Master's Degree in Informatics Engineering
Dissertation
Final Report

Smart Content Relocation in Content-Centric Networks

Vitor Alves Fonseca
fonsecav@student.dei.uc.pt

Advisors:
Prof. Paulo Simões
Eng. André Gomes

Date: 06 of July of 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Master’s Degree in Informatics Engineering
Dissertation
Final Report

Smart Content Relocation in Content-Centric Networks

Vitor Alves Fonseca
fonsecav@student.dei.uc.pt

Advisors:
Prof. Paulo Simões
Eng. André Gomes

Jury:
Prof. João Vilela
Prof. Joel Arrais

Date: 06 of July of 2015

Abstract

Information-Centric Networking (ICN) is a new networking concept devised to cope with the challenges faced by the current design of Internet communications, such as the rising number of users and the mobility of the devices. Although this concept already has some features that are relevant for current Internet usage, such as in-network caching capabilities and content replication, there are still some relevant improvements to be developed.

One of the key aspects of ICN is the ability to deal with node mobility, since there are no persistent connections, making it stand out from the current Internet design. However, although ICN supports mobility, there are still no optimizations to better serve users on the move. By tracking the contents' popularity at the users' sources, it is possible to pre-cache related contents to the ones accessed by the users at their destination. This pre-caching allows to better serve those users when they arrive, providing reduced delays while obtaining the pre-cached contents.

Content-Centric Networking (CCN) is one of the existent ICN architectures and here we propose an improvement to CCN mobility capabilities: smart relocation of contents, based on users' movement. In this work, we design, implement and validate smart content relocation mechanisms able to integrate with the core system of CCNx to anonymously monitor data requests and relocate relevant data according to the users' upcoming destination prior to their arrival. This will allow an improvement on the users' service experience, namely regarding delay, when using CCN with content relocation.

Keywords

CCNx, Content Relocation, Content-Centric Networking, Information-Centric Networking, Mobile-Cloud Networking, Monitoring, Multiple-Attribute Decision-Mechanisms

Resumo

Information-Centric Networking (ICN) é um novo conceito de redes criado com o objectivo de ultrapassar alguns dos obstáculos encontrados pela actual forma de comunicação da Internet, tais como o crescente número de utilizadores e a mobilidade dos dispositivos. A possibilidade de fazer *caching* por toda a rede e de replicar conteúdos são duas das características base deste novo conceito, que tem ainda muitos aspectos onde pode ser melhorado.

No ICN não existem ligações permanentes entre dois nós, ao contrário do actual conceito da Internet, o que faz com que este consiga lidar bem com a mobilidade dos nós, um dos principais aspectos que se salienta comparando à actualidade. No entanto não foram ainda desenvolvidas nenhum tipo de optimizações para melhor responder aos utilizadores que estão em movimento. Através da monitorização dos conteúdos populares entre os utilizadores, é possível fazer um *pre-caching* quando estes se movem, para assim melhor a sua experiência quando chegam ao destino, nomeadamente apresentando atrasos menores.

Content-Centric Networking (CCN) é uma das várias arquitecturas de ICN existentes à qual é proposta uma melhoria: relocalização inteligente de conteúdos com base no deslocamento dos utilizadores. Neste trabalho desenhamos, implementamos e validamos esta proposta e a sua integração com o sistema base do CCNx através da monitorização anónima dos conteúdos e da migração dos conteúdos relevantes para o destino dos utilizadores. Esta migração permitirá uma melhoria na experiência dos utilizadores, nomeadamente em termos de atraso quando usam CCN com a relocalização dos conteúdos.

Palavras-chave

CCNx, Content-Centric Networking, Information-Centric Networking, Mecanismos de Decisão Multi-Objectivo, Mobile-Cloud Networking, Monitorização, Relocalização de Conteúdos

Table of Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
CCN	Content-Centric Networking
CDN	Content Distribution Networks
CS	Content Store
DCE	Direct Code Execution
DONA	Data-Oriented Network Architecture
FIB	Forwarding Information Base
FMC	Follow-Me Cloud
ICN	Information-Centric Networking
ICNaaS	ICN as a Service
IoT	Internet of Things
IP	Internet Protocol
LRU	Least Recently Used
LTE	Long Term Evolution
MADM	Multiple Attribute Decision Making
MCN	Mobile Cloud Networking
MIP	Mobile IP
MOBaaS	Mobility and Bandwidth Availability Prediction as a Service
NDO	Named Data Objects
NetInf	Network of Information
NRS	Name Resolution Service
PARC	Palo Alto Research Center
PIT	Pending Interest Table
PKI	Public Key Infrastructure
PMIP	Proxy Mobile IP
PSIRP	Publish-Subscribe Internet Routing Paradigm
PURSUIT	Publish-Subscribe Internet Technology
RWP	Random Waypoint
SAIL	Scalable and Adaptive Internet
SCORE	Smart COntent RElocation
TCP	Transmission Control Protocol

1 Introduction

Nowadays, the Internet has been facing different problems and challenges such as the continuous increase of networking-capable devices, as well as the growth in Internet traffic [1], motivating a redefinition of the Internet concept and the coining of the Internet of Things (IoT).

The Internet was originally designed so that a few distant devices could share information between them, host-to-host. Protocols and respective architectures were initially created based on this assumption. The communication between a source and a destination hosts relies on addresses specified according to the Internet Protocol (IP) [2]. In this case, IP addresses assume the role of locators and identifiers, which represents an issue given the limits of IPv4 addresses. Indeed, the explosion of mobile devices connected to the Internet accelerated the use of these identifiers surpassing the maximum number of identifiers supported by IPv4. At this point, a new type of host identifier, IPv6 [3], was developed and protocols were modified to work simultaneously with both mechanisms, to avoid disruptions on existing Internet services. Nonetheless, IPv4 has mostly prevailed, and networking remains mostly as host-centric. This concept however, is not well aligned with nowadays Internet usage, where many users access the same information in a close period of time and location and where scalability is a constant issue. All these problems motivate the existence of content replication, since it allows decreasing the usage of core-network bandwidth and reducing delays while requesting popular data. Some mechanisms were designed to integrate content replication in the current Internet, such as Content Distribution Networks (CDN) [4] or Peer-to-Peer Networks [5].

Another problem faced nowadays is the growth of mobile devices and their usage while travelling. This raises challenges such as passing through multiple cells when using 3G or 4G network technologies such as Long Term Evolution (LTE) [6] – for instance while travelling by train or car. To cope with this kind of usage, some protocols for mobile IP were proposed, such as Mobile IP (MIP) [7] or Proxy Mobile IP (PMIP) [8]. However, these protocols only support addressing and route optimization. The need for this kind of mechanisms comes from the fact that some existing transport protocols, for example Transmission Control Protocol (TCP) [9], were developed for fixed nodes and use the IP address to keep the session state during the communication. If one of the nodes would move during the communication, the same IP had to be kept otherwise the communication would have to be restarted.

With all these problems and requirements in mind, a new Internet concept was proposed, focused on the *what* (the content) instead of the *where* (the hosts). Since a new concept was being designed, the objective was to solve most of the problems faced nowadays allowing for in-network caching and content replication without losing the focus on the security of the data. Information-Centric Networking (ICN) [10] was then proposed as a broad concept describing in general all the features that should be implemented as a new Internet Protocol.

ICN was designed with the purpose of solving two of the problems identified in the current Internet design, the mobility and the scalability issues. However, there are still some improvements that can be made and features that can be added being the relocation of contents based on users mobility one of them.

Since one of the key aspects of ICN is the in-network caching capability, the most accessed contents from a certain location will typically be available in nearby routers, resulting mainly into local traffic data requests. When users move to a different location however, it is likely that the desired contents are located far away from the users' new location, meaning that when the contents are requested again by the user, there will be an increased delay as well as an increase in the core-network usage.

This work proposes the Smart COntent REIocation (SCORE) mechanism, with the aim of reducing the impact in the user's perceived experience, in particular when moving from one destination to another with the help of mobility prediction mechanisms. The mechanism will monitor the traffic history, creating a complete view of what is requested at each location. In order to avoid raising privacy concerns, all the monitoring is done anonymously, since the SCORE mechanism will record what is requested at each location, but does not record who requested it. Mobility prediction is out of the scope of this work, being an external entity that will integrate with the work herein developed.

When a relocation of contents is triggered, the SCORE mechanism will check what should be migrated according to the source location of the users moving, and populate the available storages near the destination location while considering that the users already present at the destination should not be neglected.

SCORE, introduced in this document, will not only be validated in this dissertation but also in the context of Mobile Cloud Networking (MCN), a European Union FP7 Large-scale Integrating Project funded by the European Commission [11].

An analysis of the current state of the art on ICN is provided in Section 2, presenting the most important aspects that support this work, followed by the presentation of the established objectives and how they will be tackled, in Section 3. The work plan defined in the beginning of the dissertation is presented in Section 4 while also explaining a revision to that plan and its final implementation. Section 5 presents all the developed work during both semesters and Section 6 describes the performed evaluation, as well as the analysis of the obtained results. Finally, Section 7 concludes this document by presenting a final overlook on the contributions from this work.

2 State of the Art

This section will describe related works that support the activities to be conducted during this dissertation, starting with the presentation of what ICN [12] is. Some of the different approaches available in literature will then be analysed, specifically the Data-Oriented Network Architecture (DONA) [13], Content-Centric Networking (CCN) [14], Publish-Subscribe Internet Routing Paradigm (PSIRP) [15] and Network of Information (NetInf) [16]. Finally, the different types of relocation under evaluation will be reviewed, as well as existing decision and mobility prediction mechanisms.

2.1 Information-centric networking (ICN)

The current Internet concept has it focus on the hosts. Information-centric networking, instead, is a new concept focused on the contents (information) aiming to better satisfy the current Internet challenges where more and more contents and services are shared and used by many users at the same time and sometimes in nearby locations. Since ICN is a fairly recent concept that is still under definition, there are only a few proposed elements that are implemented by the available architectures under development and that are considered essential. These elements are Named Data Objects (NDO), Naming and Security, Application Programming Interface (API), Routing and Forwarding, and Caching [17] and are detailed in the following paragraphs.

The main idea behind this concept is that despite the connection or the hosts being unknown, and therefore untrusted, any content coming from them can be trusted providing that the signer of that content is trustable. Therefore, looking at Figure 1, when a user requests content that is, for instance, stored in the cache of Router 2, it can be retrieved from that location since the content itself is a trustable copy of the desired object, in spite of Router 1 and Router 2 being untrusted, as well as the connection between them. This way, content can be fetched from the nearest location without the need to establish a trusted connection.

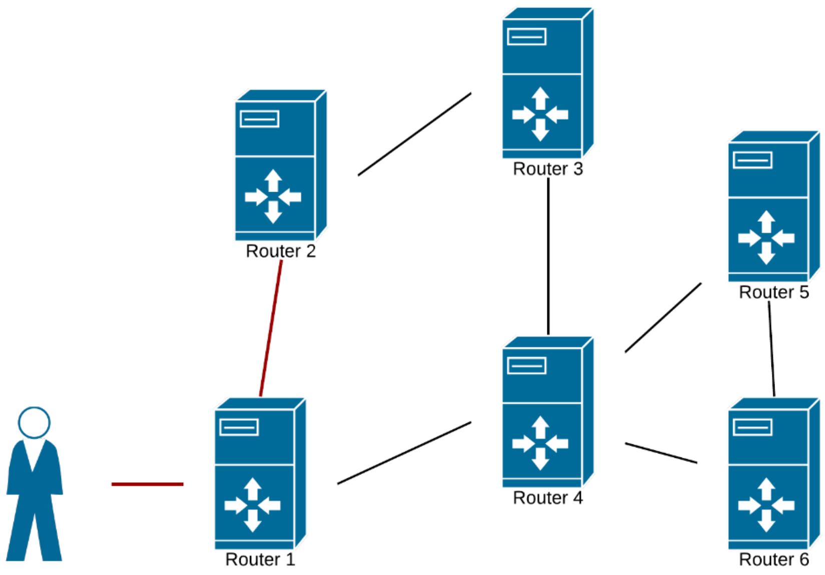


Figure 1 – ICN model

Before explaining each of the elements proposed for ICN architectures, it is important to note some relevant nomenclature aspects. For instance, contrary to the current design where the source is where the packet came from, in ICN, the source, also known as producer or publisher, depending on the approach, is always the provider of the content whereas the consumers, or subscribers, are the ones requesting said content. According to the considered approach, besides the temporary caches, the content can come either from the producers themselves or from network storages. These storages are repository applications that are part of the ICN implemented architecture.

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 and accessible to any user, such as movies, web pages, documents or even live streams. With this abstraction, every object present on the Internet, either persistent such as a photo, or temporary such as a stream, is identified entirely by its name, regardless of its location and how it was transmitted.

Naming and Security are intimately related to the NDO abstraction and are the centre of the ICN concept, which means that they are as important to ICN as the host identification is to the current Internet IP-based concept. Two important aspects have to be considered, uniquely identify each object and ensure that each object can be trusted. The need to know that each object can be trusted comes from the fact that an object can come from an unknown and/or untrusted router. To address these points each NDO, which can be either a complete file or a chunk, depending on the used approach, has a unique name – more commonly referred as prefix and certification. Two schemes are proposed to achieve this certification, one with a hierarchical approach and another with a flat namespace. The second one is a self-certifying scheme where each NDO can be verified without the usage of a third-party entity to ensure the trustfulness of a key. This self-certification is done by hashing the content of each object and embedding the hash in the name of each NDO resulting in names that typically are non-hierarchical and not human readable. The first approach, which uses a hierarchical scheme, uses a structure similar to the URLs used nowadays, which enables the possibility for aggregation of routing information. This also means that filenames can be easily identified by users, allowing them to manually type names and understand how files are positioned in the publisher hierarchy. This approach, however, requires a third-party entity to validate the keys shipped with each NDO, which can be considered as a downside.

In the majority of the approaches, the content has to be published by the producer, by making it available in the network, and only after that, can it be requested by the consumer using the NDO name in a synchronous operation. Other approaches, such as PSIRP, use a different approach where the consumer firstly registers a request for a particular NDO, being later notified whenever that content is available. To make this possible for any type of application developed or upgraded to work with ICN, it is necessary to have a very well defined API that allows any application to publish or request objects from the Internet.

The routing and forwarding features provided by ICN have also two different approaches, dependent of the used naming scheme, where names can either be aggregated or kept individually. One existing approach uses a Name Resolution Service (NRS), which keeps the location of storages in the network as well as the object's names they contain. When a consumer sends a request, the NRS resolves the name to a source, routes that request to the available source(s) that will then send the content back to the consumer. The alternative approach does not need an extra element to translate names into sources, it directly routes the requests from the consumer to the available sources in the network and when a source receives a request, the content is routed back to the consumer. In both approaches, there are different routing algorithms that can be used, depending mostly on the chosen approach and in the naming scheme being used.

Finally, the caching aspect is also one of the important key features of the ICN concept. ICN proposes that every node can have a cache whether they are in operator-run infrastructures, user-run networks, or even mobile devices. With this approach, ICN combines caching at the network edge with in-network caching allowing that any node holding a copy of the desired content can satisfy existing requests. This cache is also application independent, which means it can hold any content from any source. Typically routers store in their cache all the contents that go through them, applying a Least Recently Used (LRU) replacement policy.

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. Four different approaches will be presented, as well as their chosen solutions for handling the most relevant topics of ICN.

2.1.1 Data-Oriented Network Architecture (DONA)

In DONA's approach, the producers register the data that they can serve into the resolution infrastructure, a set of resolution handler nodes. When a consumer wants an object, it sends a request, a *Find* packet, with the name of the object to the network, which is routed throughout the resolution handlers until it reaches the source or a node with the object in cache. When the desired object is located, it can either be routed back to the consumer through the same reverse path or through a more direct route. However, if the content is sent through the direct route, the caching capability of ICN is severely affected since the content will not pass through the middle caches.

The registry of objects has associated expiry times, which means that content providers have to periodically renew the content registration. However, content providers can also register their prefix instead of every object that they possess.

2.1.2 Content-Centric Networking (CCN)

In the CCN implementation, all nodes are able to publish NDOs, either by having their own repository or by publishing to other repositories available in the network. The used routing protocols ensure that it is possible for any node connected to the network to locate the published NDOs. Since CCN uses a hierarchical naming scheme, the routing protocols are also able to perform aggregation. Regarding security, CCN uses a scheme of public key cryptography where trust in keys can be achieved by a mechanism similar to a Public Key Infrastructure (PKI). When a consumer wants an object, it sends a request (an *Interest* packet) that is resent by every router until the request lifetime ends or until a copy of the content is reached, either by reaching the source of the content or router with a copy in its cache, the Content Store (CS). In order to send the packets, each router has a Forwarding Information Base (FIB) that the routers use to send the packets in unicast or multicast based on the NDO's prefix. When a router as a local repository, it has a FIB entry with an higher priority than the priority of other routers. The data is then routed back through the reverse path populating the cache of every router it passes through. CCN routers have a Pending Interest Table (PIT) where the forwarded requests are stored as well as the origin of that request. If the request for a content that has already been forwarded is received from another source, the new source is added in the PIT entry without sending a new request. When the expected content reaches a CCN router, the corresponding PIT entry is analysed and if there are pending requests, the content is forwarded to all the requesters. Figure 2 illustrates the processing done at every router when an Interest message is received while Figure 3 shows the process flow when a Content message arrives at a router.

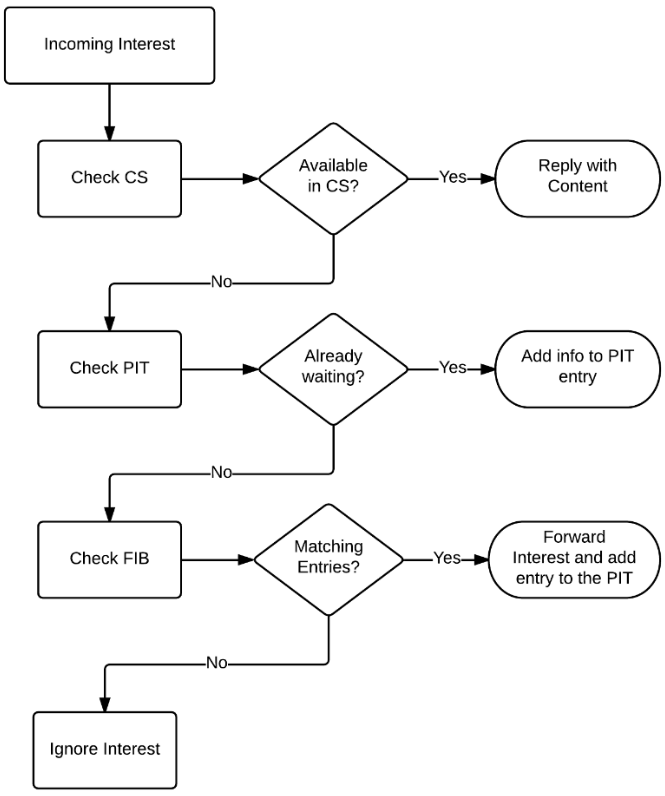


Figure 2 – Interest processing flowchart

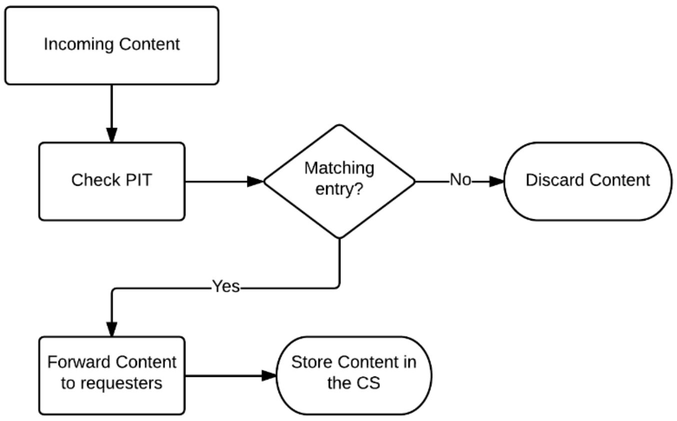


Figure 3 – Content processing flowchart

CCN allows the usage of multiple strategies for the forwarding of requests based on topology aspects or even in the observed network performance. This makes possible the existence of load balancing at every router, which may result in a more balanced use of the network aiming at better serving all the users. As well as the ICN concept itself, CCN is still under development and specification.

2.1.3 Publish-Subscribe Internet Routing Paradigm (PSIRP)

In PSIRP, the 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 a consumer wants to get an object, it sends a subscription that is later matched to a content object. The *rendezvous* system will then notify the producer to send the data to the consumer that requested it. Since it is this system that matches the subscriptions with the publish contents, it is possible to subscribe content that is not still available since the *rendezvous* system can store the subscription until there is available content. If the content becomes available before the subscription expires, the content will be sent to the subscriber, otherwise, the subscription will be cleared and the subscriber will have to send another one.

2.1.4 Network of Information (NetInf)

NetInf implements the two approaches for routing described by ICN, the name resolution service and the name-based routing, in a hybrid system, which allows taking benefit from both approaches with the selection of the "best" available source while allowing for the aggregation of the routing information. When a producer wants to make content available it can either register the NDO in the Name Resolution Service (NRS) or use a routing protocol to announce the new routing information, if necessary. Even if the producer does not register a new entry in the NRS, any other router with that NDO in its cache can register it in order to make both mechanisms available for the consumers. A consumer can then use a NRS, if available, to resolve a name or just send a request, a *Get* packet, that is transmitted in a name-based routing approach by every router until it reaches a router with the NDO in cache or the NDO producer.

This solution, when using the NRS approach has the same downside as DONA where the direct communication between the producer and the consumer means that there will not be any intermediate caches being populated with those NDOs.

2.2 Relocation Mechanisms

Presently, relocation of content according to user movement can either be the relocation of services, such as the Follow-Me Cloud (FMC) Concept [18], or the relocation of Content Delivery Network (CDN) serving points, as proposed by M. Liebsch *et al.* [19] where what is relocated are also machines, not only the content.

FMC is a mechanism developed for the migration of services in TCP/IP networks from datacentre to datacentre, which assumes that there are cloud-enabled datacentres available at the edge of the network between which the services can be migrated. FMC controllers are distributed in the network and modify the packet forwarding mechanisms. By doing so, it is possible to manage the network infrastructure without affecting the communication between the client and the server. Then, triggered by user mobility, the FMC control mechanism will decide if it should migrate the service and where it should migrate it to, leveraging the costs of the migration and the benefits brought both to the client and to the network.

Content Delivery Networks consist of distributed servers that aim to provide in-network caching capabilities across the Internet. However, these caches do not work as web caches where what is stored are the latest objects to pass through it. CDNs are contracted by content/service providers to host and deliver their content, which brings benefits both to users, through higher availability and performance, and to the content providers by reducing the amount of clients served directly from their servers, reducing the amount of processing power necessary and thus reducing the costs. CDN owners pay network operators to host their servers, aiming to have multiple serving points near the users. However, these servers are still located at the network operators' facilities meaning they are still away from the edge of the network. Unlike ICN routers, all CDN serving points will have similar contents since it is a contracted service, which means that when users move, although it can optimize the connection by changing the connection to another serving point, there will be no migration of contents according to the users' preferences.

2.3 Decision Mechanisms

When it comes to decision mechanisms, they can be either relatively simple or extremely complex, depending on their application. This complexity is impacted by the factors or inputs to be considered and the complexity of the decisions to be made. For instance, when there are many inputs with few possible values or even few inputs but with many possible values, there are a lot of possible combinations between these that will translate in the final decision. If the possible outcomes are also just a few, it means that many combinations will translate in the same result which means that the decision making process can be optimized and simplified. However, if there are also a lot of possible outcomes, the process can be very complex since a small change in one value might translate in a complete different outcome for the decision making process.

The analysis of big amounts of data can be done in various ways depending on how the result of that analysis will be used. Data mining is one process of analysing data that can itself be divided into multiple types according to its expected use. Classification and regression, such as decision trees and neural networks, for example, are two classes of data mining. This type of mechanism, however, relies on a very well defined number of parameters and values. But in some cases, the inputs for the decision making process are not so well defined and affect each other when being computed, since in some cases there is not a single correct decision, but rather a couple of fit decisions.

Multiple-Criteria Decision-Making (MCDM) [20] is very important in situations where there are many attributes to be taken into account and where some of these attributes should be maximized and others should be minimized. The decision mechanism specified by B. Sousa *et al.* [21], MeTHODICAL, belongs to this type of mechanisms, and it is essentially a scoring mechanism that determines a score for each entry by comparing the values of each attribute and the ideal one, then outputting the list of scores. TOPSIS [22] and DiA [23] are two different algorithms that aim at achieving the same results. Although it might be easier to solve possible problems by talking directly to the author of the MeTHODICAL algorithm, a comparison of the performance of the three algorithms in this application will be made in order to choose the best one.

A list of scores, however, is not exactly a decision but rather a step to get closer to it, so in cases like this, the decision making process is composed of multiple phases where different mechanisms are used in a complementary way. By using a MCDM, one can have then a set of very well defined parameters and values, that can be used by another decision mechanism, such as a decision tree, for example, that will indeed output a decision based on all the known information, reflecting the information previously acquired by training and/or self-learning.

3 Research Objectives and Approach Method

The research objectives and the methodology used for this work will be described in this section. First, the objectives will be fully explained, followed by a detailed description of the planned tasks to achieve each one.

Before detailing the objectives, and for better understanding some of them, it is important to specify in advance some of the chosen approaches, although these will be better explained and justified in section 3.2. The ICN implementation used during this project will be CCN and the content migration will be based on popularity.

3.1 Research Objectives

The main objective of this research, as the dissertation title suggests, is the development of a smart mechanism to relocate contents in content-centric networking, Smart COntent REllocation (SCORE). SCORE, being developed in the context of the MCN project, will be part of ICN as a Service (ICNaaS), one of the many services included in the project. A broad overview of the MCN project is presented in Appendix A.

In order to achieve the global objective of this dissertation, SCORE, there are smaller goals to be accomplished, such as the selection of contents that should be migrated, the development of a support mechanism to perform the migration, the monitoring of processed contents and the definition of how to trigger the decision-making mechanism. This decision mechanism is the one that will select the contents to relocate, and where to, based on the movement of the users.

The defined objectives, and related tasks that compose them, were analysed in detail as an initial planning phase, in order to better understand their implications throughout the development of the project. This subsection presents this preliminary analysis.

3.1.1 Relocation Tools

Since ICN, and in particular CCN, can support two types of data storing methods, one volatile, resorting to the caches of nodes, and one "permanent" in repositories, two different approaches for the relocation (migration) of content have to be implemented.

3.1.1.1 Cache Relocation

In this process, the content does not need to actually be migrated. Instead, the content has only to be placed at the destination cache and, considering that every node with a cache stores the content that passes through it, a simulation of a user request at the destination node will suffice to populate that cache with the desired content. Because of how the protocol was designed, the intermediate caches between the destination and the nearest source will also be populated with that content.

3.1.1.2 Repository Relocation

This is a more complex process than the previous one and the relocation/migration terms are not exactly appropriated to represent this action. What happens, in reality, is a content replication rather than relocation since the content will remain at the original source too.

This process will be much less used than the cache relocation but it is also necessary due to the existence of different caches and file sizes. Caches, being virtual, will be rather small in most of the cases, especially near the edge of the network where the great majority of the devices will be small. In fact, these are intended to serve a handful of users, such as the ones that operators install in the users' homes, meaning that these routers will have less memory and inherently smaller caches. Because of this, some files may be much larger than the available caches at the desired destination, but that same node, or a near one, might have a repository available where to the file can be copied in order to optimize the user experience and reduce the bandwidth used in the core network.

The implementation of both the Cache and Repository relocation mechanisms is detailed in Section 5.5.

3.1.2 Content Monitoring

Since the relocation will be based on content popularity, there is a need to monitor all the requested contents in order to distinguish between most popular contents and less popular ones, feeding this information to the decision algorithms. Moreover, considering that what might be popular in one area of the network may not be so in another area, implies that monitoring has to be differentiated according to the nodes' location. The opposite is also true, meaning that the same content may be popular in two distinct locations, making it likely for such content to be already present in the cache of nodes close to the destination. Besides the variation in the popularity of content according to the location where the monitoring is being done, since what is popular for some users might not be popular to others, time is also an important factor, as what is popular this month, might not have been popular last month and it is probable that it will not be popular in the next month. Therefore, the monitoring process not only records the content information but also a timestamp.

3.1.2.1 Local Monitoring

Taking into consideration the need for differentiated monitoring, each node has the possibility of monitoring all of the processed contents. However, for privacy and anonymity purposes, end nodes are not expected to perform monitoring since it would solely correspond to the usage of a single host. Consequently, the monitoring of processed content will be considered as an optional feature for each node before its initialization. This means that when starting a router, one might choose whether or not that router will perform the monitoring of the processed contents and send that information to the central database.

The modifications needed to be done in the code of the routers do not interfere in any way with is normal functions, meaning that routers with and without monitoring are able to communicate without any kind of problem.

3.1.2.2 Centralized Monitoring

While differentiated monitoring is a requirement throughout the network, it is also important to have all the collected information stored in a centralized location, allowing the decision mechanism to easily access it whenever needed.

In order to accomplish this, a database will be created whose sole purpose is to store all the monitoring data collected by every node as detailed in Section 5.6. The data will be stored while keeping a relation between the processed content and the router where it was requested, so that the differentiation is not lost when centralizing all the information. This means that besides the monitored data, the database will also contain information regarding the routers available in the network.

3.1.3 SCORE Mechanism

The SCORE mechanism is probably the most important component since it will be the centre of the entire operation. While the previously described elements are intended to gather all the necessary information, or to perform the migration of contents, it is up to the SCORE mechanism to use the gathered information and determine what should be migrated and when, after which it will instruct the migration mechanism about what should be done at specific nodes. In the context of the MCN project, the SCORE mechanism will be triggered by the Mobility and Bandwidth Availability Prediction as a Service (MOBaaS), another being developed in the scope of the MCN project.

During the development of this work, two different specifications of the SCORE mechanism were done since when testing the first approach some problems were detected.

3.1.3.1 SCORE Mechanism first specification

The decision mechanism will be triggered when a user, or group of users, is moving from one location to another, or, in the best-case scenario, it will be informed in advance about users moving from one place to another at a specific time (mobility prediction). With the received information, the mechanism will query the database that holds the monitoring information to retrieve two lists. One of these lists includes the content popularity at the current location of moving users, while the second list contains the content popularity at the users' given destination. The popularity is determined by counting the number of times each content is requested.

The decision about which contents should be migrated does not solely depend on their popularity, there are also other factors that have to be taken into account. Some of these factors are the file size, the cache size, the cost of the migration (measured in time of transfer), the number of users moving and the number of users already at the destination. By determining a ratio that considers both file and cache sizes, as per Equation 1, it is possible to eliminate, a priori, files that are larger than the available caches. It is also important to maximize cache usage, by having as many cached files as possible, being popular files with a lower file/cache ratio the most preferred.

$$file/cache\,ratio = \frac{file\ size}{cache\ size}$$

Equation 1

It is of paramount importance to know contents' information, such as its popularity and size, both at the source and destination of the users, as well as the routers' information, such as the cache size, in order to better determine what can be migrated, what should be migrated and what can be discarded. For instance, if the most popular content at the destination is the same as at the source, then there is no need to migrate that content since it is likely that that content is already at the destination caches. On the other hand, if the popular contents at the destination mismatch the popularity at the source, a compromise between both popular contents has to be reached. For this reason, the number of users moving and the number of users already at the destination have to be taken into account, where a higher number of users will weigh more in the decision process.

Based on the description done above, the metrics to be considered are:

- Number of users moving;
- Number of users at the destination;
- Popularity of contents at the users' source;
- Popularity of contents at the users' destination;
- File sizes;
- File/Cache size ratios;

After deciding which contents are going to be migrated, the mechanism will instruct nodes at the destination to retrieve such contents to their cache and/or repository. This will ensure that the most popular files are already cached near the users' arriving destination, reducing the delay when these objects are accessed.

This was the first specification of the SCORE mechanism, however, after the first run of tests, a couple of question arose, which motivated a completely new analysis of the problem in order to find a different and better suited solution. The major question was how to determine the theoretical solution to compare with the output of the algorithms in order to evaluate if it was working as expected.

Another question appeared when performing a mathematical analysis of the implementation of the MADM algorithms with the specified metrics, which proved that in this case, the usage of both the file size and file/cache size ratio metrics was unnecessary since they are directly related.

3.1.3.2 SCORE Mechanism second specification

At this point of the review of the problem, one major constraint specified from the beginning was also changed, instead of telling the routers what they should migrate, the SCORE mechanism will tell the routers what they should have stored in cache to better serve not only the new arriving users but also the ones that remain at that location. This means that now instead of lowering the priority of a file if it is very popular at the destination, it should be even higher, meaning that the popularity at source and destination should be averaged while also considering the number of users at each location. The other constraint that changed was that instead of considering one source of users and one destination, multiple sources could be considered providing that the users will arrive at the same destination at the same time. Equation 2 depicts the formula that determines the content popularity based on the popularity at each source (popSrc/reqSrc), the popularity at the destination (popDst/reqDst), the users moving from each source (movU) and the number of users that remain at the destination (dstU).

$$popContent_i = \frac{\sum_{k=1}^{N}\left(\frac{popSrc_{i,k}}{reqSrc_k} * movU_k\right) + \frac{popDst_i}{reqDst} * dstU}{N+1}$$

Equation 2

With this revision of the SCORE mechanism, the initial six metrics considered by the MADM algorithms were reduced to only two:

- Content Popularity;
- Content Size.

By analysing the final approach it is possible to map it to a well-known optimisation problem, the 0/1 knapsack problem, where the router's cache can be viewed as the sack, the content size the weight of each object and its popularity viewed as the value of each object. The concept behind both problems is the same, maximize the value considering that there is a size limit in the cache/sack. By following the knapsack algorithm, it is possible to obtain the best theoretical solution for each case, allowing the comparison with the output of the MADM algorithms, solving also the comparison issue found in the first specification. Resorting to the knapsack solution was considered as part of the SCORE mechanism, however achieving an optimal solution for such an optimisation problem is known to be NP-complete. Therefore using knapsack with the SCORE mechanism would not meet the expected real-time performance for content migration. Nonetheless, tests were performed and the SCORE mechanism could be adapted to use a knapsack implementation if it demonstrated high performance under these conditions.

3.1.4 Validation

The validation of all the developed tools and algorithms is one important aspect to consider during the proposal of a new mechanism. Since in the SCORE mechanism there are a lot of small independent tools interacting with each other, it is important to evaluate the correct work of each one individually and also evaluate the mechanism as a whole.

In order to validate the relocation tools, both tools will be triggered manually with a set of defined contents to relocate to a certain destination. After each execution of a tool, the contents present in the destination will be examined to inspect if all and the correct contents were relocated to that location. The validation of the monitoring mechanisms will be done by simulating large quantities of requests of very well defined file sets, which are later compared with the information stored in the database, ensuring, this way, that all the requests were logged.

With every tool validated individually, it is necessary to perform a validation of the entire system working together. To do so, this evaluation will be done in two steps, a validation in a small real test bed and a validation using simulation in order to perform scalability tests.

3.2 Methodology

In order to complete all the objectives described above, it is necessary to make some decisions in advance such as the ICN implementation to be used, and in which metrics will the migration of contents be based on.

3.2.1 ICN Implementation

The ICN implementation chosen was CCN due to many reasons. One of these reasons is the availability of the source code and its support. DONA, for instance, does not have an available implementation. PSIRP was continued in Publish-Subscribe Internet Technology

(PURSUIT) project[1], which in the meantime has already ended. Although the project finished with a prototype completed, the support and further development has stopped. Similar to PSIRP, NetInf was continued in the Scalable and Adaptive Internet Solutions (SAIL) project that also already ended. There are still some new developments being made in NetInf, however, the available support is limited and there is no active community.

CCN, more particularly the CCNx project, is an implementation by the Palo Alto Research Center (PARC) that is still active and under development with its new version, CCNx 1.0, released in 24 of June of 2015 under a commercial and an institutional evaluation licenses. However, until CCNx 0.8.2, the version used in this work, CCNx is an open-source framework. Besides having a team dedicated to the further development of CCNx, and to provide the needed support to the community using CCNx, there is also a great public community around the project exchanging ideas and experiments. The implementation is also very well documented and detailed having many examples and tutorials to help better understand the ICN concept and how it works in reality. This active development and support is of extreme importance since during the development of this work some changes in the base code of CCNx might be necessary.

3.2.2 Content migration

Regarding the decision about the contents to migrate, since ICN emphasizes that the focus should be on contents and not on hosts, in the ICN concept there is no single user identification. Based on this premise, the contents could not be migrated based directly on which user was moving, so, to solve this, the contents to take into account will be the most popular ones at the user location. Triggering migrations based on a single user movement would also be quite challenging both to the decision mechanism and to the network, so the migration will focus on group mobility rather than single user mobility. This also improves the probability of moving important contents because if it were only a single user moving, there was a great chance that the popular contents at that location would be completely different from the user's interest.

3.2.3 System Validation

The validation of the entire system is important to assess if it is working correctly, but also important is to see if this contribution actually brings benefits to both users and content providers. This is achieved by doing a comparison between the performance of the unmodified CCNx architecture and the architecture proposed in this work.

To perform a comparison between these two architectures it is necessary to detail specific scenarios where both architectures can be evaluated under the exact same conditions and it is also necessary to specify the metrics that will be used to perform the comparison. The metrics used can be the number of cache hits and cache misses that represent, respectively, if a router when receives a request can reply to it or has to forward the requests, but also the delay to obtain a certain object, for example.

[1] www.fp7-pursuit.eu

4 Work Plan and Implications

The initial planning of the second semester that was presented in the intermediate report is revisited in this chapter, detailing the evolution and necessary adjustments of the work conducted during both the first and second semesters.

4.1 Initial planning of the second semester

The second semester was intended to focus mainly on the validation and fine-tuning of all the components developed during this work, as well as on their integration with each other. However, in the beginning of the semester, there was still some implementation work to be done, which resulted in changes to the initial planning. Although the MADM was already implemented, it was just the first part of the global decision mechanism that still had to be designed and implemented. After the development of all the components, the goal was to validate them in a small test bed environment in order to assess the correct operation and interaction with each other. The expected output from this validation was a paper, to be submitted to a scientific conference. After completing the validation, the plan was to perform the integration of the developed components with the NS-3 simulator [24], using the Direct Code Execution (DCE) framework [25], followed by another validation of the work done, this time focusing on scalability issues, which could result in another publication extending the previous one. Finally, at the end of the semester all be obtained results and progress made would be analysed in order to write the dissertation.

In Figure 4 is shown the initial Gantt Diagram where the main tasks planned for the second semester are displayed, as well as the estimated start and end time of each of them. These tasks are detailed as initially defined, in order to better understand what would be done, while in the remaining sections of this chapter present an updated to revision to work performed during the second semester.

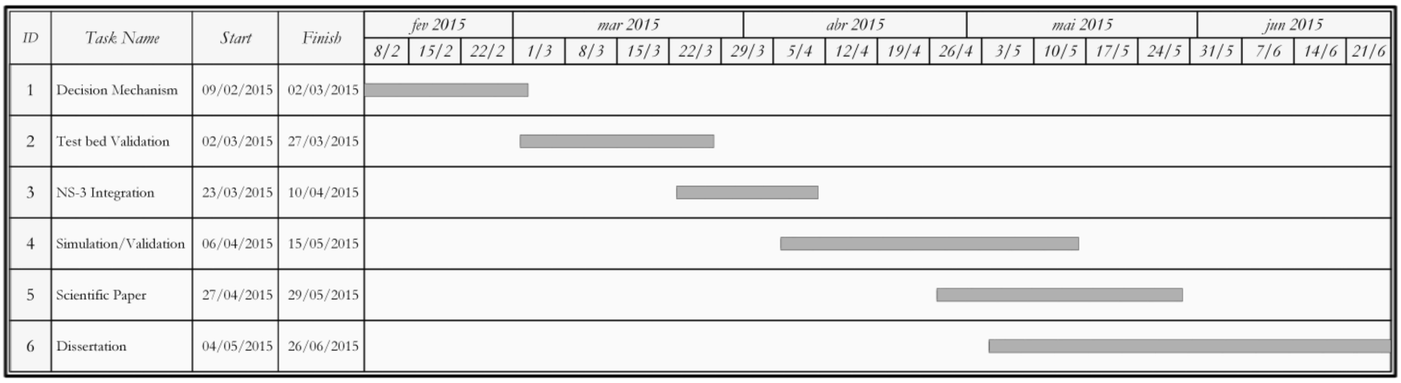

ID	Task Name	Start	Finish
1	Decision Mechanism	09/02/2015	02/03/2015
2	Test bed Validation	02/03/2015	27/03/2015
3	NS-3 Integration	23/03/2015	10/04/2015
4	Simulation/Validation	06/04/2015	15/05/2015
5	Scientific Paper	27/04/2015	29/05/2015
6	Dissertation	04/05/2015	26/06/2015

Figure 4 – Gantt Diagram with the second semester planning

Task 1 – Decision Mechanism:

This task comprises the development of the SCORE mechanism to be part of the FMC Manager, the completion of the FMC Manager tool and the final integration between all the developed components.

At the end of the first semester, only a part of the FMC Manager component was already completed. One of the key missing parts was the decision mechanism as a whole, which still had to be developed and integrated with the chosen external scoring mechanism.

After finishing the development of the decision mechanism, the FMC Manager had to be completed. In particular, the automatic communication mechanisms with the necessary nodes, for sending them the actions to be performed, had to be implemented. Moreover, the interface for the communication with the external Mobility Prediction, service that sends necessary information to the FMC Manager and triggers its actuation, needed also to be completed. To test this interface and to be able to perform controlled tests during the validation, an auxiliary tool was also to be developed to emulate the work of the external Mobility Prediction service.

Task 2 – Test bed Validation

The validation of the work was to be done in two phases, a first one in a small test bed and a second one aiming at scalability tests done by resorting to simulation. The benefits of using a small real test bed are related to the fact that there are many different components interacting with each other, which means that the possibility of finding small bugs is somewhat high. The ability to analyse the entire output from all the components is essential to solve unforeseen problems and to be sure that everything is working correctly.

In order for the validation to be accurate, the simulation would use statistical models to closely represent the distribution of files such as the distribution that exists in the Internet. The correct simulation of the retrieved files popularity was also an important factor to validate the SCORE mechanism so the distribution would also follow observed statistical models. The Zipf law [26] would be used to better simulate a real world scenario based on observations already made by other works and also on works where the content popularity distribution was simulated.

Task 3 – NS-3 Integration

To perform scalability tests the use of a real test bed would not be feasible therefore, all the developed tools were planned to be integrated with the NS-3 simulator. The best way to do this integration was by using one of the NS-3 projects, the DCE.

DCE allows the simulation of a large network while running applications that were developed to work in real life situations. Although all the code has to be compiled with different tools and in a different way in order to work with NS-3, by doing this there is no need to implement the CCNx protocol in NS-3. This is motivated by the fact that the focus of this work is not to validate the CCNx protocol but rather the migration mechanisms.

Task 4 – Simulation/Validation

At this stage, the development should be terminated and a complete set of tests would be done.

The simulation would be repeated in order to simulate different networks scenarios, different types of provided services and popularity distributions, and also the usage of mobility models to thoroughly test as many real life scenarios simulations as possible.

Although this simulation would be used to provide a reliable validation of the work done, its first purpose was to perform scalability tests of all the developed components, which may prove necessary to perform improvements in some components. In the case it was detected that some components were failing due to the complexity and size of the scenarios, there would be an effort to try to improve those components, being then retested in the previously used scenarios. This retesting is not only necessary to access if all components are working properly in the scenario where they failed, but also to measure their performance in the previous scenarios. This is important since any modifications might translate into a different performance on different scenarios.

After completing all the tests, a scientific paper was also planned to be produced either extending the previous one by performing a more complete analysis or if it were not published a paper before, produce one presenting all the work and tests done.

Task 5 – Scientific Paper

With the validation of the SCORE mechanism, the writing of a scientific paper was planned, presenting all the work done and achieved results. However, since that validation would be done in two phases, where first the working version of the SCORE mechanism would be tested, a paper might be produced containing those first results as well as the design of the mechanism, which means that the second could be an extension presenting a more complete analysis and validation of the SCORE mechanism.

Task 6 – Dissertation

This is the last task to be done and consists of writing the dissertation, reporting all the work done during this year. In the dissertation it will be presented all the research made, the problems faced during the development and the final result, both in terms of contributions to the scientific community with the research done and in terms of contributions for the field with the developed work.

Although the first part of the dissertation is to be based on the intermediate report, namely, the research made in the field and the proposed objectives, the remaining of the dissertation will present a more complete description of the work done. This description will be composed by the details of all the components made, the specification of the entire test scenarios used, both in the real test bed and in the simulation, and the results obtained with the tests made.

Finally, the dissertation will present an overview of the importance of the work done and its contribution to the scientific community, in addition to the next steps that can be taken to further improve the work done.

4.2 Revision of the second semester plan

In the beginning of the second semester there was a change in the planned tasks for the second semester. This change was based both on the feedback from the jury and on feedback from the MCN project reviewers regarding the contributions made to one of the project deliverables, which detailed the SCORE mechanism architecture. More relevance was given to the test phase, focusing on the choice of the MADM algorithm, in order to validate not only that the algorithm was correctly chosen but also that it can perform under different scenarios with an acceptable performance.

The changes in the planned tasks, however, were not only motivated by the increased focus on the test phase. In fact, due to changes by the MOBaaS service, adjustments in the defined architecture of SCORE were also necessary, and there was a revision of the SCORE mechanism, which resulted in a second specification of the mechanism and consequent adaptation of the planned tasks.

The changes in MOBaaS resulted in an increased amount of implementation work during this second semester and also an increase the necessary effort to integrate all the services. Because the MOBaaS service did not follow the initially specified API, the implementation of an extra module was also needed in order to guarantee the integration between MOBaaS and the SCORE mechanism.

2nd Semester	08/02/15	03/07/15
SCORE Mechanism	**08/02/15**	**29/05/15**
Analysis of intermediate defence feedback	08/02/15	13/02/15
MADM Evaluation	15/02/15	15/05/15
Define Tests	15/02/15	10/04/15
Develop Test and Evaluation Mechanisms	22/02/15	10/04/15
Run Tests	02/03/15	24/04/15
Analyse Results	23/03/15	15/05/15
Implementation	30/03/15	15/05/15
Scientific Paper	03/05/15	03/07/15
MCN Integration	**11/05/15**	**12/06/15**
Integration with MOBaaS	11/05/15	22/05/15
Integration with ICNaaS	11/05/15	22/05/15
Evaluation	25/05/15	12/06/15
Dissertation	**27/04/15**	**03/07/15**
Report work	27/04/15	03/07/15
Revision process	01/06/15	03/07/15

Table 1 – Revised tasks for the second semester

When comparing the initial planning with the revised one, it is possible to see that two tasks (i.e. tasks 3 and 4) were replaced for other tasks mostly focused on integration activities. These two tasks were removed because, as it was already said, the amount of implementation to be done in the second semester was significantly increased while, simultaneously, planned evaluation tests were modified.

With the revised work plan, on one hand, there was an increased focus on the selection of the MADM algorithm to be included in the SCORE mechanism and on its validation. On the other hand, in order to completely test the SCORE mechanism, it is necessary to work directly with two other components of the MCN project, the MOBaaS service, which feeds information to the SCORE mechanism, and the ICNaaS service where the SCORE mechanism is directly integrated.

Looking into more detail at each task, it is possible to better understand why so many tasks had to be replaced and the work that had to be done.

MADM Evaluation Task

As it is presented in Table 1, this task is composed of many sub-tasks. The first step was to define the different tests to be made as well as the scenarios that were going to be simulated. With the tests and scenarios defined, each scenario had to be created using the necessary machines and tools, while also the scripts to automatize the tests had to be written. With both the scenarios and scripts completed, the tests were then run. Due to their complexity, the tests yield a very long completion time, requiring a close inspection to avoid any repetitions due to unforeseen problems, which would greatly delay the obtaining of results for analysis. With the tests completed, the necessary scripts to perform the analysis of the results were also created as there was a great amount of generated output.

Implementation Task

This task includes Task 1 from the initial planning plus extra work that came with the modifications of the MOBaaS service.

It consisted of the development of the FMC Manager that complemented the existing monitoring part, which had been started during the first semester. Regarding the decision part of the FMC Manager, as well as the transmission of actions to the routers and the communication with both ICNaaS and MOBaaS services, everything still had to be implemented.

Due to the changes in the MOBaaS service, a new module to be placed between this service and the FMC Manager, as a middleware proxy, also had to be developed.

MCN Integration Task

With the completion of the FMC Manager the integration with both the ICNaaS and MOBaaS had to be checked, ensuring that any possible existing problems were solved, guaranteeing that all the services worked correctly with each other and were also tolerant to any faults that might occur.

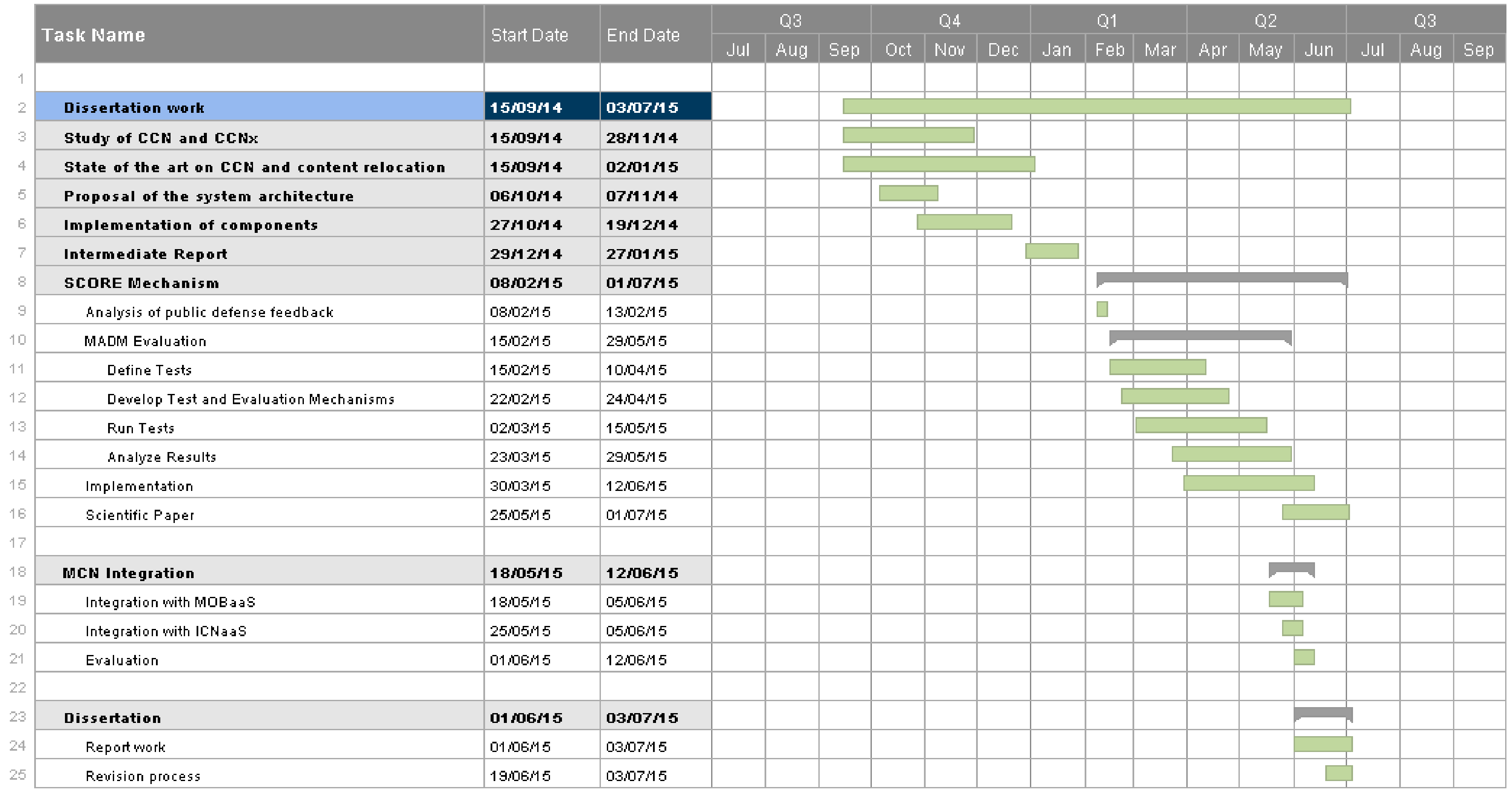
Scientific Paper and Dissertation Tasks

The focus of both these tasks, in general, remained the same, where the changes consisted of a reschedule of the starting point for these tasks.

The scientific paper and the dissertation would be developed in parallel since both would present the similar information, the SCORE system and its architecture, as well as the performed tests and obtained results.

4.3 Work evolution

Even though some possible delays were envisaged in the initial planning, it is particularly difficult to accurately plan the timeline of work to be developed when it depends on external third parties and partners from a large-scale research project, such as MCN. Furthermore, the time to run the tests could not be exactly predicted, which caused additional delays in evaluation based on the results from those tests. Nonetheless, the system as a whole was implemented, integrated and evaluated as presented in Figure 5, which depicts the final Gantt diagram and the evolution of the work done during this dissertation.



	Task Name	Start Date	End Date
1			
2	**Dissertation work**	**15/09/14**	**03/07/15**
3	**Study of CCN and CCNx**	**15/09/14**	**28/11/14**
4	**State of the art on CCN and content relocation**	**15/09/14**	**02/01/15**
5	**Proposal of the system architecture**	**06/10/14**	**07/11/14**
6	**Implementation of components**	**27/10/14**	**19/12/14**
7	**Intermediate Report**	**29/12/14**	**27/01/15**
8	**SCORE Mechanism**	**08/02/15**	**01/07/15**
9	Analysis of public defense feedback	08/02/15	13/02/15
10	MADM Evaluation	15/02/15	29/05/15
11	Define Tests	15/02/15	10/04/15
12	Develop Test and Evaluation Mechanisms	22/02/15	24/04/15
13	Run Tests	02/03/15	15/05/15
14	Analyze Results	23/03/15	29/05/15
15	Implementation	30/03/15	12/06/15
16	Scientific Paper	25/05/15	01/07/15
17			
18	**MCN Integration**	**18/05/15**	**12/06/15**
19	Integration with MOBaaS	18/05/15	05/06/15
20	Integration with ICNaaS	25/05/15	05/06/15
21	Evaluation	01/06/15	12/06/15
22			
23	**Dissertation**	**01/06/15**	**03/07/15**
24	Report work	01/06/15	03/07/15
25	Revision process	19/06/15	03/07/15

Figure 5 – Final Gantt Diagram

5 Developed Work

All the developed work in this semester in presented in this chapter, from the new developed tools to the changes made to already existing frameworks and other sources.

The authors of the MeTHODICAL algorithm, developed in R [27], provided an implementation of the MADM algorithms. However, it was necessary to adapt the provided implementation to this work, mainly the method called to run the algorithms and some extra data manipulation mechanisms, as well as some protection needed in specific cases where the data was not sufficient for the algorithm to correctly work.

5.1 SCORE System Overview

In Figure 6 is presented an overview of the SCORE system developed during this work. The various components and modules are depicted in this overview with different colours. The newly created components are depicted in a green background, components in a blue background represent existing components that suffered some changes in their original source-code, while the yellow components were kept untouched.

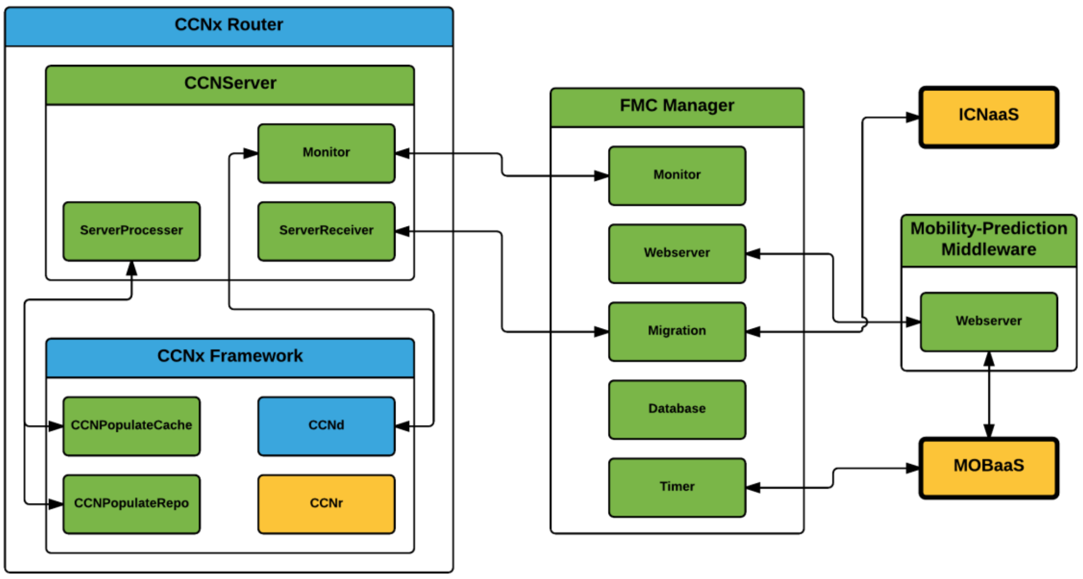


Figure 6 – SCORE System Overview

The CCNServer and its components, responsible for the communication between the CCNx Router and the FMC Manager, are fully detailed in Section 5.4 while the modules responsible for the migration of the contents, CCNPopulateCache and CCNPopulateRepo, are presented in Section 5.5. The database that is part of the FMC Manager is detailed in Section 5.6 while the remainder of its components are explained in Section 5.7. Finally, the Mobility-Prediction Middleware is depicted in Section 5.8.

5.2 Sequence Diagram

A simple sequence diagram of the interaction between all the components is depicted in Figure 7. Despite being displayed in the same diagram, there are two separate sequences:

- Steps 1 and 2 represent the actions done regarding the monitoring of the processed data and the storage of that data in the database;
- Steps 3 and 4 show what is done to perform the relocation of contents;

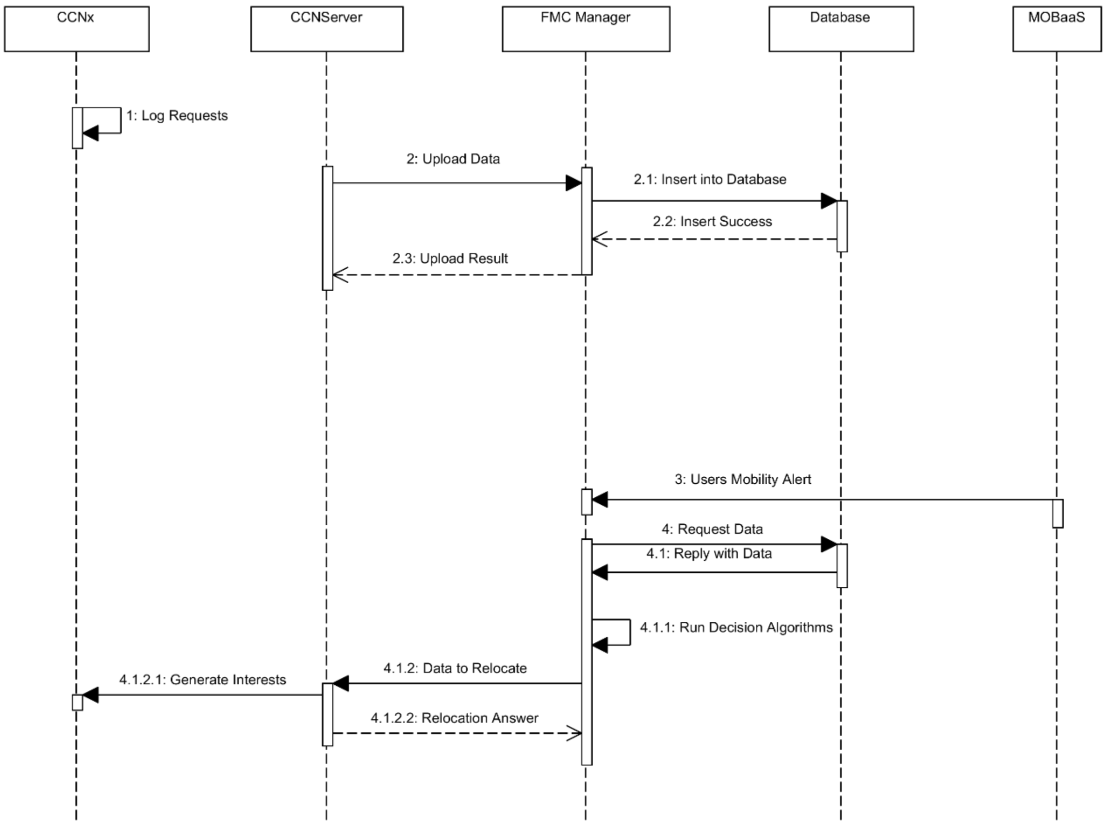


Figure 7 – SCORE System Sequence Diagram

As already stated, these two sequences are completely independent which means that they can even occur at the same time. Although not present in the diagram, the Mobility-Prediction Middleware module is located between the MOBaaS and the FMC Manager acting as a simple translator between both services.

5.3 CCNx Router

The CCNx router is the name given to any device running the CCNx daemons that can either be a machine in the core network or a router near to the edge of the network, such as the routers existing in a university campus. With the proposed architecture, in every router, besides the CCNx daemons there will be an extra one, the CCNServer daemon. The code of the CCNx framework was slightly modified so that if monitors all the requests that it processes and writes the necessary information to a local named pipe.

5.4 CCN Server

This is the main daemon created to run in every router, being responsible for all the interactions with the centralized controllers and responsible to perform all the necessary local operations. This tool, which was developed in Java, is responsible for two main actions, sending the log information collected locally to the database and receiving the orders to perform the content migration. To do so, it has multiple threads, each one with a specific purpose as presented next.

5.4.1 ServerReceiver

The ServerReceiver thread is the one responsible to receive new connections from the FMC Manager when the latter wants to transmit any action order, either a relocation of content to the local cache or to the local repository. It listens to incoming connections and every time it receives a new connection, it instantiates a new ServerProcesser thread to perform the desired actions.

5.4.2 ServerProcesser

As stated above, this thread is instantiated when a new connection is established between the FMC Manager and one CCNx Router. The thread will read the incoming message, attempt to perform the requested actions and then it will reply either confirming that everything went as expected, that it failed completely or that the process was only partially completed.

Key	Value
Type	populatecachemsg
unversioned	True | False
enumerate	True | False
Names	List with ccnnames
Timeout	Integer value

Table 2 – Message format to request a cache migration

Table 2 and Table 3 specify the format of the messages received by the ServerProcesser thread, containing all the necessary information to guarantee the execution of the right action. Table 4 specifies the format of the message sent by the ServerProcesser thread after completing the task or when something fails while performing it.

Key	Value
Type	populaterepomsg
unversioned	True | False
enumerate	True | False
names	List with ccnnames
timeout	Integer value
verify	True | False
repositories	Integer value

Table 3 – Message format to request a repository migration

5.4.3 Monitor

In order to send the collected monitoring data to the centralized database, this thread will run periodically, establishing a new connection to the FMC Manager and uploading the recorded data. When this thread is initialized, it instantiates another thread, LogReader, whose only purpose is to read the data that the CCNx framework outputs to a shared named pipe. The LogReader thread adds every entry to a synchronized array that is then, periodically, processed by the Monitor thread, which finally sends all the currently available data to the centralized database.

Key	Value
type	reply
code	200 | 250 | 500
names	Null | List with ccnnames

Table 4 – Message format of the reply

To send the data, the Monitor thread sends a message with the format described in Table 5, expecting a reply message with the same format as the one described in Table 4. If the message says that something failed or if the reply message is not received, the Monitor thread will keep the data and will try to send it in the next iteration, otherwise if the reply message says that everything went ok, then the data that was sent is then deleted since it is already stored in the centralized database.

Key	Value
Type	interestlogmsg
Repository	True | False
Edge	True | False
entries	List with log entries
location	String

Table 5 – Message format of the log transfer

5.5 CCNx Modules

In order to perform the migration of contents, it is necessary to communicate with other CCNx routers using its protocol. To perform this, two modules were created based on a tool provided with the CCNx framework. One module to perform the migration to the local cache, **CCNPopulateCache**, and other module to migrate contents to the local repository, **CCNPopulateRepo**.

Although there are some differences in the modules, since the destination of the contents differ between each other, there are still a lot of common steps between both modules. Therefore, the majority of the parameters that both modules can receive are common with the exception of two. These parameters are:

- **unversioned**: Files are added to the repository without the CCNx versioning control;
- **enumerate**: The retrieved CCNNames are not complete names, they are only prefixes. Therefore, before importing files, the module should perform a recursive enumeration and only then import all files found under those prefixes;
- **timeout:** The timeout used when performing an enumeration;

- **verify**: Before importing any files, the module constructs a list with all the existing files present in the repository (local enumeration). This allows the system to import only missing files or files with newer versions; *(CCNPopulateRepo only)*
- **repositories**: the number of repositories that should reply to the enumeration request. While developing the CCNPopulateRepo module there was the need to perform a small modification to the CCNx Java API. This allowed performing local enumerations, which was not supported by the Java API since it was not prepared to specify the necessary fields to restrain an enumeration locally. *(CCNPopulateRepo only)*

5.6 Database

The database used to store all the monitoring data will have to deal with a great number of entries as well as concurrency of accesses. It should also be efficient and as fast as possible while having the relational feature since the data as to be associated with the location where it was collected. With all this in mind, the technology chosen was PostgreSQL [28], since it is an open source relational database system fully compliant with the ACID properties, atomicity, consistency, isolation and durability.

The database will contain two tables, one to store all the information regarding the existent routers in the network and another to store all the monitoring data collect by each of the routers.

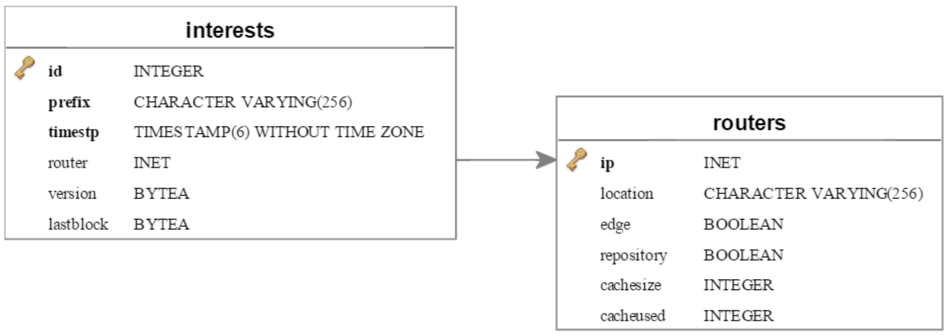


Figure 8 – Database Entity-Relation Diagram

Figure 8 shows the Entity-Relationship Diagram of the created database with the existent fields in both tables. In the **routers** table, not only the IP and the location of the router are stored, but also other data that can later become useful for the decision mechanism. For instance, knowing if a router has a repository or not is extremely important when deciding the type of migration to perform. In the **interests** table, the prefix (unique name) of the content and the timestamp are stored, as well as other information such as the number of the last block of the object, which allows knowing, approximately, the size of the file. This information is also extremely important for the decision mechanism as the size of the file gives a good idea of the cost of the migration of the content in the network.

5.7 FMC Manager

The FMC Manager is the main component of the work being responsible for the communication with every CCNx Router, receiving the monitor data to add to the database, making every decision regarding the migration of contents and transmitting the actions to the necessary CCNx Routers.

Although it is named after a concept that already exists, the FMC Manager is the application of the Follow-Me Cloud concept in this new scenario, which means that the FMC Manager is a new component implemented and designed from scratch.

This component was developed in Python, since it integrate with other external tools, such as PostgreSQL for the database and R for the MADM tool. Being implemented in Python, it is much easier to integrate with the other tools thanks to the many existing and tested modules, such as rpy2 [29], use for the communication between R and Python, and psycopg [30] used for the integration with PostgreSQL.

Since this is the brain of the entire concept, it is composed of multiple threads running simultaneously, responsible for different actions, such as, the web service that listens to incoming messages from MOBaaS, or in this case, from the module developed to translate the MOBaaS information, the thread that receives the monitoring information from the routers and the threads that are instantiated every time a migration of contents is triggered. Due to the way the MOBaaS service was designed, there is also a thread, of the type Timer, which does a registration on the MOBaaS web service every day at the beginning of the day.

5.7.1 Monitor

With the objective of receiving the monitoring data from the CCNx routers, this thread listens for incoming connections processing the messages detailed above in section 5.4.3.

When new information is received, a preliminary check verifies if that router is already present in the database, adding it if necessary, appending then the received log data from that router to the database.

Since scalability tests have not been performed so far, this is working as a single-threaded server. However, if the scalability tests prove the need for more performance, the conversion to a multi-thread server has already been accounted for, where the only sensible aspect is the verification/addition on the routers table.

5.7.2 Webserver

This is the implementation of the web service that the FMC Manager keeps running in order to receive mobility triggers.

When it receives a POST message with the correct JSON message, with all the details regarding the migration, such as the origin cells of the users, the number of users coming from each cell, the destination cell and the number of users at that cell, a new migration is triggered.

In this project this messages will come from the MOBaaS service, however, any other source of prediction or detection of mobility can be used to trigger a migration of contents.

5.7.3 Migration

There are several steps since a new migration is triggered until the routers are notified with the list of contents that they should possess in its cache:

1. Process the incoming information
2. Get the IPs of the routers of all the cells from ICNaaS
3. Obtain content popularity from each source
4. Obtain content popularity from the destination
5. Determine the global popularity of each content
6. Prepare data for MADM algorithm
7. Run MADM algorithm
8. Obtain results of the MADM algorithm
9. Compile list of contents based on the cache size
10. Send list to all the routers at the destination cell

Since the information regarding the source and destination of users that is sent by the MOBaaS service contains cell ids and not router IP addresses, it is necessary to connect with the ICNaaS web service to obtain the list of routers that are present in each cell, and obtain the respective IPs. After getting all the IPs, it will fetch the data from its own database where all the monitoring data is stored. This data is used to determine the global popularity of each content, considering the universe composed of all the sources and the destination cells. Before running the MADM algorithm with this data, there is the need to do a mapping between the content prefixes and an index since the MADM algorithm cannot deal with strings. After executing the MADM algorithm, it is done a reverse mapping while compiling the list of contents to be moved. This list is created based on the size of the cache of the destination routers that was early obtained from the database. When the list is complete, it is sent to all the routers in the destination cell, since there is no specific information of the routers where the users are going to move to.

5.8 Mobility-Prediction Middleware

This module was needed to implement due to the changes in the MOBaaS service from the planned API. It implements a web service that listens for incoming POST messages from the MOBaaS service, manipulates the data, and then sends that data in another POST message to the FMC Manager, being this the trigger for a migration of contents.

In the first planned API, the MOBaaS service was supposed to send the triggers to the FMC Manager every time there was a migration of users, triggering the decision-making mechanism. However, this was later changed by the MOBaaS service. Instead, the information received from MOBaaS is a list of users, with their current cell id and a list of cell ids and the probability of them moving to those cells. This list however can contain the same cell, which means that there is also a chance of they remaining at the same cell.

When analysing and parsing the data, the module will compile multiple lists, according to the different destinations, with the information provided from the MOBaaS service. It is considered a valid possible migration, if the destination cell is different of the source one (not moving) and if the probability of going to that cell is higher than 50%.

Each of these lists is considered a trigger, which means that a single message from MOBaaS can translate in multiple triggers sent to the FMC Manager service.

6 Tests and Evaluation

This chapter presents the setup and test bed used to perform the different tests, as well as the scenarios under evaluation. It is also explained how the different tests were performed, and are presented the results obtained with an analysis of them.

The results of both specifications of the SCORE mechanism were achieved by analysing the outputs of several evaluation scripts developed in the R language [31]. The availability of existing implementations of Scoring/Decision mechanisms in this language motivated the use of R and the development of all the necessary scripts and methods to perform this analysis in R.

6.1 Test bed Specification

In Figure 9 are depicted the machines that compose the test bed and the components that are running in each one. In this case, the Machine 2 are completely dedicated to the FMC Manager and its own database, while Machine 1 acts as a router that is serving final users.

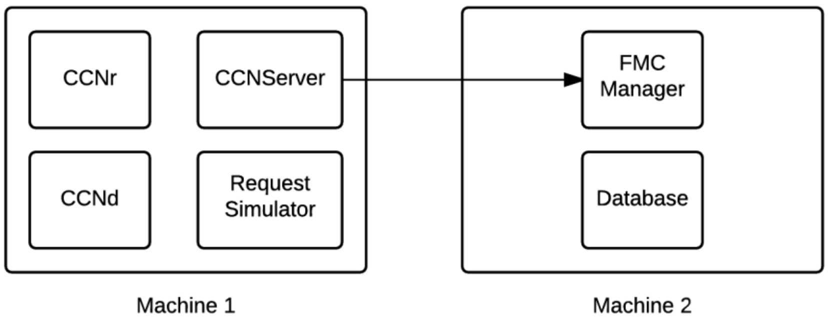


Figure 9 – Components of the FMC evaluation

The components running in Machine 2 are already described in Section 5, regarding the components of Machine 1 they will be now explained.

CCNd and CCNr are two daemons that are part of the CCNx framework, the first daemon is the main one of CCNx responsible to send and receive messages, while the latter is the daemon responsible for the local repository of contents. This repository is populated with the amount of files specified in each scenario, following the distributions of sizes and classes specified in each case. The CCNServer daemon was also detailed above, and in this case, its only function is to upload the local monitoring data to the FMC Manager. The final component, the Request Simulator, is a script developed in Python that simulates the requests coming from multiple users. This script is continuously generating requests for the files present in the local repository based on the Zipf distribution specified in each of the scenarios under evaluation.

Regarding the machines, both are virtual machines running Ubuntu and since the test bed is small, in some cases it was replicated, meaning that two different scenarios were running simultaneously using four machines. Since the analysis needs the data considering at least two distinct routers populated with requests, each scenario was run twice to create monitoring data representing a source and a destination of users. In order to simulate highly populated routers that have been serving users for a long time, each test run had a long runtime in order to generate high amounts of requests, around 250000 requests for each router in the smaller scenarios and around 1000000 requests per router in the bigger scenarios.

6.2 Scenarios Specification

Aiming at perform a generic validation, different scenarios were specified so that the FMC Manager could be evaluated against multiple representations of what can be found in a real Internet applications. Table 6 summarize the details of each of the different scenarios specified for this validation.

Parameter	Normal	YouTube	Webserver
Request Popularity	Zipf distribution		
	$\alpha = 1$	$\alpha = 2$	$\alpha = 1$
Number of Popularity Classes	10	20	20
File sizes per class	Normal Distribution $\mu = 30.60$, $\sigma^2 = 15.72$ Min. 150KB Max. 70MB	Gamma distribution $\alpha = 1.8$, $\beta = 5500$ Min. 500KB Max. 100MB	Gamma distribution $\alpha = 1.8$, $\beta = 1200$ Min. 50KB Max. 50MB
File Distribution per Class	Zipf distribution with reversed classes		
	$\alpha = 2$	$\alpha = 1$	$\alpha = 1$
Total number of files	2000	2000, 10000	2000, 20000

Table 6 – Scenarios Specification

The request popularity follows a Zipf distribution as specified by existing works in the literature [32], considering the parameters of each scenario, such as the value of α and the number of files. The distribution of files per popularity class also follows a Zipf distribution but this time with a reverse mapping based on the related work [33] and [34]. A reverse mapping is applied because it has been demonstrated that the great majority of files are unpopular with only a few amount of files becoming very popular.

The file sizes in the Normal scenario follow a Normal distribution with mean 30.6MB and variance of 15.72MB. The distribution is based on a survey made of the current Internet statistics, such as the average size of a Webpage (2 MB) [35] and the average size of a minute YouTube video with a resolution of 720p (40MB) [36]. The second scenario, YouTube, follows a model already defined in the literature [37] while the third scenario, Webserver, is based on the observed growth of the size of files available at file webservers [38]. These models consider Gamma distributions with $\alpha = 1.8$ for both, $\beta = 5500$ for the YouTube model and $\beta = 1200$ for the Webserver model, respectively.

6.3 Tests Specification

When planning the tests for the first specification of the SCORE mechanism the intention was not only to compare the different MADM algorithms, in order to identify the best one, but also to analyse how each metric affected the final output of the algorithms. This second part would be important later when fine-tuning the weight of each of the metrics. With this in mind, a set of tests was created, which is detailed in Table 7, aiming at answering both questions. The MADM algorithms chosen consider two different types of metrics: benefits, that should be maximized, and costs, that should be minimized. Based on this, there is only one benefit, the popularity at the source of the users, and there are three costs, the popularity at the destination of the users, the size of each file, and the ratio between the size of the files and the size of the cache of the destination router. When getting the monitoring data from the database, the amount of requests could also be manipulated. For instance, it was possible to consider that there are no data at the destination (a new instantiated router) or that it has much less data than the source one (recently instantiated router).

Looking at the test specification, Test 1, for example, only considers the popularity at source and the popularity at destination. However, since it is considered that there is no information recorded at the destination router, this means that the output is based on a single metric, the popularity at the source of users. Test 4 is very similar to this one, with the only difference being that it considers less recorded requests at the source of the users. Unlike these, Tests 2, 3, 5 and 6 both consider two metrics, popularity at source and at the destination, changing from test to test the amount of requests recorded at source and/or destination. Tests 7 and 8 also consider two metrics, however the considered ones are the popularity at source and the file size. Tests 9 to 12 already consider three metrics, popularity at source, popularity at destination and file size, while Testes 13 to 16 consider all the four metrics. In both this cases, the amount of requests recorded at source and destination are also manipulated in order to simulate different situations. By adding one metric at a time, it is possible to observe the impact it has, if any, on the output obtained from the different MADM algorithms.

TestID	Pop_Src	Pop_Dst	Weight_Pop_Src	Weight_Ratio	Weight_Size	Weight_Pop_Dst
1	100	0	1	0	0	1
2	100	100	1	0	0	1
3	100	50	1	0	0	1
4	50	0	1	0	0	1
5	50	50	1	0	0	1
6	50	100	1	0	0	1
7	100	N/A	1	0	1	0
8	50	N/A	1	0	1	0
9	100	100	1	0	0.5	0.5
10	100	50	1	0	0.5	0.5
11	50	100	1	0	0.5	0.5
12	50	50	1	0	0.5	0.5
13	100	100	1	0.33	0.33	0.33
14	100	50	1	0.33	0.33	0.33
15	50	100	1	0.33	0.33	0.33
16	50	50	1	0.33	0.33	0.33

Table 7 – First test set specification

With the second specification to the SCORE mechanism, however, this set of tests no longer made sense and could be applied, so new tests had to be specified and executed. With the revision of the mechanism, previously stated, both popularity metrics were merge into one, and the metric of the ratio between the size of the files and the size of the cache disappeared since it was related with the file size metric and could be removed without affecting the output of the MADM algorithms. The MADM algorithms would then consider one benefit, the popularity of contents and one cost, the size of the files. Since now the evaluation of the MADM algorithms would be done by comparing the outputs with the ideal solution obtained with the knapsack algorithm, a new set of tests was made.

The knapsack algorithm gives the maximum achievable value for the given cache size, while it is also possible to obtain the list of files that were selected, but this means it was necessary to specify the cache sizes to consider. The cache sizes chosen were 256 and 512 MB, 1, 2, 4 and 8 GB since the cache is stored in RAM due to the need of low access latency. The value of the files stored in cache is the sum of its individual value, their popularity, which has been normalized when Equation 2 was applied.

6.4 Results

This section presents the results from the evaluation of both specifications of the SCORE mechanism. Since these evaluations are very different from each other, and both generated a large amount of data to be analysed, the results were divided into two different subsections.

6.4.1 First specification of the SCORE mechanism

With the first specification of the SCORE mechanism, one of the major questions was how to determine the theoretical solution onto which would be compared the output of each MADM algorithm, and also how to perform this comparison.

To answer the first question, the path followed was a comparison with a sorted list of each metric considered in that test. This comparison would be made with a line-by-line matching with five different degrees:

- Same position – Green;
- One position above or below [x-1 ; x+1] – Yellow;
- Five positions above or below [x-5 ; x+5] – Orange;
- Ten positions above or below [x-10 ; x+10] – Red;
- None of the above (out of scope) – Grey.

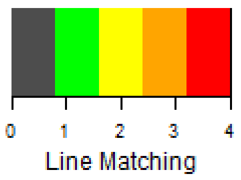


Figure 10 – Line matching colour scheme

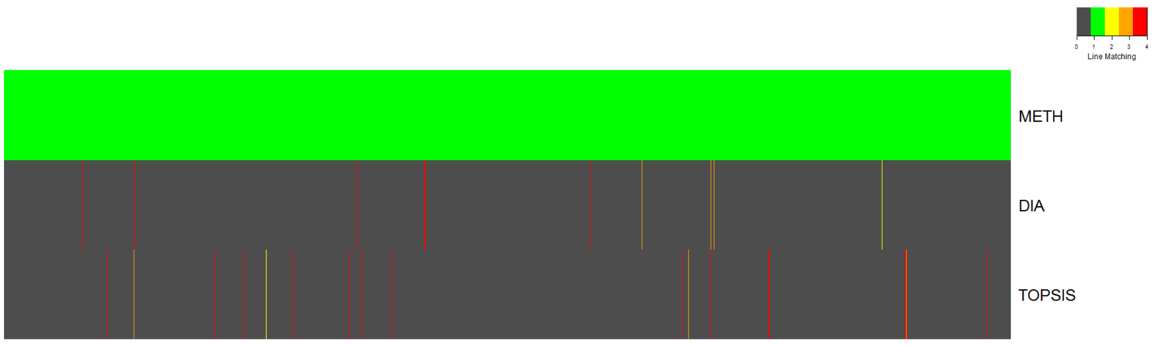


Figure 11 – Results of test 1 in the Normal scenario

Figure 11 shows this analysis applied to Test 1 of the Normal scenario. Since in Test 1 only one metric is considered, there is only one comparison per MADM algorithm. As it is possible to observe, the MeTHODICAL algorithm as a perfect matching with the sorted list of contents based on the popularity at source alone. DiA and TOPSIS on the other hand, show a poor matching with only a few files near the expected position in this case.

If there was only one metric to compare, this method might be further exploited, however as it can be seen in Figure 12, Figure 13 and Figure 14, which have, respectively, 2, 3 and 4 metrics to compare per algorithm, this analysis doesn't provide a global result per metric.

Furthermore, this presentation of the results makes it easy to see that the file size metric and the ratio between the file size and the cache size give the exact same result, which means that only one of them is enough in this application.

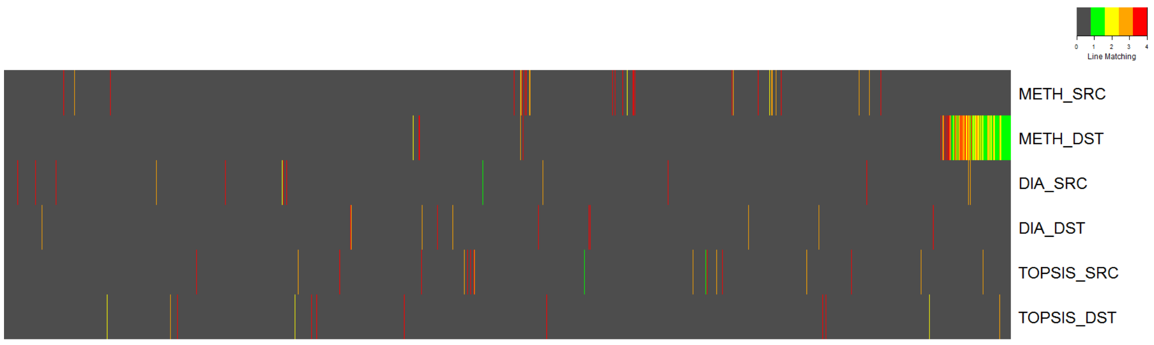


Figure 12 – Results of test 2 in the Normal scenario

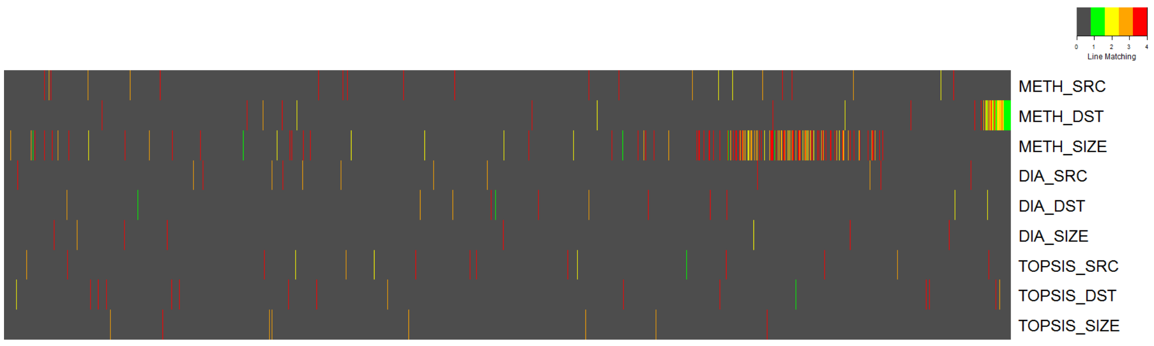


Figure 13 – Results of test 9 in the Normal scenario

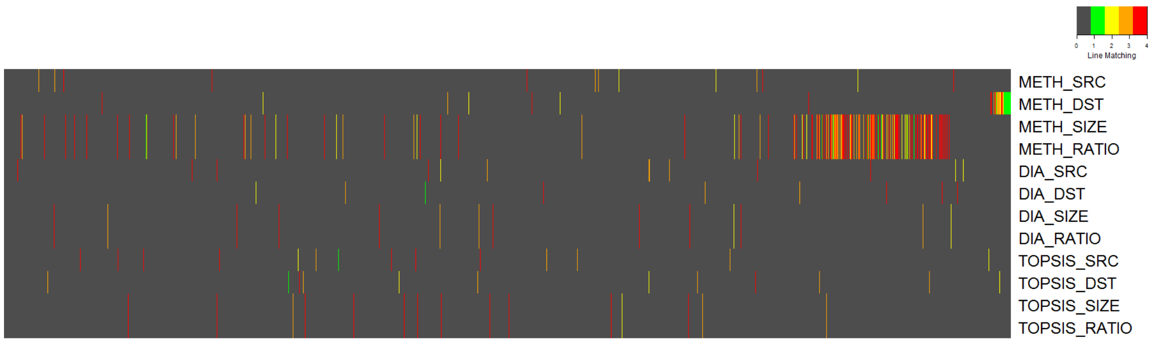


Figure 14 – Results of test 13 in the Normal scenario

With the objective of combining all the comparisons per test, it was done a merge with the comparison of each metric. This merge considered the different metrics and choosing the best match in each case, outputting this way only one analysis per MADM algorithm per test. Figure 15, Figure 16 and Figure 17 show this analysis applied to the same tests presented above. It is possible to better compare the performance of the three algorithms being clear that the MeTHODICAL algorithm outperforms the other two, however this analysis is not very accurate and can even be biased since when merging the results it is selected the best one. One way of solving this would be to take the average of the results from the multiple metrics but this analysis would still remain little accurate.

This same analysis, for all the remaining scenarios, can be found in Appendix B. The results follow the same line of thought, although there is an extra difficulty in the scenarios with 10000 and 20000 files, resulting from a loss in the resolution as there are too many vertical lines and only clusters can be differentiated.

All these difficulties, combined, motivated a fresh look at the problem itself from which resulted a revision and the second specification of the SCORE mechanism.

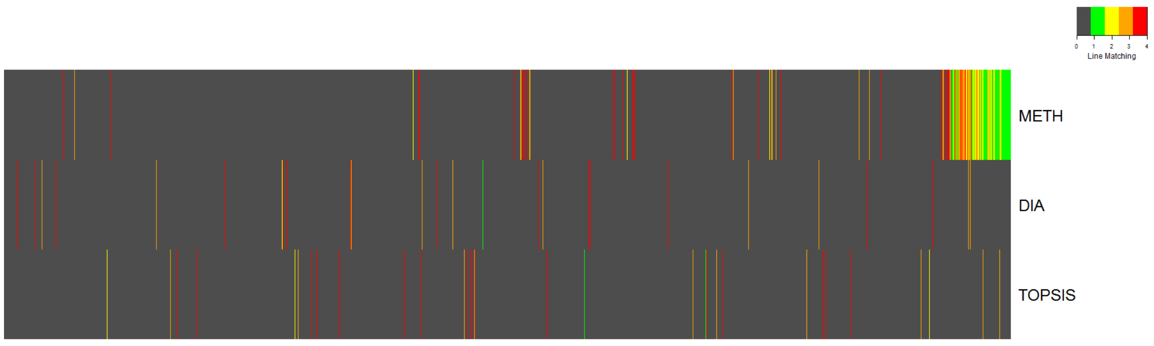


Figure 15 – Merged results of test 2 in the Normal scenario

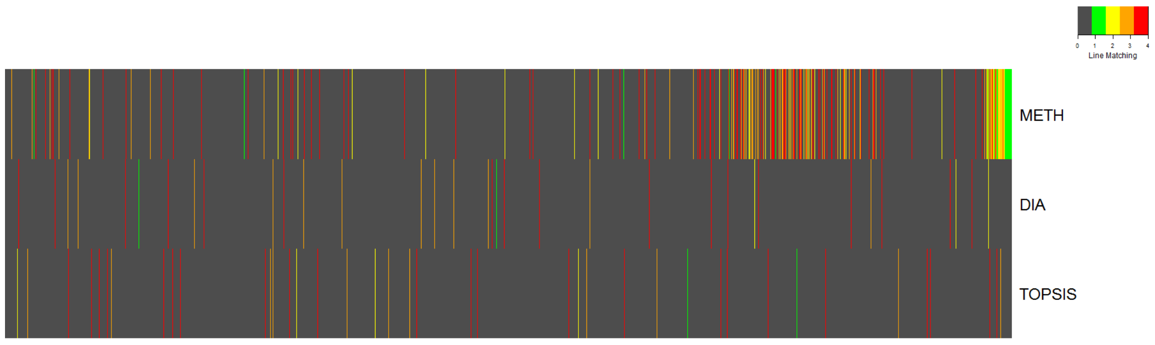


Figure 16 – Merged results of test 9 in the Normal scenario

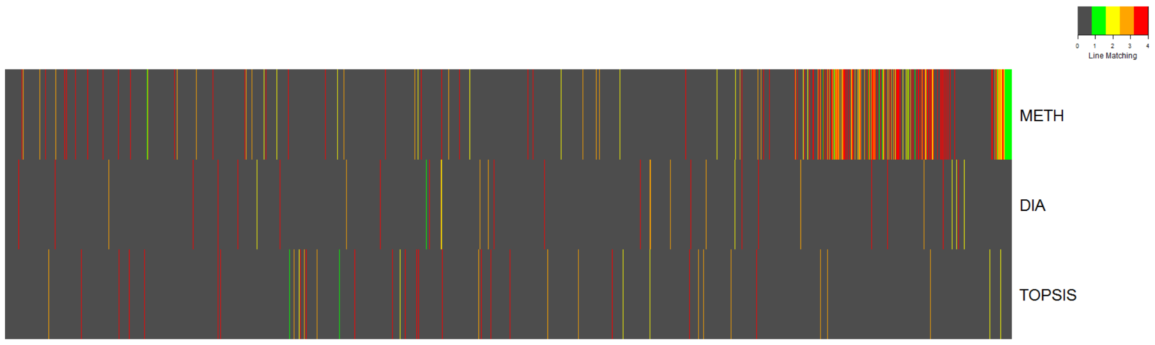


Figure 17 – Merged results of test 13 in the Normal scenario

6.4.2 Second specification of the SCORE mechanism

With this second specification of the SCORE mechanism, the initial planned tests were not entirely performed. The scenarios with 10000 and 20000 files, as well as the cache size of 8GB, were not completed since the vast amount of data being considered made the implemented methods in R crash. The reason of the crashes was related with the matrixes' implementation in R. These had a limit in the number of rows and columns that could be used and, in order to overcome this situation, a different structure to store the data would have to be considered, as well as another implementation of the algorithms. The analysis was therefore restricted to the three scenarios with 2000 files and the cache sizes 256 and 512MB, 1, 2 e 4 GB.

Table 8 presents the results obtained with the knapsack algorithm for each scenario and cache size considered. The results represent the ideal solution for each combination of scenario and cache size, the maximum achievable value with the total size of files limited to the cache size. These results allow a better understanding of the impact that the popularity distributions cause on each scenario, where, for instance, 47 files in the YouTube scenario account for 71% of the requests received, whereas to achieve a similar value in the Webserver scenario 366 files are needed.

	Normal		YouTube		Webserver	
Cache Size	Number of files	Value in Cache (%)	Number of files	Value in Cache (%)	Number of files	Value in Cache (%)
256MB	15	29,71	47	70,69	233	63,06
512MB	29	43,1	83	82,07	366	75,74
1GB	47	56,31	160	90,5	648	87,75
2GB	78	69,67	300	95,28	1202	96,32
4GB	156	80,6	558	98,12	1980	99,95

Table 8 – Knapsack results per scenario

Figure 18 presents the percentage of files that each MADM algorithm correctly identified, when compared against to the ones selected by the knapsack algorithm, while Figure 19, presents the percentage of files in that are correct between all the moved files by the MADM algorithm. These results represent the average of the performance of each algorithm in the different cache sizes considered. With these results, it is very clear that not only the MeTHODICAL algorithm has a very high performance, but also delivers a consistent performance independently of the cache size (low standard deviation), correctly identifying around 89% of the files averaging all the scenarios. As for the other two MADM algorithms, they deliver a much lower performance together with far less consistency (high standard deviation). The cause for this high standard deviation can be better understood when analysing the value results. Although they might seem very similar, these two metrics evaluate two distinct things that are very important. On one hand, it is important to observe if the MADM algorithms have chosen all or most of the correct files, but on the other hand, it is also important that they do not choose a long list of incorrect files, since this would mean that a lot of unnecessary files would be moved to the router’s local cache.

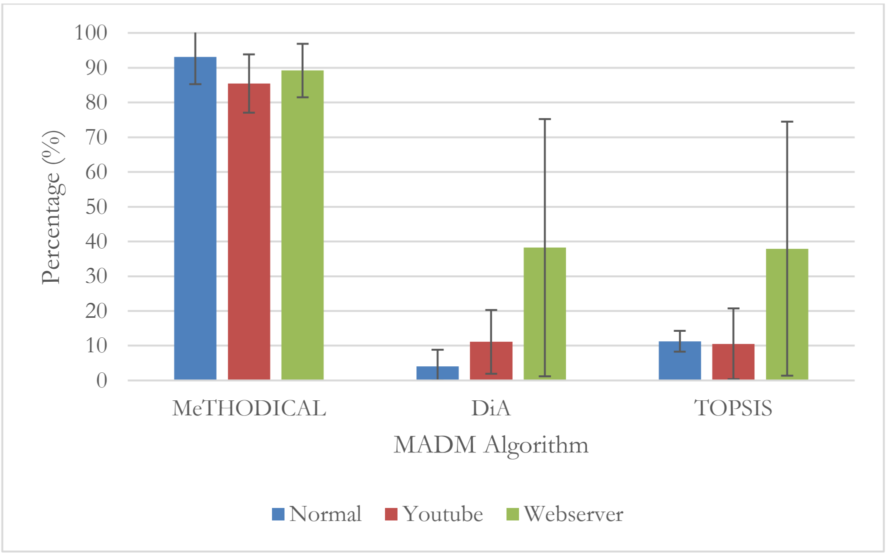


Figure 18 – Percentage of files correctly identified

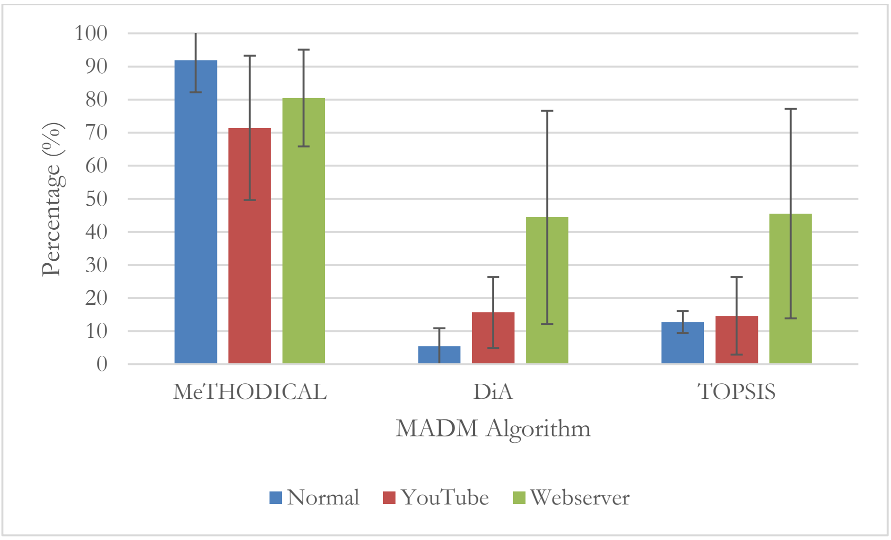


Figure 19 – Files correctly moved

Following the trend set by the percentage of files correctly chosen and correctly moved, in Figure 20, it is shown the percentage that each algorithm can achieve with the files it chooses to move, compared to the highest possible value determined with the knapsack algorithm, in the Normal scenario. As it was explained, the value in cache represent the sum of the percentage of requests each content received, this means that a router with a cache value of 70%, where the users are requesting the files based on the same popularity distribution as before, can serve around 70% of the requests it receives using only its local cache, without the need to forward the requests to other routers. Figure 21 and Figure 22 depict the same reasoning for the YouTube and Webserver scenarios, respectively.

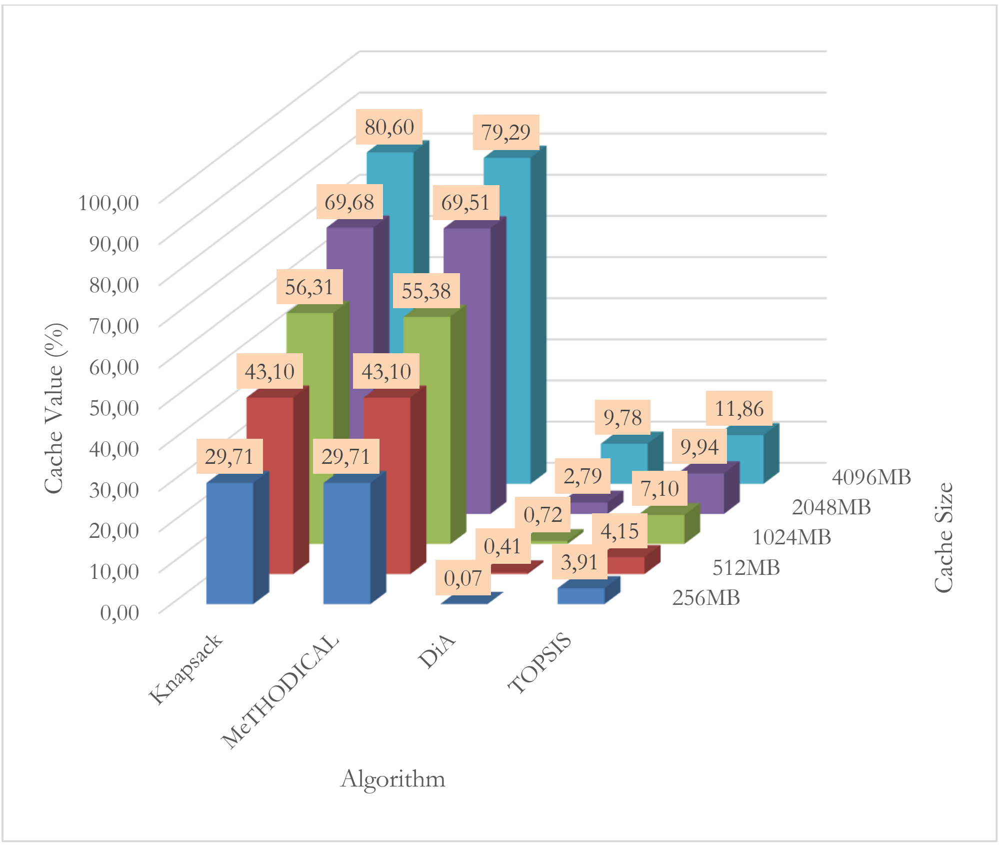


Figure 20 – Percentage of value achieved in the Normal scenario

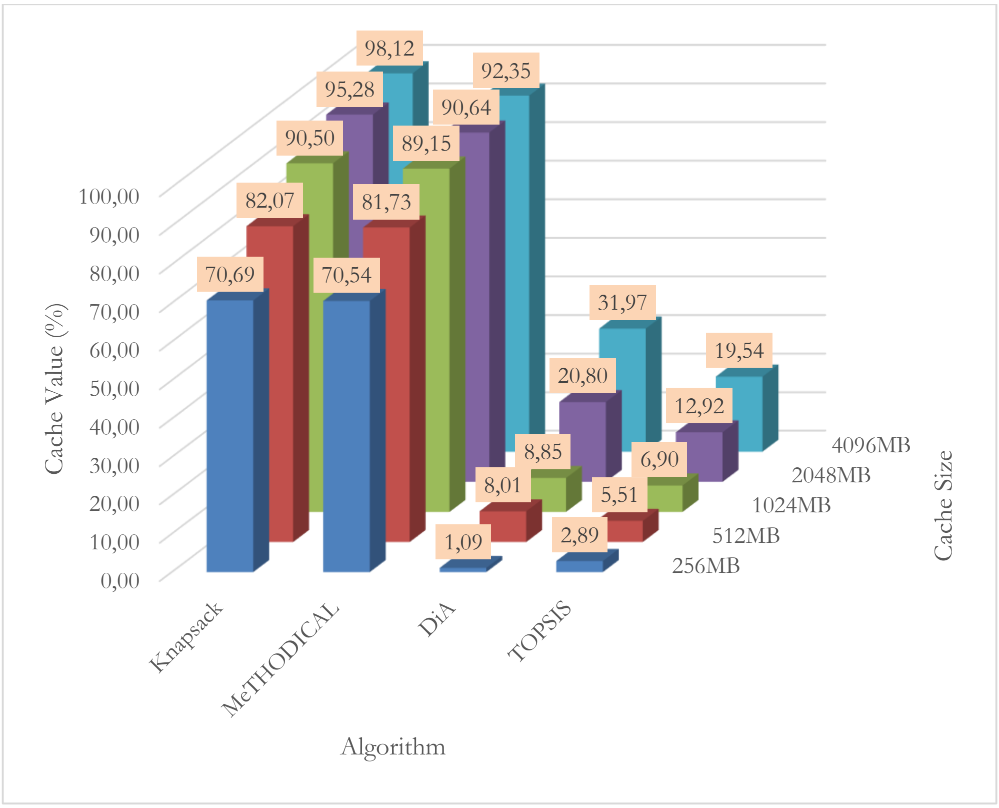


Figure 21 – Percentage of value achieved in the YouTube scenario

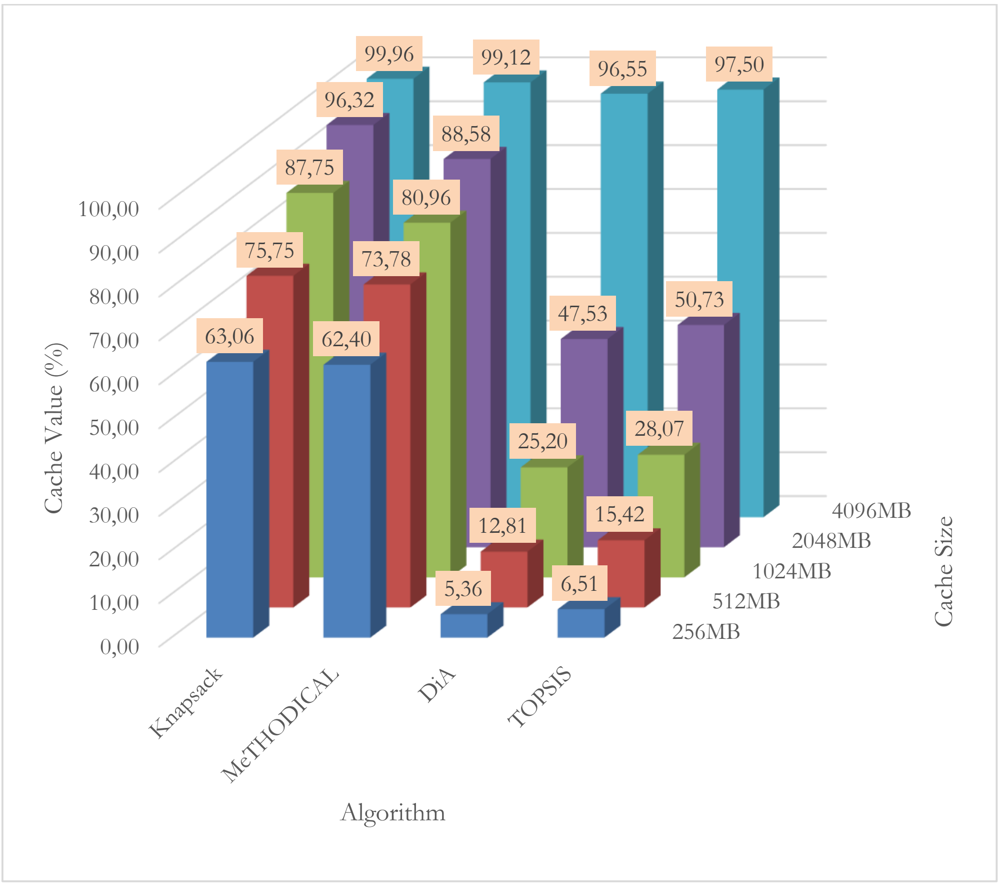


Figure 22 – Percentage of value achieved in the Webserver scenario

By analysing the graphics and correlating these results with the percentage of correct files identified and moved, it is possible to see that the MeTHODICAL algorithm outperforms with a large difference the other two MADM algorithms. Indeed, DiA and TOPSIS can only achieve a high value when the cache size is big enough to accommodate almost all the files from the scenario, as it happens with the Webserver scenario with a cache size of 4GB, as it can be seen in Figure 22. This is the reason why there is very high standard deviation in the percentage of files correctly identified, Figure 18, and in the percentage of files correctly moved, Figure 19.

At this point, based solely on the accuracy, although MeTHODICAL shows a high performance, it is still outperformed by the knapsack algorithm, but has it was stated, the efficiency was also a major aspect to take into account being this a real-time application. Table 9 presents the processing time for the different algorithms in the Normal scenario. As it can be seen, the processing time of the knapsack algorithm increases with the size of the cache, which is not the case of the MeTHODICAL and related algorithms. The same behaviour can be seen in Table 10 and Table 11, which depict the processing time of the YouTube and Webserver scenarios, respectively.

Algorithm / Cache Size	256MB	512MB	1024MB	2048MB	4096MB
Knapsack	672,84	1398,865	2845,641	5705,489	13094,158
MeTHODICAL			0,03		
DiA			0,102		
TOPSIS			0,104		

Table 9 – Processing time of the MADM algorithms in the Normal scenario (seconds)

The reason for the MADM algorithms to not present different processing times for each cache size is due to the fact that they do not determine a different solution for each case. These algorithms are deterministic, so as long as the input is the same, they always output the same result. In this case, they output an ordered list of contents to be moved, being the first element the most important to be moved. Later on, the list is iterated to add the files that can be moved until the cache size limit is reached.

Algorithm / Cache Size	256MB	512MB	1024MB	2048MB	4096MB
Knapsack	518,365	1054,535	2124,875	4262,135	8547,494
MeTHODICAL			0,023		
DiA			0,075		
TOPSIS			0,075		

Table 10 – Processing time of the MADM algorithms in the YouTube scenario (seconds)

The good accuracy of the MeTHODICAL algorithm combined with its much superior efficiency prove that this MADM algorithm is a good choice for this application, since in a real-time application, high efficiency is extremely important to deliver a good performance without the need to use a lot of resources to decrease the processing time.

Algorithm / Cache Size	256MB	512MB	1024MB	2048MB	4096MB
Knapsack	645,19	1293,765	2590,413	6371,213	12004,459
MeTHODICAL			0,031		
DiA			0,096		
TOPSIS			0,094		

Table 11 – Processing time of the MADM algorithms in the Webserver scenario (seconds)

The analysis of the results in these three scenarios allowed not only to choose the best MADM algorithm to use but also showed some other interesting results. On one hand, it is clear the impact of the popularity distributions, has it was viewed when comparing the results for the YouTube and the Webserver scenarios. On the other hand, it showed that having in-network caching capabilities, even considering medium cache sizes, for instance 512MB, could provide an increase in the users' perspective, as it can store around fifty percent of the contents requested at that location.

6.5 SCORE mechanism validation

With the SCORE mechanism fully implemented, a focused validation of the entire running mechanism was done. Due to the changes that occurred throughout the semester, this could not be done with the simulation of a large network scenario, but instead with a representative realistic scenario.

In order to validate that the SCORE mechanism can in fact be an improvement in the users' perspective, for those accessing popular contents while not increasing the delays to the other users being served by that router, a test set was designed.

In this test set, composed by multiple individual tests, two different users are considered, one that requests ten randomly selected files, User 1, following the previously considered popularity distribution, and another user, User 2, that requests the ten top files selected by the SCORE mechanism when that user moves from one router to another. The ten random selected files account for 286,26MB while the ten top files add up to 229,30MB.

Considering the Normal scenario and routers with a cache of 512MB, three different situations were considered: one where the router has the cache filled with files that neither of the users will request; another where the router has the cache populated with random files, also following the same popularity distribution; and finally one where the router has the cache populated with the popular files chosen by the SCORE mechanism, as if a migration had been triggered. The first setup represents the worst-case scenario for both users where none of the files they want are present in the local cache. With the second scenario, the typical behaviour of CCNx is represented, where some of the desired files may fortuitously be present in the local cache, and the third scenario represents the CCNx enhanced with the SCORE mechanism, which should have populated the cache with the required files.

The scenario used to perform this validation is presented in Figure 23. Three routers were considered, with User 1 and User 2 connecting to Router 1. Both Router 1 and Router 2 act as two common CCNx routers without repository. Only Router 3 has the repository populated with all the files used previously in the Normal scenario. As stated above, all the routers have a local cache of 512MB while none of the users has a local cache. This means that in the best-case scenario, the contents come from Router 1's cache and in the worst-case scenario they come from the repository in Router 3, passing through all the routers.

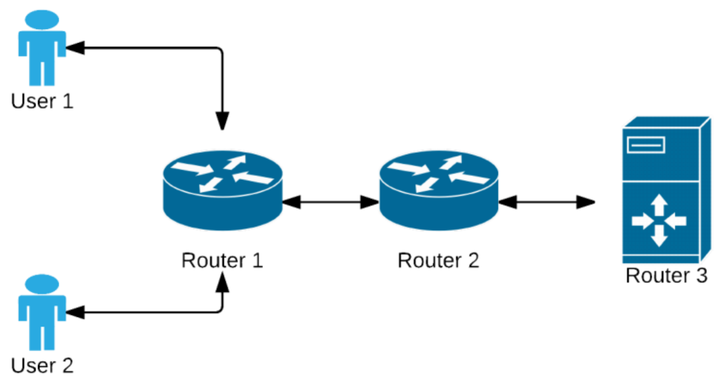


Figure 23 – SCORE validation scenario

The delay required to obtain one file is measured by considering the moment between the file being requested and the moment at which last block of the file is received. The processing time of the users between the requests of each file is not included in the delay measurement. In order to achieve more reliable, representative and statistically valid results, each test was run five times with different random seeds. There was no need to perform further repetitions, as there was no significant variation of the standard deviation.

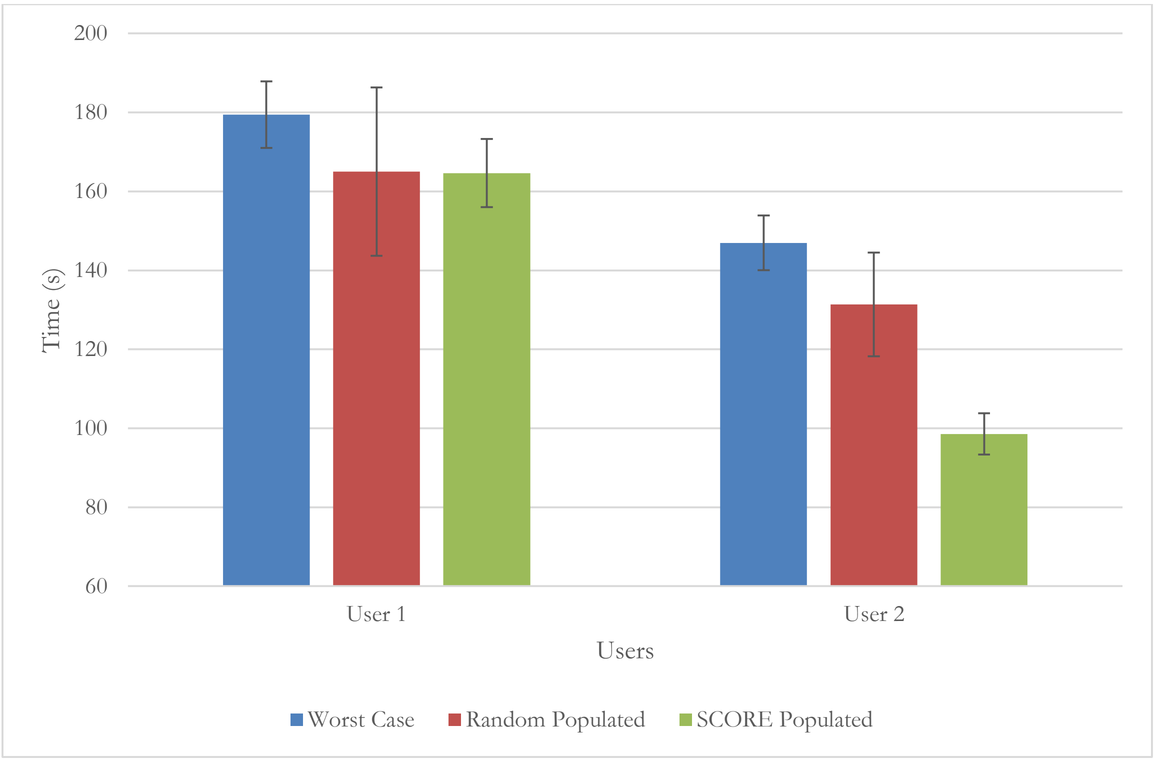


Figure 24 – SCORE validation delay results

In Figure 24 are presented the results of the validation tests in the perspective of both users. Even though these results represent the time took to obtain the ten files by each user, what is relevant is the difference experienced by each user in the different situations. These are influenced by additional mechanisms of retrieving the necessary files and that, most importantly, will be directly impacted by network conditions. The remaining time, longer or shorter, depends on the processing capability of the machine and can be disregarded.

In the studied scenario, User 1 requests ten randomly selected files, while User 2 requests the top ten most popular files identified by the SCORE mechanism. By looking at the results obtained for User 2, it is clear that reduction in the delay is achieved by the SCORE mechanism, which places all the files that are going to be requested in the local cache, opposed to the random populated cache, which may or may not contain the files requested by the user. This is the reason why the randomly populated cache shows a larger standard deviation in the results registered than the SCORE populated cache. As the name implies, in the Worst Case situation, the worst-case scenario, none of the requested files are available in the local cache, being this the reason why it presents the highest delay value.

The reduction in the delays for the users that follow the trends at each location are not the only aspect to take into account. It is also relevant to determine the impact that the SCORE mechanism causes on the delays of the users that do not follow exactly the trends. These users are represented by the User 1 that requests ten random selected files. The worst case scenario once again shows the highest delay, while in the other two cases, it is possible to see that they register almost the same delay, with the difference being in the standard deviation which is much larger for the Random Populated cache. Since it is randomly populated in each test, this means that while in one run it succeeds in having several of the requested files cached, in another one it may have few of the requested files already available at the local cache.

With this validation of the SCORE mechanism, a very important conclusion can be achieved. In a scenario where this mechanism is implemented, the quality of the perceived experience by users that follow popularity trends is improved, decreasing the delays when obtaining popular files, while not decreasing the quality of the perceived experience for the other users. It has to be taken into account that this was performed in a small scenario with only one user requesting files at a time. A larger scenario with several concurrent users would present much more expressive results and other metrics, such as the bandwidth usage and the cache hits/misses, could also be considered.

7 Conclusion and Future Work

This work intends to improve even further a new Internet concept, ICN, which itself aims to better suit the current and future expected Internet demands.

With the amount of users constantly growing, the ICN concept solved some of the identified problems with its in-network caching and content replication capabilities, thanks to the new paradigm where the focus is on the content rather than on the hosts. One of the abilities of this new paradigm is that it can cope with the mobility of users. However, there are still improvements that can be made. One of those improvements is proposed in this work, the relocation of contents based on the users' mobility in order to improve their overall satisfaction while using the Internet.

By monitoring the users' sanitized (i.e. anonymised) Internet usage and by resorting to intelligent mechanisms implemented in the network, it is possible to relocate the contents that a moving user will probably access at a new location. This is translated by an improvement in the users' experience starting right from the moment it arrives at new destinations. The mobility prediction development is out of the scope of this work, but is an important part that can integrate with all the other mechanisms created to monitor the Internet usage and to relocate the contents.

Throughout the first semester, there was a thorough analysis of the CCNx protocol and its implementation in order to design the SCORE system. After completing the design of the SCORE system and its architecture, the implementation was started. The components that perform the migration of contents were the first to be developed, followed by the necessary monitoring tools, since these represent the core of the system.

In the second semester, there was a review of the tasks to be developed, giving more importance to the study of the SCORE mechanism itself, namely the decision making component. This motivated a series of tests comparing different MADM algorithms' implementations, which lead to some issues, namely related with how to perform an evaluation of these algorithms. As a consequence of this unexpected problem, a revision of the SCORE mechanism was specified, modifying the approach initially intended. After applying the changes in the design motivated by this revision, a new series of tests were performed to evaluate the MADM algorithms and the expected performance of the SCORE mechanism.

The assessment of the SCORE mechanism allowed choosing between the best-suited algorithms for the decision-making aspect, from which the MeTHODICAL algorithm, a MADM algorithm, stood out. It also allowed to see the impact that the in-network caching capabilities of the ICN architecture can have on the users' experience, namely, in terms of delay when obtaining contents, which can be decreased with the usage of the SCORE mechanism. Another aspect that was very clear included the impact that the different popularity distributions cause, with the application of the different Zipf distributions according to each scenario.

A scientific paper is being produced to report the development done in this work. The paper is currently under internal revision, its current state can be seeing in Appendix C, but is to be submitted soon. It presents the SCORE system as well as the results of the tests made with the second specification of the SCORE mechanism.

Regarding future work, a more complete validation of the SCORE mechanism is envisaged, presenting the impact in the delay when considering repositories that are located away from the users, as implemented currently by network operators. Bandwidth usage shall also be measured in order to clearly see the impact of the SCORE mechanism, both on the network and on users' experience. The scalability of the SCORE mechanism is at this point unknown and is also expected to be the focus of a more complete assessment.

Currently, a possible extension of the project that motivated this work is being considered, motivating further developments and testing of the SCORE mechanism. Regardless of the extension of the project, the SCORE mechanism will be one of the modules on the final demonstration of the project. In fact, the SCORE mechanism was included in two project deliverables, D5.3 [39] and D5.4 [40], as a result from direct contributions provided during the development of this dissertation. In the first deliverable, the SCORE mechanism architecture and the initial development was presented, while on the second deliverable the final architecture and specification of the SCORE mechanism were included as the project's progress, as well as the results obtained with the performed tests.

8 References

[1] "The Zettabyte Era: Trends and Analysis," Cisco, 2014.

[2] J. Postel, Ed., *Internet Protocol,* STD5, RFC 791, 1981.

[3] S. Deering and R. Hinden, Eds., *Internet Protocol, Version 6 (IPv6) Specification,* RFC 2460, 1998.

[4] H. Woo, S. Han, E. Heo, J. Kim and S. Shin, "A Virtualized, Programmable Content Delivery Network," *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering,* Oxford, pp. 159-168, 2014.

[5] K. Miller and A. Wolisz, "Transport Optimization in Peer-to-Peer Networks," *2011 19th International Euromicro Conference on Parallel, Distributed and Network-based Processing,* Ayia Napa, pp. 567-573, 2011.

[6] R. Golshan, "LTE Advanced," 2011.

[7] C. E. Perkins, "Mobile IP," *Communications Magazine, IEEE,* vol. 35, no. 5, pp. 84-99, 1997.

[8] K. Leung, V. Devarapalli, K. Chowdhury and B. Patil, *Proxy Mobile IPv6,* S. Gundavelli, Ed., RFC 5213, 2008.

[9] J. Postel, Ed., *Transmission Control Protocol,* RFC 793, 1981.

[10] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan and J. Wilcox, "Information-Centric Networking: Seeing the Forest for the Trees," in *Proc. HotNets-X*, Cambridge, 2011.

[11] "Mobile Cloud Networking," [Online]. Available: http://www.mobile-cloud-networking.eu/site/. [Accessed September 2014].

[12] "Information-Centric Networking Research Group (ICNRG)," Internet Research Task Force (IRTF), [Online]. Available: https://irtf.org/icnrg. [Accessed October 2014].

[13] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture," in *Proc. SIGCOMM '07*, Kyoto, 2007.

[14] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs and R. L. Braynard, "Networking Named Content," in *Proc. CoNEXT '09*, Rome, 2009.

[15] M. Ain, "D2.3 – Architecture Definition, Component Descriptions, and Requirements," Deliverable, PSIRP 7th FP EU-funded project, 2009.

[16] M. D'Ambrosio, M. Marchisio and V. Vercellone, "D-6.2 Second NetInf architecture description," 4WARD 7th FP EU-funded project, 2010.

[17] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher and B. Ohlman, “A Survey of Information-Centric Networking,” *Communications Magazine,* vol. 50, no. 7, pp. 26-36, 2012.

[18] R. Bifulco, M. Brunner, R. Canonico, P. Hasselmeyer and F. Mir, “Scalabilty of a Mobile Cloud Management System,” in *Proc. MCC '12*, Helsinki, 2012.

[19] M. Liebsch and F. Z. Yousaf, “Runtime Relocation of CDN Serving Points - Enabler for Low Costs Mobile Content Delivery,” *Wireless Communications and Networking Conference (WCNC),* Shanghai, pp. 1464-1469, 2013.

[20] G.-H. Tzeng and J.-J. Huang, Multiple Attribute Decision Making: Methods and Applications, CRC Press, 2011.

[21] B. Sousa, K. Pentikousis and M. Curado, “MeTHODICAL: Towards the next generation of multihomed applications,” *Computer Networks,* pp. 21-40, June 2014.

[22] C.-L. Hwang, Y.-J. Lai and T.-Y. Liu, “A new approach for multiple objective decision making,” *Computers & Operations Research,* vol. 20, no. 8, pp. 889-899, October 1993.

[23] P. N. Tran and N. Boukhatem, “The Distance to the Ideal Alternative (DiA) Algorithm for Interface Selection in Heterogeneous Wireless Networs,” *Proc. MobiWac '08,* Vancouver, pp. 61-68, 2008.

[24] “NS-3,” [Online]. Available: http://www.nsnam.org/. [Accessed October 2014].

[25] “NS-3 DCE,” [Online]. Available: http://www.nsnam.org/overview/projects/direct-code-execution/. [Accessed October 2014].

[26] D. M. W. Powers, “Applications and Explanations of Zipf's Law,” in *Proc. NeMLaP3/CoNLL '98*, Sydney, 1998.

[27] R Foundation for Statistical Computing, “R: A Language and Environment for Statistical Computing,” Vienna, 2014.

[28] “PostgreSQL,” [Online]. Available: http://www.postgresql.org/. [Accessed November 2014].

[29] “rpy2,” [Online]. Available: http://rpy.sourceforge.net/.

[30] “PostgreSql + Python | Psycopg,” [Online]. Available: http://initd.org/psycopg/. [Accessed October 2014].

[31] “R: The R Project for Statistical Computing,” [Online]. Available: http://www.r-project.org/. [Accessed November 2014].

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

[33] “CHART OF THE DAY: Half Of YouTube Videos Get Fewer Than 500 Views,” 5 May 2009. [Online]. Available: http://www.businessinsider.com/chart-of-the-day-

youtube-videos-by-views-2009-5?IR=T. [Accessed March 2015].

[34] T. Yu, C. Tian, H. Jiang and W. Liu, "Measurements and Analysis of an Unconstrained User Generated Content System," *IEEE International Conference in Communications (ICC),* Kyoto, pp. 1-5, 2011.

[35] "HTTP Archive," [Online]. Available: http://httparchive.org/interesting.php. [Accessed January 2015].

[36] "Recommended upload encoding settings," [Online]. Available: https://support.google.com/youtube/answer/1722171?hl=en. [Accessed March 2015].

[37] A. Abhari and M. Soraya, "Workload generation for YouTube," *Multimedia Tools and Applications,* vol. 46, no. 1, pp. 91-118, January 2010.

[38] A. Williams, M. Arlitt, C. Williamson and K. Barker, "Web Workload Characterization: Ten Years Later," *Web Content Delivery,* vol. 2, no. 1, pp. 3-21, 2005.

[39] M. Corici, "D5.3: Final Implementation of IMSaaS, DSN and Mobile Platform," Deliverable, MCN 7th FP EU-funded project, 2015.

[40] G. A. Carella, "D5.4: Evaluation of Mobile Platform, IMSaaS and DSN," Deliverable, MCN 7th FP EU-funded project, 2015.

[41] T. M. Bohnert and A. Edmonds, "D2.2: Overall Architecture Definition," Deliverable, MCN 7th FP EU-funded project, 2013.

[42] I. Aad, "D5.1: Design of Mobile Platform Architecture and Services," Deliverable, MCN 7th FP EU-funded project, 2013.

[43] T. Taleb, "D4.1: Mobile Network Cloud Component Design," Deliverable, MCN 7th FP EU-funded project, 2013.

9 Appendix A

Mobile Cloud Networking (MCN)[2] is a Large-scale Integrating Project funded by the European Commission Seventh Framework Programme (EU FP7). The project started in November 2012 for the period of 36 months being SAP[3] the project coordinator and ZHAW[4] the technical leader with a total of 19 partners from industry and academia. The aim of the MCN project is to define and evaluate Europe's vision of mobile cloud computing.

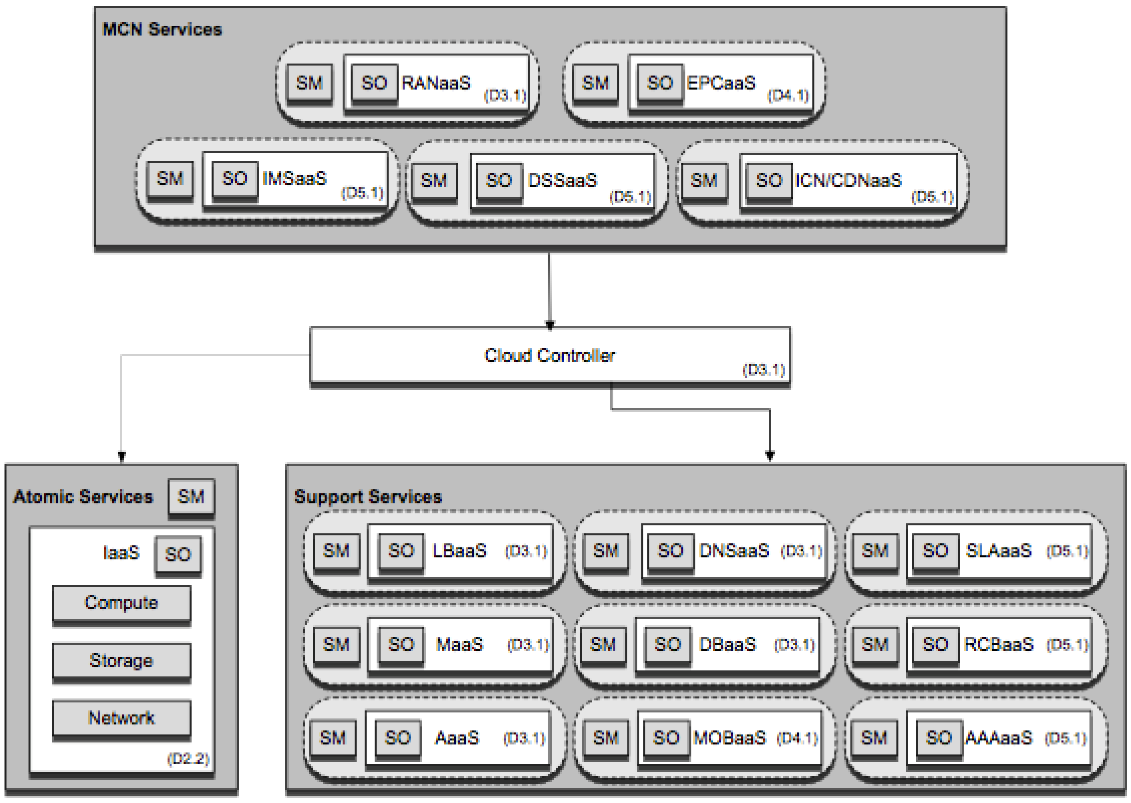


Figure 25 – MCN Entities [41]

In Figure 25 are depicted all the services that form the MCN project, as well as the Deliverables where each of them are presented. The two services that are relevant to this work are ICN/CDNaaS [42], where the work developed in this dissertation is included, and MOBaaS [43].

ICN/CDNaaS combines the features of both Content Delivery Networks and Information-Centric Networking. This service will deploy CDN serving points through the network that will work together with the ICN to serve the user requests. When a user sends a request for a content object to an ICN router, it will serve the content from its own cache, or if necessary, it can connect to a CDN serving point and forward the request that came from the user.

[2] http://www.mobile-cloud-networking.eu/site/
[3] http://www.sap.com/corporate-en/about.html
[4] http://www.zhaw.ch/en/

As it can be seen in Figure 25, MOBaaS is considered a MCN support service. This service will predict information regarding three different aspects: the movement of individual end users, its location in a future moment in time, the traffic that each end user will be generating at a certain location, and the bandwidth available at a certain location in a future moment in time. Being a support service, the information it generates can be used by any of the other services, as the one developed in this dissertation, to trigger the migration of contents based on the users' movement.

10 Appendix B

In this appendix are presented the results of the tests with the first specification of the SCORE mechanism in the other scenarios. First are presented the graphics comparing with each individual metric, and then are presented the graphics with the merged results for each test.

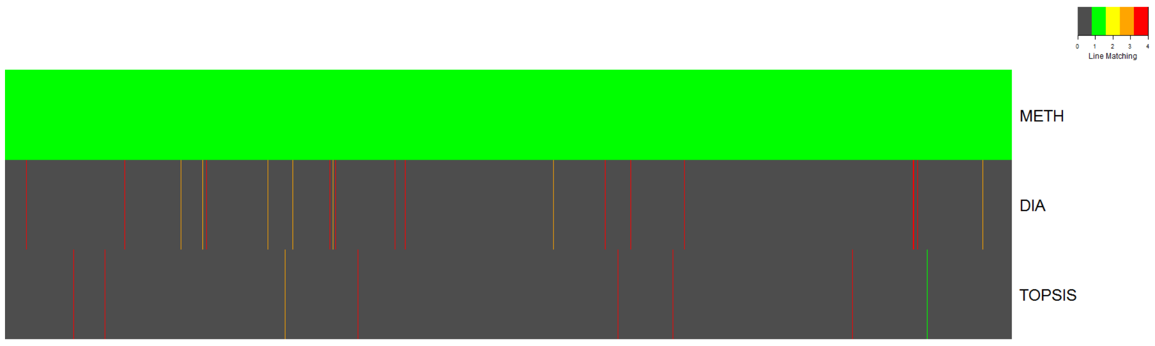


Figure 26 – Results of test 1 in the Webserver scenario with 2000 files

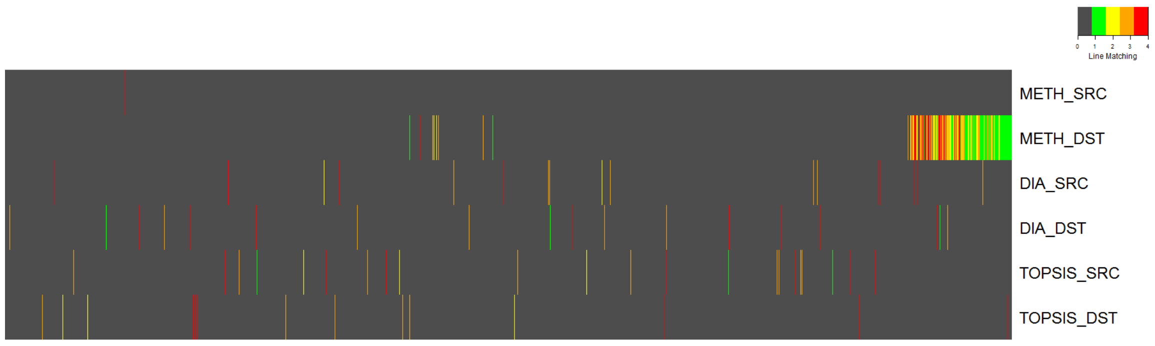


Figure 27 – Results of test 2 in the Webserver scenario with 2000 files

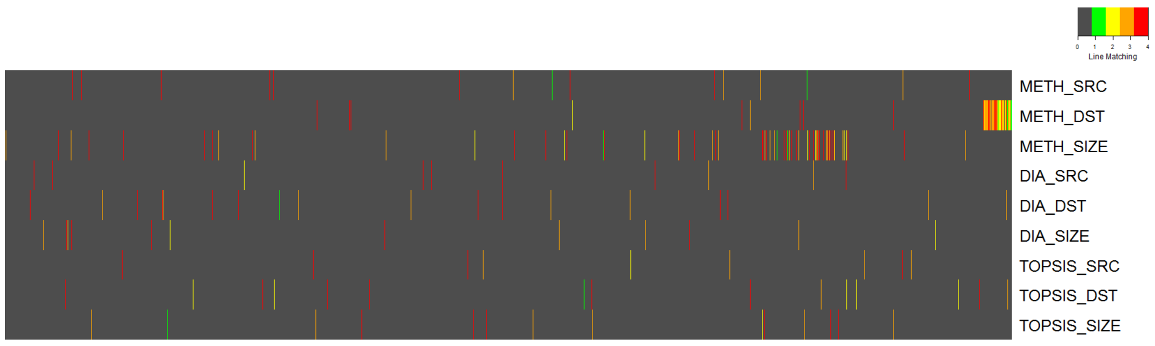


Figure 28 – Results of test 9 in the Webserver scenario with 2000 files

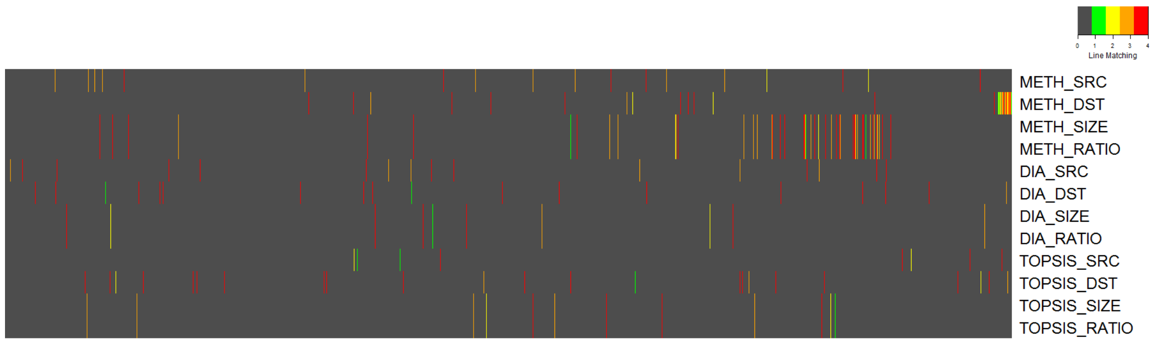


Figure 29 – Results of test 13 in the Webserver scenario with 2000 files

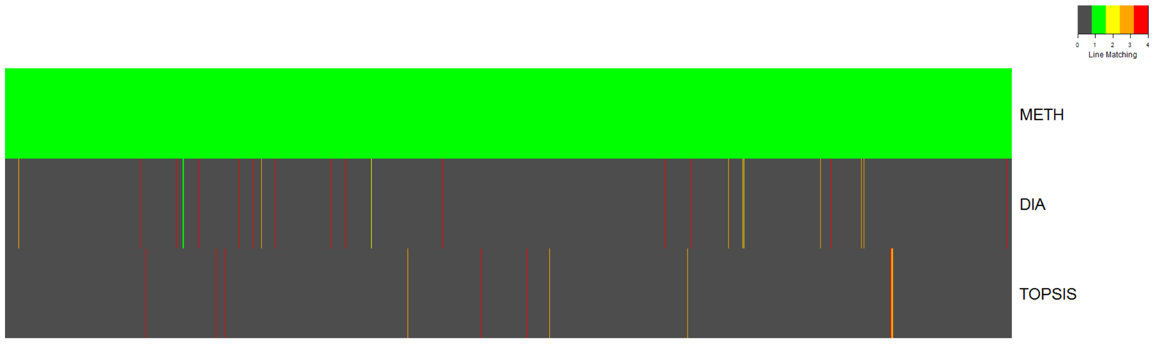


Figure 30 – Results of test 1 in the YouTube scenario with 2000 files

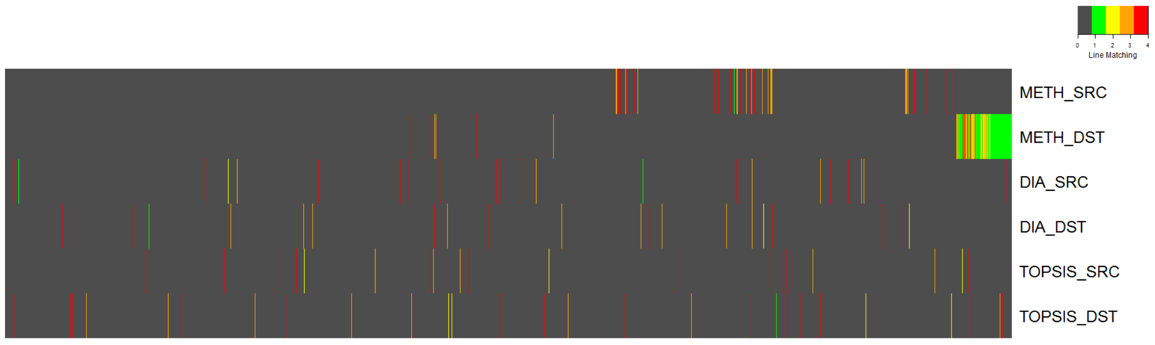


Figure 31 – Results of test 2 in the YouTube scenario with 2000 files

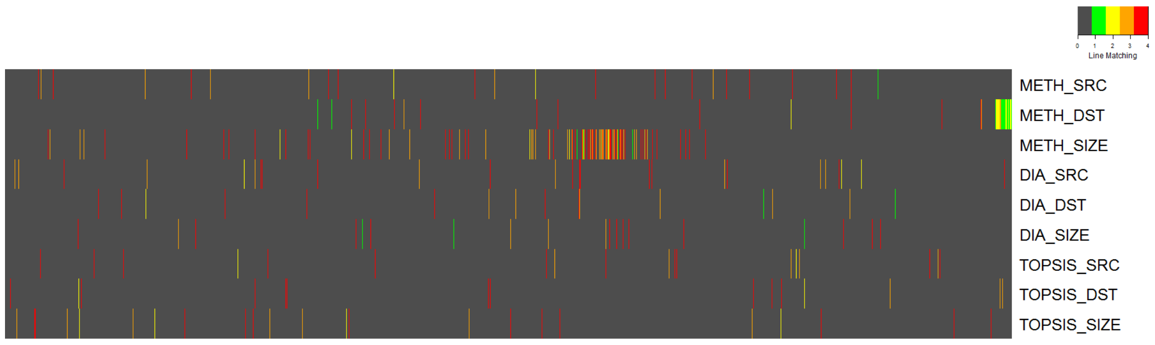


Figure 32 – Results of test 9 in the YouTube scenario with 2000 files

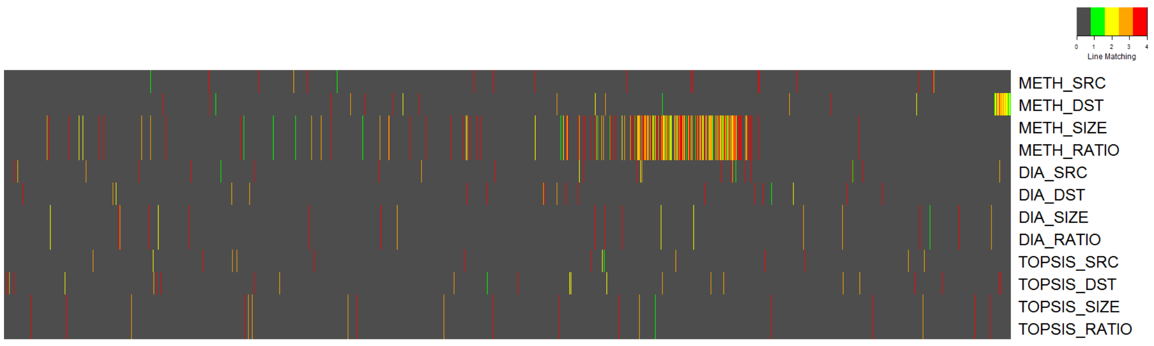


Figure 33 – Results of test 13 in the YouTube scenario with 2000 files

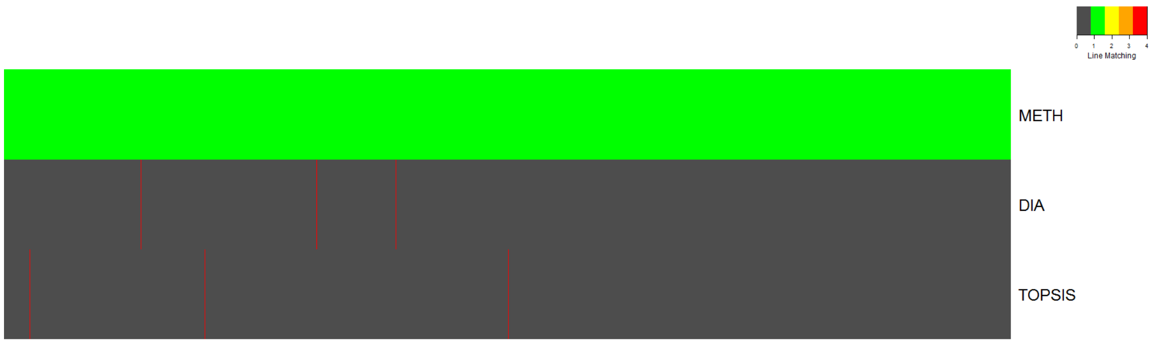


Figure 34 – Results of test 1 in the Webserver scenario with 20000 files

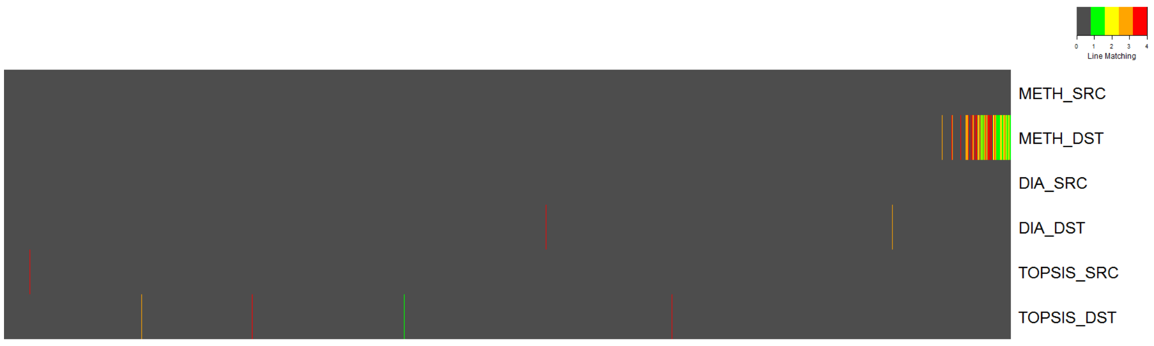


Figure 35 – Results of test 2 in the Webserver scenario with 20000 files

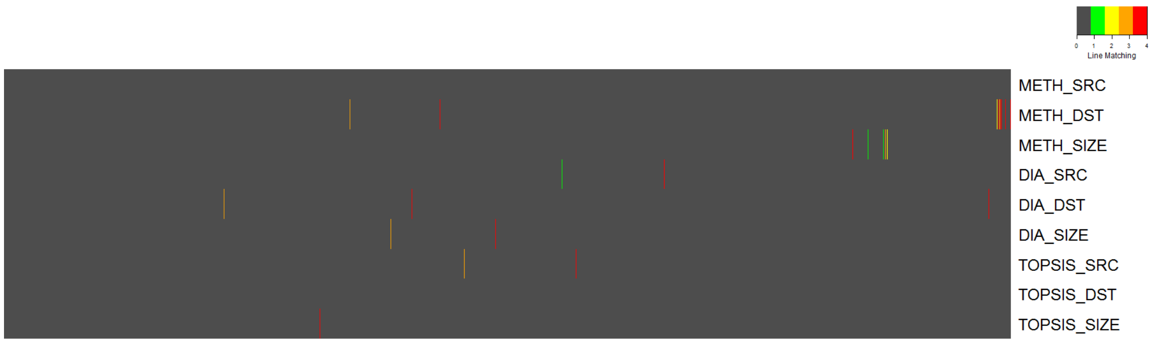


Figure 36 – Results of test 9 in the Webserver scenario with 20000 files

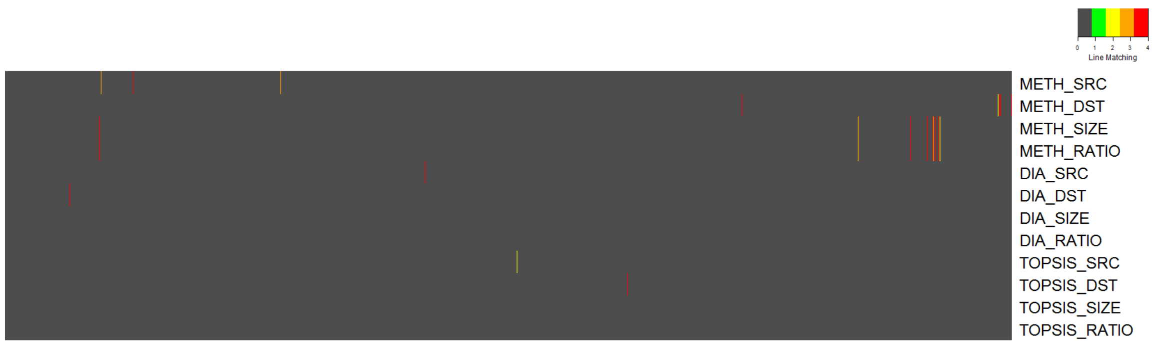


Figure 37 – Results of test 13 in the Webserver scenario with 20000 files

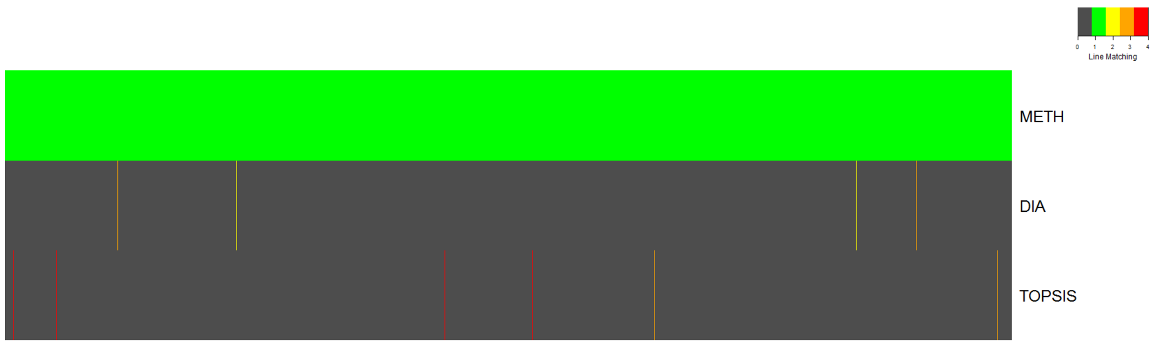


Figure 38 – Results of test 1 in the YouTube scenario with 10000 files

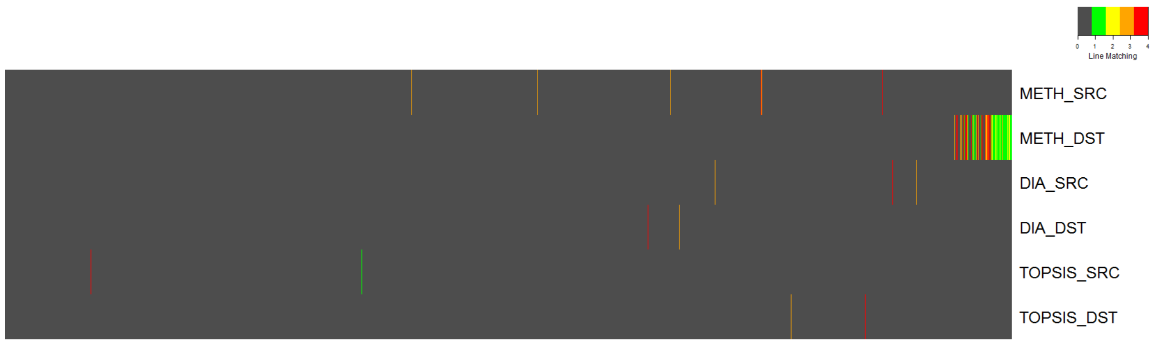


Figure 39 – Results of test 2 in the YouTube scenario with 10000 files

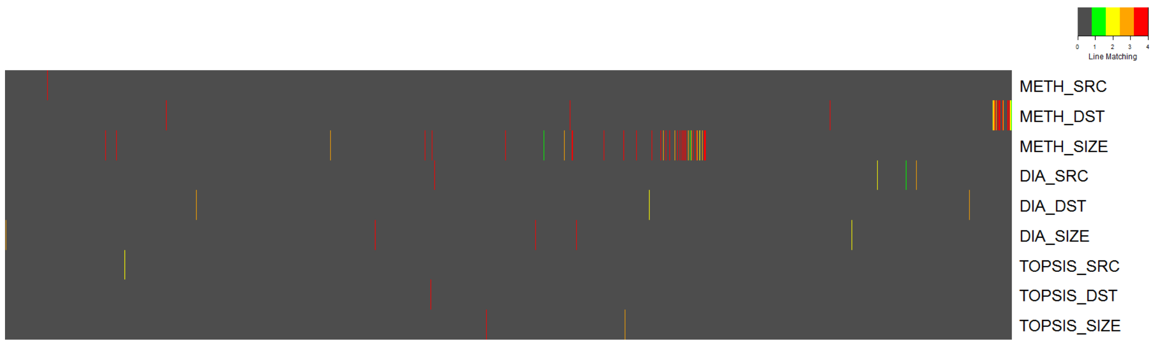


Figure 40 – Results of test 9 in the YouTube scenario with 10000 files

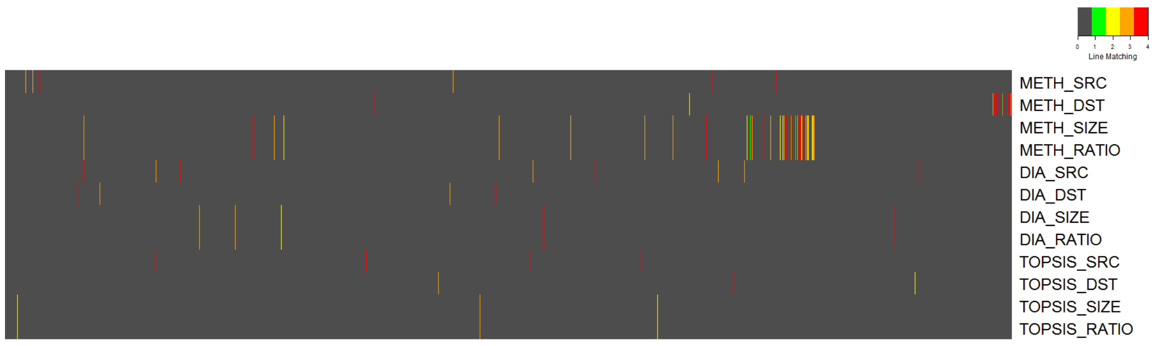


Figure 41 – Results of test 13 in the YouTube scenario with 10000 files

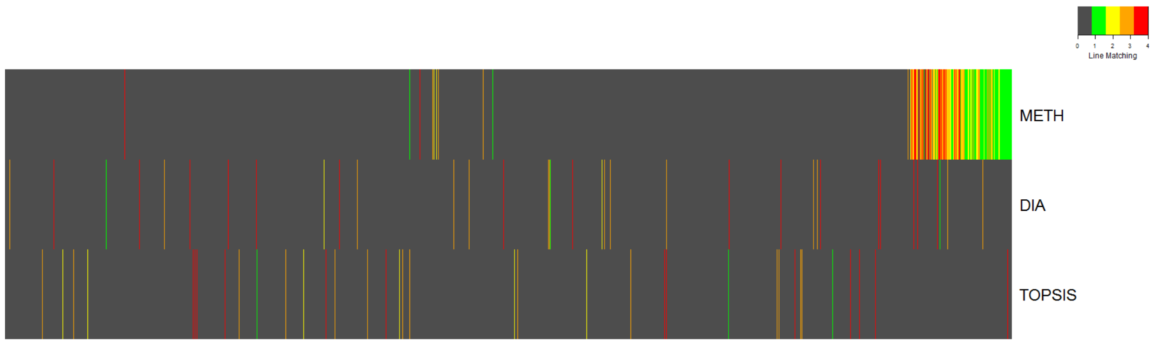


Figure 42 – Merged results of test 2 in the Webserver scenario with 2000 files

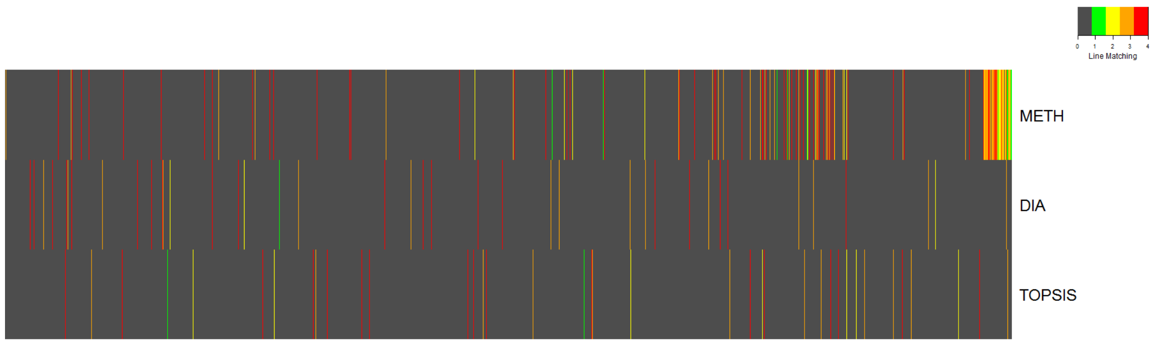


Figure 43 – Merged results of test 9 in the Webserver scenario with 2000 files

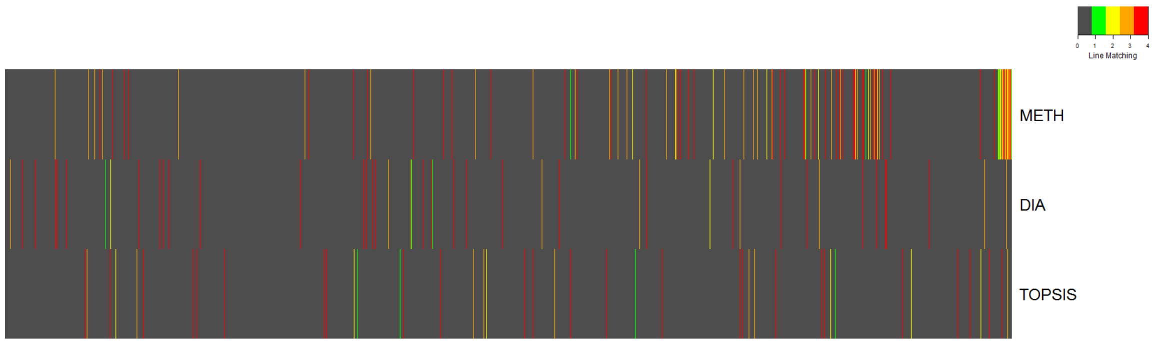


Figure 44 – Merged results of test 13 in the Webserver scenario with 2000 files

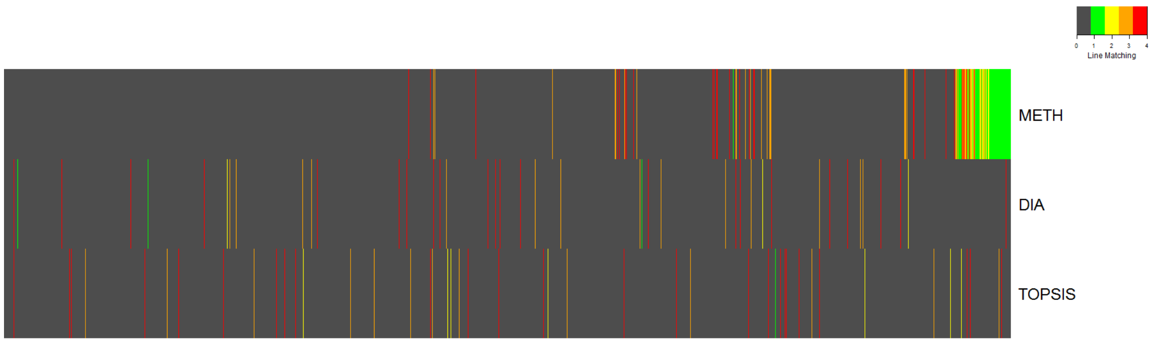


Figure 45 – Merged results of test 2 in the YouTube scenario with 2000 files

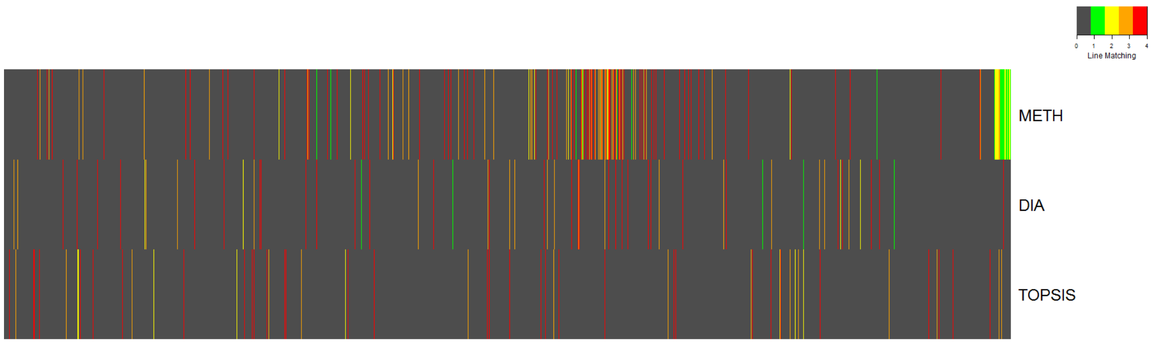


Figure 46 – Merged results of test 9 in the YouTube scenario with 2000 files

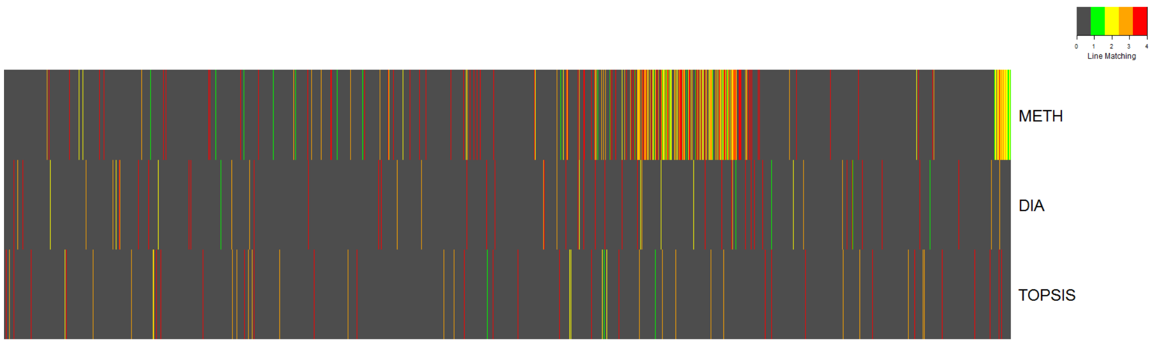


Figure 47 – Merged results of test 13 in the YouTube scenario with 2000 files

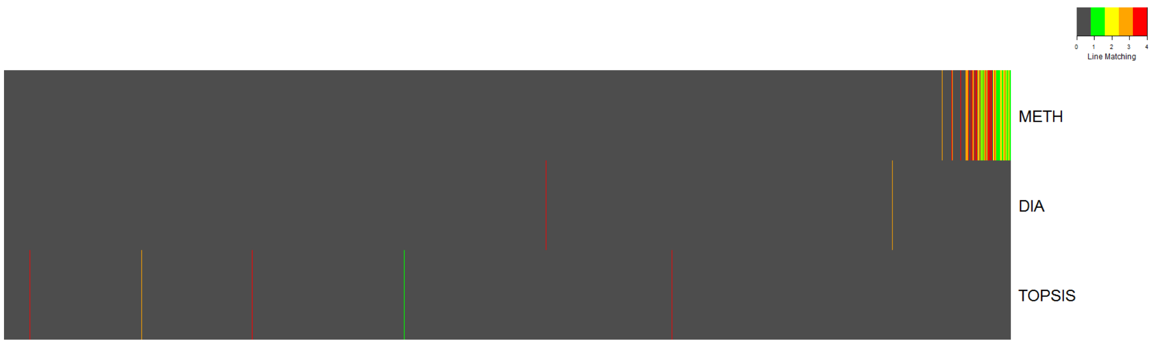


Figure 48 – Merged results of test 2 in the Webserver scenario with 20000 files

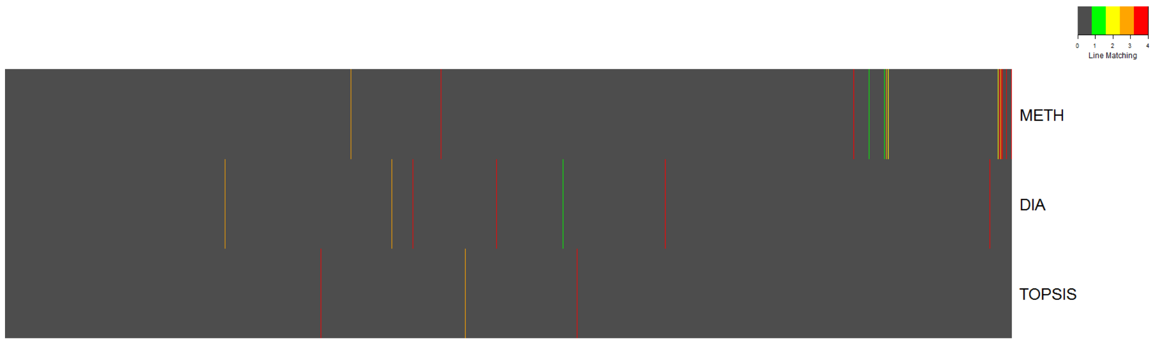


Figure 49 – Merged results of test 9 in the Webserver scenario with 20000 files

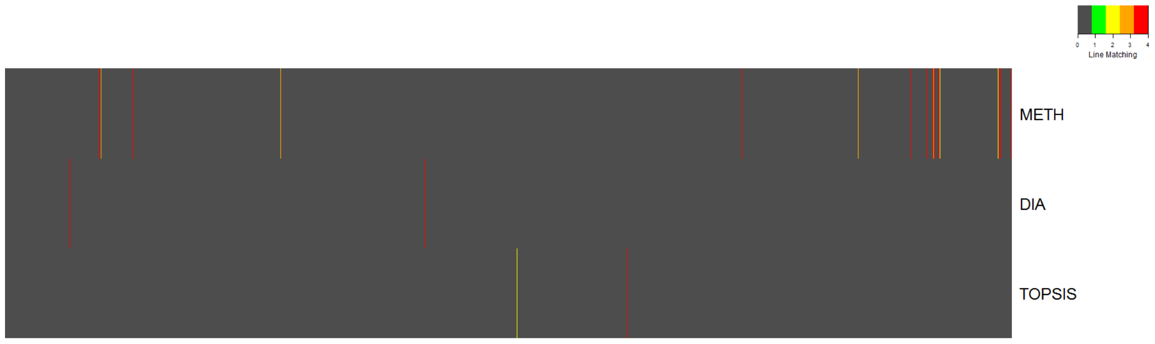


Figure 50 – Merged results of test 13 in the Webserver scenario with 20000 files

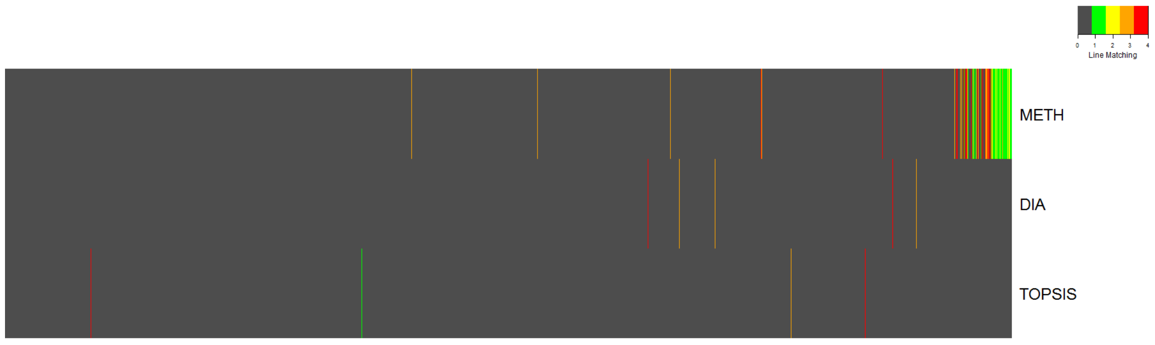


Figure 51 – Merged results of test 2 in the YouTube scenario with 10000 files

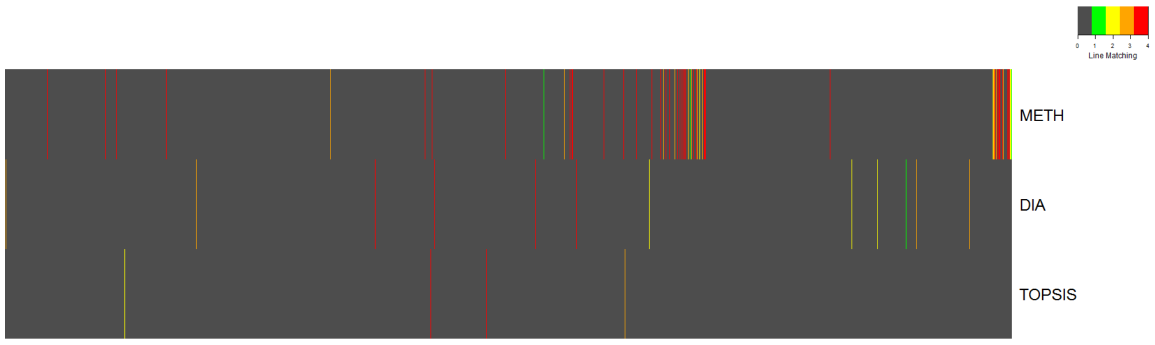


Figure 52 – Merged results of test 9 in the YouTube scenario with 10000 files

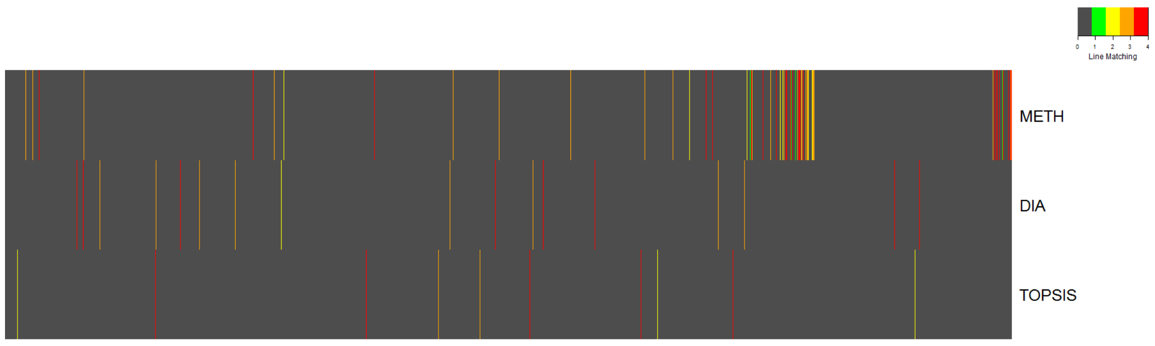


Figure 53 – Merged results of test 13 in the YouTube scenario with 10000 files

11 Appendix C

Please turn the page.

A Mobile Follow-Me Cloud Content Caching Model

Andre Gomes[1 2], Vitor Fonseca[2], Bruno Sousa[1], David Palma[1], Paulo Simoes[2], Edmundo Monteiro[2] and Luis Cordeiro[1]

[1]OneSource, Consultoria Informática Lda.
Coimbra, Portugal
{gomes,bmsousa,palma,cordeiro}
@onesource.pt

[2]Department of Informatics Engineering
University of Coimbra, Portugal
{asng,fonsecav,psimoes,edmundo}
@dei.uc.pt

ABSTRACT

With a boom in the usage of mobile devices for traffic-heavy applications, mobile networks struggle to deliver good performance while saving resources to support more users and save on costs. In this paper, we propose a model to handle content caches migration at the edge of cloudified mobile networks based on popularity and user mobility, leveraging both the concept of Information-Centric Networking and the benefits brought by the mobile cloud computing paradigm. With such a system, caches at the edge of mobile networks are optimized in terms of resources usage and content also becomes closer to users interested in it. Results show that proposed strategies are efficient and have a high rate of accuracy when identifying which content objects should be migrated between caches, enabling larger cache hit rates, lower content access latencies and bandwidth savings at the core of the mobile network.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.4 [**Computer Systems Organization**]: Performance of Systems—*Design studies*

General Terms

Experimentation, Performance

Keywords

Information-Centric Networking, Content Migration, Caching, Mobile Cloud.

1. INTRODUCTION

Mobile network 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 tremendous demand of bandwidth [1], 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 with such dynamic networks is still an open challenge, which is currently being addressed by a number of emerging concepts and technologies.

As far as the network is concerned, new 5G concepts such as Network Function Virtualization (NFV) [2] are emerging, allowing networks to adapt more dynamically to different conditions and requirements. One of these efforts is Cloud Radio Access Network (C-RAN) [3][4]. It brings the possibility to virtualize the entire 3GPP LTE radio infrastructure, except for the antennas. Virtualized infrastructures extend the cloud computing concept to the Radio Access Network (RAN), and explore the modularity of the components together with the usage of general-purpose hardware infrastructure to run evolved Node Bs (eNBs). Such fact transforms C-RAN into an enabler for deployment of value-added services closer to the edge of the network, i.e. in very close proximity to mobile users.

At the same time, Future Internet (FI) concepts such as Information-Centric Networking (ICN) [5], which proposes a change in the current paradigm of requesting content, are becoming increasingly important. Such fact is explained by the advantages brought together. Currently, when a user wants to request a content object, it has to query one specific server to request a given resource. With ICN, the user has no knowledge about servers and just requests the network for content. It first sends an Interest message that usually specifies the content provider and the name of the content object. 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, without any further interaction. This approach has multiple advantages over traditional Internet Protocol (IP) networks. First, performance is greatly improved. This is truly relevant in current mobile networks where resources are scarce and traffic loads have exploded. Content retrieval is much quicker due to faster lookups and because of the inherent caching, which is native in ICN. Ultimately, this also contributes to bandwidth savings, which then translate into cost savings for operators. Second, mobility support is greatly improved [6], and it becomes easier to deal with a globalized world where everyone is constantly on the move. Finally, ICN also brings advantages in terms of security [7], a factor with even greater importance today due to the rising levels of cybercrime.

With such concepts in mind, proposals appear to take advantage of their combination. Gomes et al. [8] evaluate the feasibility of deploying ICN together with LTE mobile networks, leveraging the C-RAN concept and its role as an enabler for the deployment of additional services within these networks. In that work, authors conclude that there are clear benefits of deploying ICN routers co-located with LTE eNBs, such as bandwidth savings at the core network and lower latency to retrieve content derived from the proximity to end users. At the same time, the impact at the processing of LTE frames is low, and it also becomes visible that ICN provides extra benefits when comparing to HyperText Transfer Protocol (HTTP) even when caching mechanisms provided by both are considered.

With ICN-based caching at the edge of 3GPP LTE networks at its inherent benefits [9], one may assume that major performance improvements and reduction of traffic at the core of the network are already achieved by default. However, efficient caching strategies are needed in order to populate those caches and maintain content where it will yield the most benefit. As users are increasingly mobile and tend to move between different locations quite often, it is safe to assume that what is cached at a location is not necessarily what is going to be requested by the users that will be there in the next few hours or days. That fact leads to the question: how does user mobility affect caching strategies? Such question does not have a simple answer, as some assumptions have to be made and challenges need to be considered to reach a preliminary conclusion. First, it is important that user mobility is predicted to perform preemptive actions, and proposals exist to deal with it [10][11][12]. Then, if a set of users is predicted to be at a location, decisions need to be made. The first decision is whether content from caches at the origin of the users should be migrated to other caches at the possible destinations. Once that is established, an important question arises: which subset of the content should be migrated?

In this paper, we attempt to answer that question by analyzing great amounts of collected information and applying decision making strategies. When decisions are made, content can then be migrated and end users will experience much lower content access latencies on average. At the same time, mobile network operators will save on network resources and therefore save on costs or even allow for a greater number of users to experience a high quality service.

In section 2, existing proposals to address content and services migration are analyzed. Section 3 introduces our model for content following the user in a mobile cloud environment. Section 4 describes experimentation scenarios for the evaluation of the proposed model. Section 5 presents the results of the performed experiments. Finally, in section 6, the main achievements of this work are highlighted.

2. RELATED WORK

When considering strategies that take into account the mobility of users to decide on placement/migration of services or content within mobile networks, only a few works are actually feasible and valid.

One proposal assumes that mobility of the user is considered for services placement and scaling [13]. 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 Data Center (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. [14]. 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 [15][16].

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. [17] look at the problem in Peer-to-Peer (P2P) networks from the provider perspective. Considering a typical hierarchical Content Delivery Network (CDN) 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 that already maintain caches hierarchically based on local popularity [18].

Another proposal, by Vasilakos et al. [19], 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.

Considering all the proposals and the issues they fail to address, we propose a system that relies on their positive findings and at the same time attempts to address the challenges not taken into consideration.

3. MOBILE FOLLOW-ME CLOUD MODEL

The mobile FMC model consists of two parts: the overall ICN service running on the cloud, named ICN as a Service (ICNaaS), and the FMC components. This architecture is depicted below in Fig. 1 and it is described in the following subsections.

3.1 ICNaaS

ICNaaS is a service aimed at deploying an ICN infrastructure in a cloud environment, leveraging the advantages of cloud principles and pushing the boundaries of existing content delivery technologies. This service is first introduced as part of the Mobile Cloud Networking project [20], and detailed in related deliverables [21][22][23].

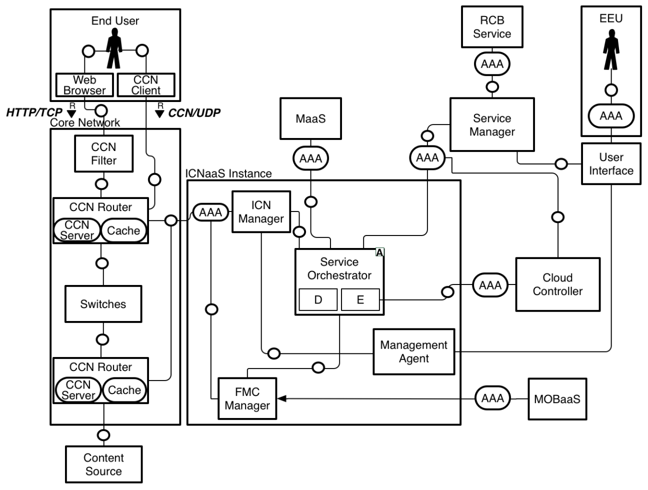


Figure 1: ICNaaS Overall Architecture

It consists of 6 main components: CCN Routers, CCN Filter, ICN Manager, Service Orchestrator, Service Manager and Management Agent. Initially, when an Enterprise End User (EEU) wants an instance of the service, it contacts the Service Manager (SM). This component has a catalog of ICN services that can be offered, and upon request will deploy a Service Orchestrator (SO) by contacting the Cloud Controller (CC), i.e. component that manages the cloud platform. Once the SO is deployed, it will use its Execution (E) sub-component to deploy all the remaining components of the service instance. First it will deploy the ICN Manager (component responsible for the management of the ICN layer), the CCN Filter (converts HTTP requests into ICN Interests), the CCN Routers (routers implementing a subset of ICN functionalities, in particular CCN) the FMC components (to be described below) and the Management Agent (interface for the EEU to have deeper control of the service instance). At the same time, the SM handles dependencies of the ICN service. A first dependency is Monitoring as a Service (MaaS), used to monitor components and provide information to the Decision (D) sub-component of the SO for scaling in and scaling out decisions. The second dependency is Mobility Prediction as a Service (MOBaaS), used by the FMC components to get input related with user mobility, either predicted or detected. A third and last dependency is Rating, Charging and Billing as a Service (RCBaaS), used by the SM to charge and bill the EEU for the resources used by its services instances.

3.2 FMC Components

FMC has two key components, one that runs at every router, CCNServer, and a main one that is responsible for all the decision making and control of the routers actions, FMCManager.

The CCNServer is responsible to send all the monitoring data to the FMCManager so it can be stored at the centralized database and is also responsible to migrate the contents when the FMCManager sends that action. The monitoring is done thanks to a minor change in the code of the CCNx framework.

The FMCManager stores all the monitoring data coming from the routers in the centralized database, and when a migration is needed, it fetches the necessary data from the database and determines the list of contents that the router should have in its cache to better serve its current users and also the arriving ones. It then sends that list to the router telling it to migrate those contents.

The Follow-Me Cloud requires an accurate and efficient decision mechanism, no matter the scenario where it is operating. Therefore, the decision mechanism takes into consideration two metrics: content popularity and content size, since the local caches of routers have size limitations. The content popularity is formulated based on three parameters, which include the content popularity at the sources (popSrc/reqSrc) and destination routers (popDst/reqDst); the number of users moving to the destination cell per source cell (movU); and the number of users at the destination router (dstU). Eq. 1 depicts the formula that determines the content popularity.

$$popContent_i = \frac{\sum_{k=1}^{N}(\frac{popSrc_{i,k}}{reqSrc_k} * movU_k) + \frac{popDst_i}{reqDst} * dstU}{N+1} \quad (1)$$

4. EVALUATION

The validation of the optimization algorithm to be employed in Follow-Me Cloud is described in the following paragraphs. According to the requirements of the FMC, accurateness and efficiency, the choice falls into a Multiple Attribute Decision Mechanism (MADM). The first goal is assessed by employing the knapsack algorithm that determines the best theoretical solution, the one that provides the content to be copied to the destination router's cache in order to ensure low content access latency to the most requested content and, at the same time maximizes the profit by fulfilling the cache of routers. Indeed, the optimization of FMC follows the optimization pursued in the 0/1 knapsack problem, where given the size of the destination router's cache, the contents to fulfill the same must be selected under the constraints of the content popularity and content size. The Multiple Attribute Decision Mechanism (MADM) algorithms have been considered since the first specification of FMC, mainly due to the flexibility and efficiency of such mechanisms for providing optimal solutions independently the scenario. As such, three MADM algorithms can be applied to perform the decisions of optimizing the content to be copied: MeTHODICAL [24], TOPSIS [25] and DIA [26]. The validation of such algorithms is performed according the Follow-Me Cloud requirements: First, Accurateness, assessed by the number of correct files identified by the MADM mechanism when compared with the best solution, and the value they can achieve with the files that they decide to move compared to the maximum possible value determined by the knapsack algorithm. Finally, Efficiency, assessed in terms of processing time to determine the optimal content to be copied.

In particular the accurateness, besides considering the number of correct files also considers the percentage of value in cache, which is determined by considering the number of requests for a certain content and the total number of requests performed in the source and destination routers. Regarding the evaluation of the correct files, two analysis are made. The first one measures the percentage of files that each algorithm correctly identifies when compared to the knapsack results. The second analysis identifies the percentage of files that are correct in the total of the files to be moved based on the algorithms decisions. The evaluation of the value in cache is determined by comparing the value obtained by each algorithm with the maximum possible value achieved for each cache size determined with the knapsack algorithm.

Different sizes are considered for the cache size, namely size 256 and 512MB, 1, 2 and 4GB. Since the cache size is one crucial aspect, it is stored in RAM to reduce the content access latency. By comparing these algorithms in three testing scenarios described hereafter, it is possible to select the most appropriate algorithm for the Follow-Me Cloud decision system.

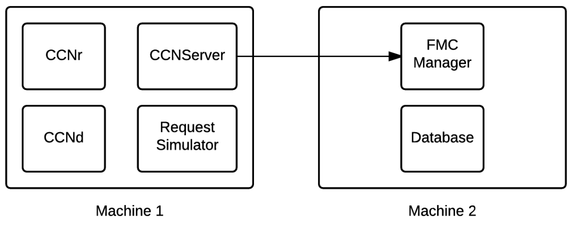


Figure 2: Components of FMC evaluation scenario

Figure 2 depicts the machines and the components of the FMC to perform the evaluation. The request simulator component was configured to perform the requests according to the configurations of each scenario. A machine was dedicated to the FMC manager, where the optimization algorithms are placed. The CCNServer receives the commands to perform the migration of content according to the decisions of the FMC Manager.

The Normal, the YouTube and the Webserver scenarios, summarized in Table 1, were specified to evaluate the Follow-Me Cloud decision mechanism algorithms. The content popularity was defined according to the Zipf distribution (as specified by the literature [27] considering the specificities 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 [28] and [29]. For instance, it has been demonstrated that the majority of the files are unpopular while only a few amount of files are extremely popular. Finally, the total number of files content objects and the content distribution per class follow the distribution that is characteristic of each scenario.

In the normal scenario, the content object's size follows a Normal Distribution with mean 30.6MB and variance of

Table 1: **FMC configuration parameters**

Parameter	Normal	Youtube	WebServer
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 $\alpha = 1.8$ $\beta = 5500$ 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

15.72MB. This distribution is based on a survey of current Internet statistics, such as the average size of a Webpage (2 MB) [30] and the average size of 1 minute YouTube video with a resolution of 720p (40MB) [31]. The second scenario, YouTube, follows an already defined model [32]. As for the third scenario, Webserver, a model was defined based on the observed growth of the size of files available at file webservers [33]. These models consider a gamma distributions with $\alpha = 1.8$ for both, and $\beta = 500$ for the YouTube model andor $\beta = 1200$ for the YouTube model and the Webserver model, respectively.

5. RESULTS

The Follow-Me Cloud evaluation does not focus on the scaling performance (i.e. increasing the number of FMC Managers or other components), but is devoted to the requirements of FMC, namely accurateness and efficiency.

Table 2: Knapsack Results per Scenario

Cache Size	Normal		Youtube		WebServer	
	Content objects	Value in Cache (%)	Content objects	Value in Cache (%)	Content objects	Value in Cache (%)
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	80,6	558	98,12	1980	99,95

In Table 2 one may observe the results obtained with the knapsack algorithm for each scenario and for the diverse sizes of cache that have been considered. These results are the ideal solution for each combination of scenario and cache size. With these results, it is possible to better understand the impact that the popularity distributions cause on each scenario, where, for instance, 47 files in the YouTube scenario account for 71% of the requests received, while to achieve a similar value in the Webserver scenario 366 files are required.

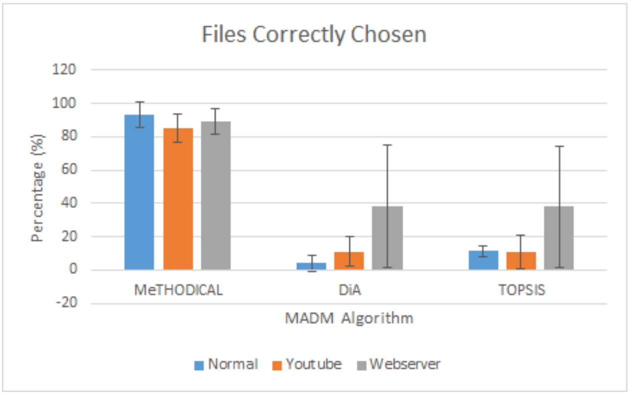


Figure 3: Percentage of files correctly identified

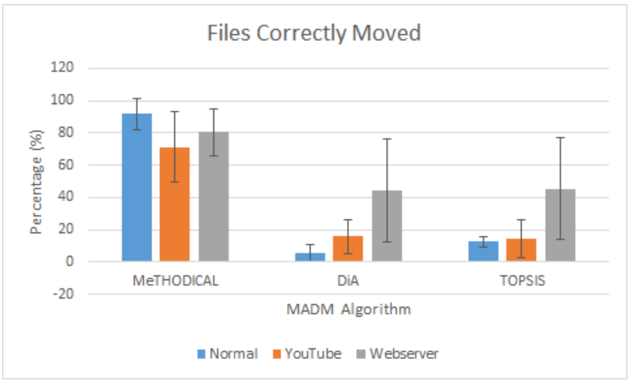


Figure 4: Percentage of files correctly moved

In Figure 3, it is presented the percentage of the files that each MADM algorithm correctly identified when comparing to the ones selected by the knapsack algorithm. Figure 4 on the other hand, presents the percentage of files that are correctly moved. It is very clear that within these three scenarios, not only the MeTHODICAL algorithm has a very high performance, but also delivers a consistent performance independently of the cache size. As for the other two MADM mechanisms, they deliver a much lower performance together with far less consistency (high standard deviation). Although they might seem very similar, these two metrics evaluate two distinct things that are very important. On one hand, it is important to see if the the MADM mechanisms have chosen all or most of the correct files, but on the other hand, it is also important that they don't choose also a long list of incorrect files, since this would mean that a lot of unnecessary files will be place at the router's local cache.

Following the trend set by the percentage of files correctly chosen, in Figure 5, it is shown the percentage of value each algorithm can achieve with the files it chooses to move, compared to the highest possible value determined by the knapsack solution, in the Normal scenario. Figure 6 and 7 depict the same reasoning for YouTube and WebServer scenarios, respectively.

By analysing the graphics, and correlating it with the num-

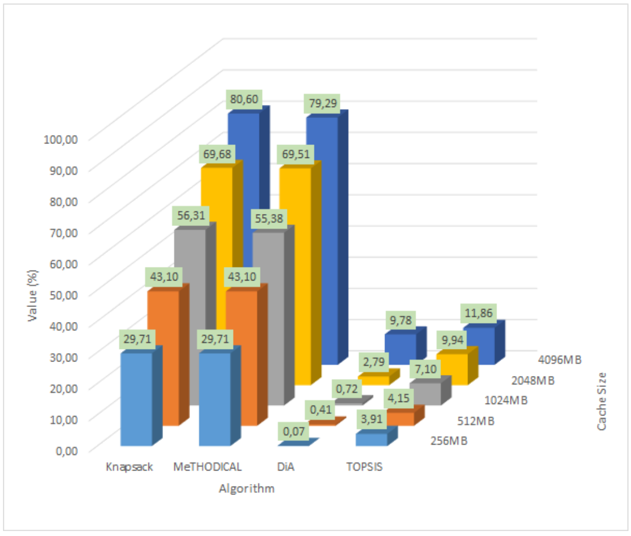


Figure 5: Percentage of value achieved in the Normal Scenario

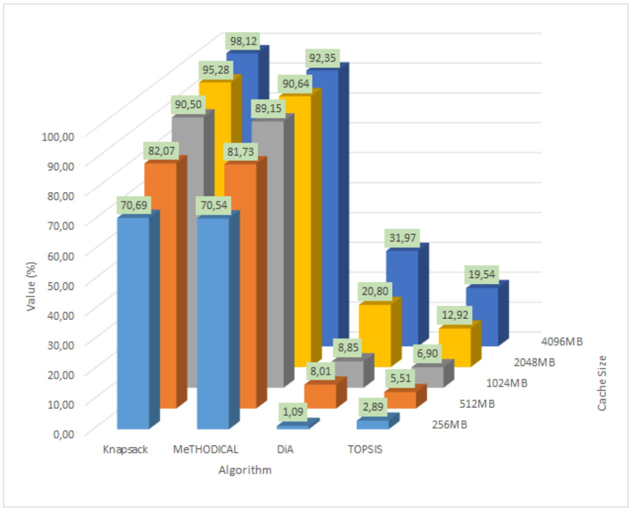


Figure 6: Percentage of value achieved in the YouTube Scenario

ber of correct files, it is possible to see that the MeTHODICAL algorithm outstands in comparison to similar mechanisms. Indeed, DiA and TOPSIS have only a high percentage when the cache size is able to accommodate all the files, as happens with the WebServer scenario, depicted in Figure 7.

The accuracy requirement leads to the selection of Knapsack and MeTHODICAL, nonetheless when considering the efficiency requirement the choice falls into MeTHODICAL. Table 3 depicts the processing time for the different algorithms in the YouTube scenario. The processing time of Knapsack increases with the size of the cache, which is not the case of MeTHODICAL and related algorithms. This is possible with the MADM algorithms because they don't need to determine a different solution for each cache size. Based on the inputs they output and ordered list, and after

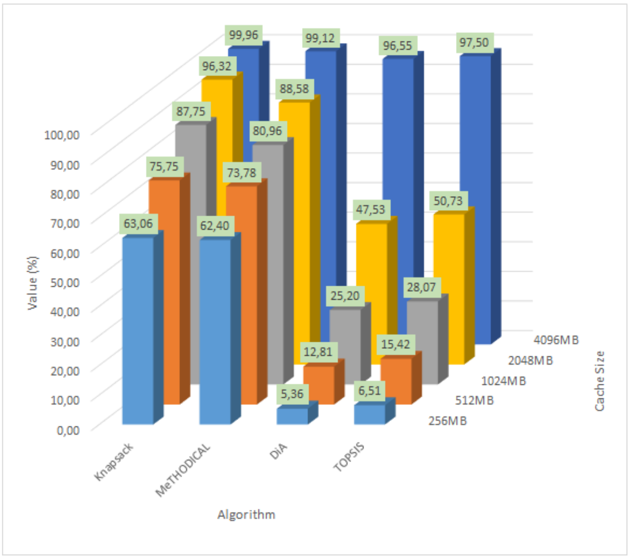


Figure 7: Percentage of value achieved in the Webserver Scenario

that, based on the cache size, the first 50 or 100 files, for example, are selected maximizing the local cache usage. This high efficiency of the MeTHODICAL algorithm combined with its good accuracy prove that it is a good choice for this application, since in a real-time application, high efficiency is extremely important to deliver a good performance.

Table 3: Processing Time of FMC decision algorithms (seconds)

Algorithm / Cache Size	**256MB**	**512MB**	**1GB**	**2GB**	**4GB**
Knapsack	518,365	1054,535	2124,875	4262,135	8547,494
MeTHODICAL			0,023		
DiA			0,075		
TOPSIS			0,075		

6. CONCLUSIONS

In this paper, a model and strategies for the migration of content within mobile networks were introduced, enabling multiple benefits both from the user and network perspective. As the users move to different locations, they still want to access content on which they are interested with a 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 content kept in those caches (with limited resources) is the right content, i.e. popular content that local users are very interested on.

A number of proposals to handle this issue already exist, and were described thoroughly in Section 2. However, they cannot be applied to content (only to services) or have other limitations, often assuming a very specific scenario or scope and not handling important issues or considering certain requirements. Therefore, we propose a broader approach to deal with content migration, handling decisions with multiple criteria and deciding multiple factors that will trigger content migration of a given subset of content.

The proposed model was evaluated in multiple scenarios in terms of the percentage of files correctly identified for migration and value achieved by those migrations. Results show that the selected algorithm is quite efficient and very accurate, resulting in a smooth operation with real-time applications. With such a model, the goal of delivering content with lower latency to end users while saving well-valued network bandwidth can therefore be achieved.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Seventh Framework Programme Mobile Cloud Networking project (FP7-ICT-318109).

8. REFERENCES

[1] Cisco Systems Inc. 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`, Feb 2015.

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

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

[4] Bernd Haberland, Fariborz Derakhshan, Heidrun Grob-Lipski, Ralf Klotsche, Werner Rehm, Peter Schefczik, and Michael Soellner. Radio Base Stations in the Cloud. *Bell Labs Technical Journal*, 18(1):129–152, 2013.

[5] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. 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, 2009. ACM.

[6] Do-hyung Kim, Jong-hwan Kim, Yu-sung Kim, Hyun-soo Yoon, and Ikjun Yeom. 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, 2012. ACM.

[7] Diana Smetters and Van Jacobson. Securing network content. Technical report, PARC, October 2009.

[8] A. Gomes and T. Braun. 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, April 2015.

[9] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce Maggs, K.C. Ng, Vyas Sekar, and Scott Shenker. Less pain, most of the gain: Incrementally deployable icn. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 147–158, New York, NY, USA, 2013. ACM.

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

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

[12] Yohan Chon, Hyojeong Shin, E. Talipov, and Hojung Cha. 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, March 2012.

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

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

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

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

[17] Haiqin Liu, Yan Sun, and Min Sik Kim. Provider-level content migration strategies in p2p-based media distribution networks. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 337–341, Jan 2011.

[18] Carlos Anastasiades, Andre Gomes, Rene Gadow, and Torsten Braun. Persistent Caching in Information-Centric Networking. Technical Report IAM-15-001, IAM, University of Bern, Switzerland, May 2015.

[19] Xenofon Vasilakos, Vasilios A. Siris, George C. Polyzos, and Marios Pomonis. 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, 2012. ACM.

[20] MCN Consortium. EC FP7 Mobile Cloud Networking project. `https://www.mobile-cloud-networking.eu/`, May 2015.

[21] Imad Aad et al. Design of Mobile Platform Architecture and Services. Technical Report D5.1, MCN Consortium, November 2013.

[22] Yago Ruiz et al. Implementation of IMSaaS, DSN and

Mobile Platform. Technical Report D5.2, MCN Consortium, May 2014.

[23] Marius Corici et al. Final Implementation of IMSaaS, DSN and Mobile Platform. Technical Report D5.3, MCN Consortium, January 2015.

[24] Bruno Sousa, Kostas Pentikousis, and Marilia Curado. Methodical: Towards the next generation of multihomed applications. *Computer Networks*, 65:21–40, June 2014.

[25] Ching-Lai Hwang, Young-Jou Lai, and Ting-Yun Liu. A new approach for multiple objective decision making. *Computers & Operations Research*, 20(8):889–899, October 1993.

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

[27] Dario Rossi and Giuseppe Rossini. Caching performance of content centric networks under multi-path routing (and more). 2011.

[28] Chart of the day: 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`.

[29] Tianlong Yu, Jiang Chen, and Wenyu Liu. Measurements and analysis of an unconstrained user generated content system. *IEEE Internation Conference in Communications (ICC)*, pages 1–5, 2011.

[30] Http archive. `http://httparchive.org/interesting.php`.

[31] Recommended upload encoding settings. `https://support.google.com/youtube/answer/1722171?hl=en`.

[32] Abdolreza Abhari and Mojgan Soraya. Workload generation for youtube. *Multimedia Tools and Applications*, 46(1):91–118, January 2010.

[33] Adepele Williams, Martin Arlitt, Carey Williamson, and Ken Barker. Web workload characterization: Ten years later. *Web Content Delivery*, 2(1):3–21, 2005.