coding meets information-centric networking: An architectural case for information dispersion through native network coding. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pages 31–36, New York, NY, USA. ACM.

[NGMN, 2013] NGMN (2013). Suggestions on Potential Solutions to C-RAN by NGMN Alliance. Technical report, The Next Generation Mobile Networks (NGMN) Alliance.

[Ni et al., 2003] Ni, J., Tsang, D., Yeung, I., and Hei, X. (2003). Hierarchical content routing in large-scale multimedia content delivery network. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 2, pages 854–859 vol.2.

[Nikaein et al., 2014] Nikaein, N., Marina, M. K., Manickam, S., Dawson, A., Knopp, R., and Bonnet, C. (2014). Openairinterface: A flexible platform for 5g research. *SIGCOMM Comput. Commun. Rev.*, 44(5):33–38.

[ns-3 Project, 2016a] ns-3 Project (2016a). ns-3: Direct Code Execution. `http://www.nsnam.org/overview/projects/direct-code-execution/`.

[ns-3 Project, 2016b] ns-3 Project (2016b). ns-3 Simulator. `https://www.nsnam.org/`.

[Nuggehalli et al., 2006] Nuggehalli, P., Srinivasan, V., Chiasserini, C., and Rao, R. (2006). Efficient cache placement in multi-hop wireless networks. *Networking, IEEE/ACM Transactions on*, 14(5):1045–1055.

[OpenStack Foundation, 2016] OpenStack Foundation (2016). OpenStack.

[Pal et al., 2005] Pal, S., Das, S. K., and Chatterjee, M. (2005). User-satisfaction based differentiated services for wireless data networks. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 2, pages 1174–1178 Vol. 2.

[PARC, 2015] PARC (2015). Project CCNx.

[Park and Macker, 1999] Park, V. and Macker, J. (1999). Anycast routing for mobile networking. In *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*, volume 1, pages 1–5 vol.1.

[Pavlou, 2011] Pavlou, G. (2011). Keynote 2: Information-centric networking: Overview, current state and key challenges. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 1–1.

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

[Podlipnig and Böszörmenyi, 2003] Podlipnig, S. and Böszörmenyi, L. (2003). A survey of Web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398.

[Psaras et al., 2012] Psaras, I., Chai, W. K., and Pavlou, G. (2012). 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. ACM.

[Psaras et al., 2011] Psaras, I., Clegg, R. G., Landa, R., Chai, W. K., and Pavlou, G. (2011). Modelling and evaluation of ccn-caching trees. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING'11, pages 78–91, Berlin, Heidelberg. Springer-Verlag.

[Qiu et al., 2001] Qiu, L., Padmanabhan, V., and Voelker, G. (2001). On the placement of web server replicas. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1587–1596 vol.3.

[R Foundation, 2014] R Foundation (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Rajagopal et al., 2002] Rajagopal, S., Srinivasan, N., Narayan, R., and Petit, X. (2002). Gps based predictive resource allocation in cellular networks. In *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, pages 229–234.

[Ramanan et al., 2013] Ramanan, B., Drabeck, L., Haner, M., Nithi, N., Klein, T., and Sawkar, C. (2013). Cacheability analysis of http traffic in an operational lte network. In *Wireless Telecommunications Symposium (WTS), 2013*, pages 1–8.

[Ravindran et al., 2012] Ravindran, R., Lo, S., Zhang, X., and Wang, G. (2012). Supporting seamless mobility in named data networking. In *Communications (ICC), 2012 IEEE International Conference on*, pages 5854–5869.

[Rimal et al., 2009] Rimal, B., Choi, E., and Lumb, I. (2009). A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pages 44–51.

[Rinne et al., 2009] Rinne, M., Kuusela, M., Tuomaala, E., Kinnunen, P., Kovacs, I., Pajukoski, K., and Ojala, J. (2009). A performance summary of the evolved 3g (e-utra) for voice over internet and best effort traffic. *IEEE Transactions on Vehicular Technology*, 58(7):3661–3673.

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

[Rossini and Rossi, 2013] Rossini, G. and Rossi, D. (2013). Evaluating {CCN} multi-path interest forwarding strategies. *Computer Communications*, 36(7):771 – 778.

[Ruiz et al., 2014] Ruiz, Y., Gomes, A., Crosta, P. S., and et al. (2014). First Implementation of IMSaaS, DSN and Mobile Platform. Technical report, FP7 MCN Consortium.

[Saaty, 1987] Saaty, R. (1987). The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, 9(3–5):161 – 176.

[Saaty, 1980] Saaty, T. L. (1980). *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. Mcgraw-Hill, New York, NY.

[Saaty, 1990] Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, 48(1):9 – 26. Decision making by the analytic hierarchy process: Theory and applications.

[Samaan and Karmouch, 2005] Samaan, N. and Karmouch, A. (2005). A mobility prediction architecture based on contextual knowledge and spatial conceptual maps. *Mobile Computing, IEEE Transactions on*, 4(6):537–551.

[Sarkissian, 2013] Sarkissian, H. (2013). The Business Case for Caching in 4G LTE Networks. `http://www.wireless2020.com/docs/LSI_WP_Content_Cach_Cv3.pdf`.

[Savic, 2011] Savic, Z. (2011). LTE Design and Deployment Strategies. `http://www.cisco.com/web/ME/expo2011/saudiarabia/pdfs/LTE_Design_and_Deployment_Strategies-Zeljko_Savic.pdf`.

[Seggern, 2006] Seggern, D. H. v. (2006). *CRC Standard Curves and Surfaces with Mathematica, Second Edition (Chapman & Hall/Crc Applied Mathematics and Nonlinear Science)*. Chapman & Hall/CRC.

[Serpanos et al., 2000] Serpanos, D., Karakostas, G., and Wolf, W. (2000). Effective caching of web objects using zipf's law. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 2, pages 727–730 vol.2.

[Shaikh et al., 2001] Shaikh, A., Tewari, R., and Agrawal, M. (2001). On the effectiveness of dns-based server selection. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1801–1810 vol.3.

[Siris et al., 2013] Siris, V. A., Anagnostopoulou, M., and Dimopoulos, D. (2013). Improving mobile video streaming with mobility prediction and prefetching in integrated cellular-wifi networks. *CoRR*, abs/1310.6171.

[Skowron et al., 2012] Skowron, M., Namal, S., Pellikka, J., and Gurtov, A. (2012). Inter Technology Load Balancing Algorithm for Evolved Packet System. In *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, pages 1–5.

[Smetters and Jacobson, 2009] Smetters, D. and Jacobson, V. (2009). Securing network content. Technical report, PARC.

[Sousa et al., 2016] Sousa, B., Cordeiro, L., Simoes, P., Edmonds, A., Ruiz, S., Carella, G., Corici, M., Nikaein, N., Gomes, A. S., Schiller, E., Braun, T., and Bohnert, T. M. (2016). Towards a fully cloudified mobile network infrastructure. *IEEE Transactions on Network and Service Management*, PP(99):1–1.

[Sousa et al., 2014] Sousa, B., Pentikousis, K., and Curado, M. (2014). Methodical: Towards the next generation of multihomed applications. *Computer Networks*, 65:21–40.

[Stamoulis et al., 1999] Stamoulis, G. D., Kalopsikakis, D., and Kyrikoglou, A. (1999). Efficient agent-based negotiation for telecommunications services. In *Global Telecommunications Conference, 1999. GLOBECOM '99*, volume 3, pages 1989–1996 vol.3.

[Taha and Kamal, 2003] Taha, A.-E. and Kamal, A. (2003). Optimal and near optimal web proxy placement algorithms for networks with planar graph topologies. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 911–914.

[Taleb and Ksentini, 2013a] Taleb, T. and Ksentini, A. (2013a). An analytical model for follow me cloud. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 1291–1296.

[Taleb and Ksentini, 2013b] Taleb, T. and Ksentini, A. (2013b). Follow me cloud: interworking federated clouds and distributed mobile networks. *Network, IEEE*, 27(5):12–19.

[Thouin and Coates, 2007] Thouin, F. and Coates, M. (2007). Video-on-demand networks: Design approaches and future challenges. *Network, IEEE*, 21(2):42–48.

[Tran and Tavanapong, 2005] Tran, M. and Tavanapong, W. (2005). Peers-assisted dynamic content distribution networks. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 123–131.

[Tran and Boukhatem, 2008] Tran, P. N. and Boukhatem, N. (2008). The distance to the ideal alternative (dia) algorithm for interface selection in heterogeneous wireless networs. In *Proc. MobiWac '08*, pages 61–68.

[TubeMogul, 2009] TubeMogul (2009). Half of youtube videos get fewer than 500 views. `http://www.businessinsider.com/chart-of-the-day-youtube-videos-by-views-2009-5?IR=T`.

[Vakali and Pallis, 2003] Vakali, A. and Pallis, G. (2003). Content delivery networks: status and trends. *Internet Computing, IEEE*, 7(6):68–74.

[Vasilakos et al., 2012a] Vasilakos, X., Siris, V. A., Polyzos, G. C., and Pomonis, M. (2012a). Proactive selective neighbor caching for enhancing mobility support in information-centric networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pages 61–66, New York, NY, USA. ACM.

[Vasilakos et al., 2012b] Vasilakos, X., Siris, V. A., Polyzos, G. C., and Pomonis, M. (2012b). Proactive selective neighbor caching for enhancing mobility support in information-centric networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pages 61–66, New York, NY, USA. ACM.

[Wang et al., 2014a] Wang, J., Ren, J., Lu, K., Wang, J., Liu, S., and Westphal, C. (2014a). An optimal cache management framework for information-centric networks with network coding. In *Networking Conference, 2014 IFIP*, pages 1–9.

[Wang et al., 2014b] Wang, X., Chen, M., Han, Z., Wu, D., and Kwon, T. (2014b). Toss: Traffic offloading by social network service-based opportunistic sharing in mobile social networks. In *INFOCOM, 2014 Proceedings IEEE*, pages 2346–2354.

[Williams et al., 2005] Williams, A., Arlitt, M., Williamson, C., and Barker, K. (2005). Web workload characterization: Ten years later. In Tang, X., Xu, J., and Chanson, S. T., editors, *Web Content Delivery*, volume 2 of *Web Information Systems Engineering and Internet Technologies Book Series*, pages 3–21. Springer US.

[Wittie et al., 2010] Wittie, M. P., Pejovic, V., Deek, L., Almeroth, K. C., and Zhao, B. Y. (2010). Exploiting locality of interest in online social networks. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 25:1–25:12, New York, NY, USA. ACM.

[Wu et al., 2001] Wu, D., Hou, Y., Zhu, W., Zhang, Y.-Q., and Peha, J. (2001). Streaming video over the internet: approaches and directions. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(3):282–300.

[Wu et al., 2013] Wu, Q., Li, Z., and Xie, G. (2013). Codingcache: Multipath-aware ccn cache with network coding. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 41–42, New York, NY, USA. ACM.

[Xu, 2012] Xu, D.-L. (2012). An introduction and survey of the evidential reasoning approach for multiple criteria decision analysis. *Annals of Operations Research*, 195(1):163–187.

[Xylomenos et al., 2014] Xylomenos, G., Ververidis, C., Siris, V., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K., and Polyzos, G. (2014). A survey of information-centric networking research. *Communications Surveys Tutorials, IEEE*, 16(2):1024–1049.

[Yang and Chi, 2007] Yang, F.-H. and Chi, C.-H. (2007). Using hierarchical scheme and caching techniques for content distribution networks. In *Semantics, Knowledge and Grid, Third International Conference on*, pages 535–538.

[Yousaf et al., 2013] Yousaf, F., Liebsch, M., Maeder, A., and Schmid, S. (2013). Mobile cdn enhancements for qoe-improved content delivery in mobile operator networks. *Network, IEEE*, 27(2):14–21.

[Yu et al., 2011] Yu, T., Tian, C., Jiang, H., and Liu, W. (2011). Measurements and analysis of an unconstrained user generated content system. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5.

[Yuan et al., 2012] Yuan, H., Song, T., and Crowley, P. (2012). Scalable ndn forwarding: Concepts, issues and principles. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–9.

[Zhang et al., 2015] Zhang, G., Li, H., Zhang, T., Li, D., and Xu, L. (2015). A multi-path forwarding strategy for content-centric networking. In *2015 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 1–6.

[Zhu et al., 2013] Zhu, J., He, J., Zhou, H., and Zhao, B. (2013). Epcache: In-network video caching for lte core networks. In *Wireless Communications Signal Processing (WCSP), 2013 International Conference on*, pages 1–6.

List of Publications

Refereed Papers (Journals, Conferences, Workshops)

- A. F. Antonescu, A. Gomes, P. Robinson and T. Braun, "SLA-driven predictive orchestration for distributed cloud-based mobile services," 2013 IEEE International Conference on Communications Workshops (ICC), Budapest, Hungary, 2013.
  CONFERENCE CORE RANK: B
  ACCEPTANCE RATE: 39%
- 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," Advances in Computing, Communications and Informatics (ICACCI), 2014 International Conference on, New Delhi, India, 2014, pp. 651-657.
  CONFERENCE CORE RANK: N/A
  ACCEPTANCE RATE: 30%
- L. S. Ferreira, D. Pichon, A. Hatefi, A. Gomes, D. Dimitrova, T. Braun, G. Karagiannis, M. Karimzadeh, M. Branco, and L. M. Correia, "An architecture to offer cloud-based radio access network as a service," Networks and Communications (EuCNC), 2014 European Conference on, Bologna, Italy, 2014, pp. 1-5.
  CONFERENCE CORE RANK: N/A
  ACCEPTANCE RATE: UNKNOWN
- G. Karagiannis, A. Jamakovic, A. Edmonds, C. Parada, T. Metsch, D. Pichon, M. Corici, S. Ruffino, A. Gomes, P. S. Crosta, and T. M. Bohnert, "Mobile Cloud Networking: Virtualisation of cellular networks," Telecommunications (ICT), 2014 21st International Conference on, Lisbon, Portugal, 2014, pp. 410-415.
  CONFERENCE CORE RANK: N/A
  ACCEPTANCE RATE: UNKNOWN
- A. Gomes and T. Braun, "Feasibility of Information-Centric Networking Integration into LTE Mobile Networks", 30th Annual ACM Symposium on Applied Computing (SAC ’15), Salamanca, Spain, 2015, pp. 627-633.
  CONFERENCE CORE RANK: B
  ACCEPTANCE RATE: 24%

- I. Alyafawi, E. Schiller, T. Braun, D. Dimitrova, A. Gomes and N. Nikaein, "Critical issues of centralized and cloudified LTE-FDD Radio Access Networks," 2015 IEEE International Conference on Communications (ICC), London, UK, 2015, pp. 5523-5528.
  CONFERENCE CORE RANK: B
  ACCEPTANCE RATE: 38%
- A. Gomes and T. Braun, "Load balancing in LTE mobile networks with Information-Centric Networking," 2015 IEEE International Conference on Communication Workshop (ICCW), London, UK, 2015, pp. 1845-1851.
  CONFERENCE CORE RANK: B
  ACCEPTANCE RATE: 38%
- C. Anastasiades, A. Gomes, R. Gadow and T. Braun, "Persistent caching in information-centric networks," Local Computer Networks (LCN), 2015 IEEE 40th Conference on, Clearwater Beach, FL, USA, 2015, pp. 64-72.
  CONFERENCE CORE RANK: A
  ACCEPTANCE RATE: 30%
- A. S. Gomes, V. Fonseca, B. Sousa, D. Palma, P. Simoes, E. Monteiro, and L. Cordeiro, “A Mobile Follow-Me Cloud Content Caching Model,” IEEE/IFIP Network Operations and Management Symposium (NOMS 2016) – Short Paper, Istanbul, Turkey, 2016, pp. 763-766.
  CONFERENCE CORE RANK: B
  OVERALL ACCEPTANCE RATE: 25%
- A. S. Gomes, T. Braun, and E. Monteiro, “Enhanced Caching Strategies At the Edge of LTE Mobile Networks,” IFIP Networking 2016 Conference (Networking 2016), Vienna, Austria, 2016, pp. 341-349.
  CONFERENCE CORE RANK: A
  ACCEPTANCE RATE: 28%
- A. S. Gomes, B. Sousa, D. Palma, V. Fonseca, Z. Zhao, E. Monteiro, T. Braun, P. Simoes, and L. Cordeiro, “Edge Caching with Mobility Prediction in Virtualized LTE Mobile Networks,” in Future Generation Computer Systems, Elsevier, 2016.
  IMPACT FACTOR: 2.430
- B. Sousa, L. Cordeiro, P. Simões, A. Edmonds, S. Ruiz, G. Carella, M. Corici, N. Nikaen, A. S. Gomes, E. Schiller, T. Braun, and T. M. Bohnert, “Towards a Fully Cloudified Mobile Network Infrastructure,” in IEEE Transactions on Network and Service Management, IEEE, 2016.
  IMPACT FACTOR: 2.920

Unrefereed Works (Unpublished Papers, Technical Reports)

- D. Dimitrova, L. S. Ferreira, A. Gomes, N. Nikaein, A. Georgiev, and A. Pizzinat, “Challenges ahead of RAN virtualization in LTE,” in European

Conference on Networks and Communications 2014 (EuCNC 2014) - Unpublished Workshop Paper, Bologna, Italy, 2014.

- A. Gomes, S. Ruiz, G. Carella, P. Comi, and P. S. Crosta, "Cloud-based Orchestration of Multimedia Services and Applications," in European Conference on Networks and Communications 2014 (EuCNC 2014) - Unpublished Workshop Paper, Bologna, Italy, 2014.
- I. Alyafawi, E. Schiller, T. Braun, D. Dimitrova, A. Gomes, and N. Nikaein, "Critical Issues of Centralized and Cloudified LTE FDD Radio Access Networks", (Technical Report IAM-14-002). Bern, Switzerland: Institut für Informatik und angewandte Mathematik, Universität Bern.
- C. Anastasiades, A. Gomes, R. Gadow, and T. Braun, "Persistent Caching in Information-Centric Networking", (Technical Report IAM-15-001). Bern, Switzerland: Institut fur Informatik, Universität Bern.

Mobile Cloud Networking Project Deliverables

- Deliverable 2.1: Reference Scenarios and Technical System Requirements Definition (April 1st, 2013).
- Deliverable 2.2: Overall Architecture Definition, Release 1 (October 30th, 2013).
- Deliverable 2.3: Market Analysis and Impact of Mobile Cloud Concepts (November 8th, 2013).
- Deliverable 2.5: Final Overall Architecture Definition, Release 2 (April 30th, 2015).
- Deliverable 3.1: Infrastructure Management Foundations – Specifications & Design for MobileCloud framework (November 8th, 2013).
- Deliverable 3.2: Infrastructure Management Foundations – Components First Release (April 29th, 2014).
- Deliverable 3.3: Infrastructure Management Foundations – Components Final Release (December 23th, 2014).
- Deliverable 3.4: Infrastructure Management Foundations – Final Report on component design and implementation (June 30th, 2015).
- Deliverable 4.3: Algorithms and Mechanisms for the Mobile Network Cloud (November 7th, 2014).
- Deliverable 5.1: Design of Mobile Platform Architecture, IMSaaS and DSN applications (November 8th, 2013).
- Deliverable 5.2: First Implementation of IMSaaS, DSN and Mobile Platform (May 8th, 2014).
- Deliverable 5.3: Final Implementation of IMSaaS, DSN, and Mobile Platform (January 17th, 2015).
- Deliverable 5.4: Evaluation of Mobile Platform, IMSaaS and DSN (June

30th, 2015).

- Deliverable 6.2: Initial Report on Testbeds, Experimentation, and Evaluation (October 31st, 2014).
- Deliverable 6.4: Final Report on Testbeds, Experimentation, and Evaluation, Demonstration (November 30th, 2015).
- Deliverable 7.1: Web Site (November 8th, 2013)