Aalto University
School of Science
Master's Programme in ICT Innovation

Morteza Neishaboori

Implementation and Evaluation of Mobile-Edge Computing Cooperative Caching

Master's Thesis
Espoo, July 28, 2015

Supervisor: Professor Antti Ylä-Jääski, Aalto University

Advisor: Teemu Kämäräinen M.Sc. (Tech.)

Aalto University
School of Science

Aalto University
School of Science
Master's Programme in ICT Innovation

ABSTRACT OF
MASTER'S THESIS

Author: Morteza Neishaboori

Title:
Implementation and Evaluation of
Mobile-Edge Computing Cooperative Caching

Date: July 28, 2015 **Pages:** 54

Professorship: Data Communications Software **Code:** T-110

Supervisor: Professor Antti Ylä-Jääski

Advisor: Teemu Kämäräinen M.Sc. (Tech.)

Recent expanding rise of mobile device users for cloud services leads to resource challenges in Mobile Network Operator's (MNO) network. This poses significant additional costs to MNOs and also results in poor user experience. Studies illustrate that large amount of traffic consumption in MNO's network is originated from the similar requests of users for the same popular contents over Internet. Therefore such networks suffer from delivering the same content multiple times through their connected gateways to the Internet backhaul. On the other hand, in content delivery networks(CDN), the delay caused by network latency is one of the biggest issues which impedes the efficient delivery and desirable user experience.

Cooperative caching is one of the ways to handle the extra posed traffic by requesting popular contents repeatedly in MNO's network. Furthermore Mobile-Edge Computing(MEC) offers a resource rich environment and data locality to cloud applications. This helps to reduce the network latency time in CDN services. Thus in this Thesis an aggregation between Cooperative Caching and MEC concept has been considered.

This Thesis demonstrates a design, implementation and evaluation for a Mobile-Edge computing Cooperative Caching system to deliver content to mobile users. A design is presented in a failure resilient and scalable practice using a light-weight synchronizing method. The system is implemented and deployed on Nokia Networks Radio Application Cloud Servers(Nokia Networks RACS) as intelligent MEC base-stations and finally the outcome of the system and the effect on bandwidth saving, CDN delay and user experience are evaluated.

Keywords: mobile edge computing, cooperative caching, content delivery networks, Nokia Networks RACS, MEC, CDN

Language: English

Acknowledgements

I would like to give my special thanks to my supervisor Prof. Antti Ylä-Jääski and my adviser and instructor Teemu Kämäräinen for their help and patience to guide me through different stages of this Thesis. I thank Petteri Pöyhönen and Timo Knuutila from Nokia Networks for sharing their experience on Nokia Networks RACS which made everything easier. I also appreciate the help of my colleague Eduardo Castellanos in different parts of the deployment process.

In addition I want to express my deepest gratitude to all the professors and staff members of University of Rennes 1, Aalto University and EIT Digital programme, especially to Stéphanie Halochet for her unlimited and unconditional help and to prof. Jukka K. Nurminen for believing in me when it was the most needed.

And finally, I dedicate this work to all my loved ones, my parents and my little sister.

Espoo, July 28, 2015

Morteza Neishaboori

Abbreviations and Acronyms

MCC	Mobile Cloud Computing
MEC	Mobile-Edge Computing
QoS	Quality of Service
CDN	Content Delivery Network
RAN	Radio Access Network
MNO	Mobile Network Operators
LTE	Long Term Evolution
VM	Virtual Machine
API	Application Interface
MEC CO-Caching	Mobile-Edge Computing Cooperative Caching
RACS	Nokia Networks Radio Application Cloud Servers

Contents

Chapter 1

Introduction

Significant demand of users to have different types of services on their mobile devices and the related issues of these devices like energy consumption, poor resources and low connectivity [32] has led to the appearance of Mobile Cloud Computing (MCC) to try to offload the application tasks and also necessary storage to a cloud server. MCC mitigates certain challenges of mobile devices in a desirable manner. For example energy efficiency and reliability are reached by several intelligent solutions, mainly by offloading methods.

Despite these achievements there are still issues like low bandwidth, high latency, service availability, quality of service (QoS) and service cost [14] to be addressed. These concerns arise mostly from rapid growth in the number of mobile users and their expectations of MCC services. Bandwidth is limited in wireless networks compared to normal wired networks. Users need more availability despite mobile devices lack of connectivity and they demand higher QoS with less service cost. Also network latency is a big burden in improving QoS and user experience while using a distant cloud. These problems are more tangible in applications that offer cognition or virtual reality services which demand low latency and high bandwidth [33].

Considering these problems, researchers realized utilizing resources and services with more locality is more cost efficient with better availability, faster connectivity and less latency. This has led to the concept of "Cloudlet" [33]. A cloudlet is a computer or a cluster of computers connected at the edge of the network to provide low-latency access to computing resources for mobile devices. The mission of cloudlets is to alleviate resource constraints of mobile devices and also to reach better network latency. Speech recognition, natural language processing, computer vision and graphics, machine learning, augmented reality and other computation-intensive applications would benefit the most from the cloudlet approach [33][19].

Realizing the advantages of bringing cloud services and resources closer,

Radio Access Network (RAN) Operators started to study the idea of cloudlets on RAN base stations as the very first hop of network edge to serve mobile users. Serious efforts have begun making this idea practical for business environment. This has led to the emergence of Mobile-Edge Computing (MEC).

In ETSI's executive briefing [5], MEC is defined as follows: "Mobile-edge Computing offers application developers and content providers cloud-computing capabilities and an IT service environment at the edge of the mobile network. This environment is characterized by ultra-low latency and high bandwidth as well as real-time access to radio network information that can be leveraged by applications."

As mentioned, meeting the demanding quality-of-service (QoS) requirements of emerging real-time, interactive and media services has led to MEC paradigms, where the focus is on delivering high QoS, at low operational costs. In addition, the fast growth of mobile users utilizing these services poses significant network traffic growth which needs to be addressed according to limited backhaul bandwidth capacity and mobile applications desirable response time.

Employing cooperative caching in the edge of the network enhances data locality and helps to balance network workload. With the emergence of mobile-edge computing, it is now possible to deploy cooperative caching approaches at the edge of network in proximity of mobile users by utilizing MEC base-stations. In this way, by decoupling the time in which a content is downloaded from a distant cloud from the time which the content is delivered to the mobile user, a Mobile-Edge Computing Cooperative Caching network can boost the service experience and reduce the latency while saving the network backhaul resources.

In other words, Mobile-Edge Computing Cooperative Caching(MEC Co-Caching) refers to the use of cooperative caching in intelligent base-stations which are equipped with MEC server capabilities, to store content at the very first hop from end users. For instance, if a mobile user requests a popular content that is cached, he will receive the content directly from the relevant base-station rather than the relevant CDN server or original distant cloud server until the session or cache record consistency expires. This however requires the base-stations to cooperate in content caching and distribution.

1.1 Problem statement

Each Mobile Network Operator(MNO) utilizes few gateways to conduct the mobile user traffic out of their inter-network toward the Internet backbone.

This infrastructure architecture combined with increasing number of mobile users leads to network bandwidth and resource challenges [15][39]. Also large amount of traffic consumption on the Internet is caused by requesting popular contents by users. As an example in Youtube, 80% of views are dedicated only to 10% of popular contents [9]. In this way MNO's networks suffer from delivering the same content multiple times while it's not necessary [38]. Equally, this congestion causes poor QoS and user experience. Cooperative caching is one of the ways to handle this extra traffic in edge networks. On the other hand, content delivery networks can suffer drastically from network latency caused by the distance from the original cloud server or middle proxy server. MEC brings the opportunity of having rich resources and more data locality for mobile users. Cloud services can benefit from more locality to reduce the network latency using MEC. MEC Co-Caching tries to address these issues in a cooperative caching system implemented on MEC. While there are studies on caching in cell networks, the efforts have been mostly on theoretical analysis by modeling. In addition there are decisions to be made on how to deploy a MEC cooperative caching system. In such system, base-stations should be able to synchronize themselves with as less overhead as possible. There should be also approaches for cache records consistency.

1.2 Objectives of the Thesis

This Thesis demonstrates a design, implementation and evaluation for a Mobile-Edge computing Cooperative caching system which delivers content to mobile users. The main focus is on the study of feasibility and performance of using MEC approach for cooperative caching in a real world implementation. A simple content delivery application is implemented which uses the designed MEC cooperative caching system to handle the caching and to reduce the bandwidth usage in the MNO's network. Later on, the performance of delivering the content, bandwidth saving and user experience are measured and analyzed.

1.3 Structure and focus

The rest of the thesis is structured as follows: Chapters 2 and 3 introduce the background study and necessary concepts toward implementation of the Mobile-Edge computing cooperative caching system. Chapter 2 summarizes the concepts of Mobile Cloud Computing (MCC), Mobile-Edge Computing (MEC) and intelligent Base-Stations. Chapter 3 overviews different perspec-

tives on cooperative caching in mobile related networks. Different approaches have been studied for cooperative caching, such as, cashing on mobile nodes, caching on intermediate or proxy nodes or caching on the edge of network. Chapter 4 is dedicated to provide deeper details in MEC cooperative caching in general and in particular the design characteristics of the implemented system. Consecutively in Chapters 5 and 6 we present the implementation plan, deployment environment and evaluation results. Finally the Thesis concludes with discussion and conclusions in Chapters 7 and 8 respectively.

Chapter 2

Mobile-Edge computing

Chapters 2 and 3 introduce the background and necessary concepts toward the implementation of the Mobile-Edge computing cooperative caching system. This section summarizes the concepts of Mobile Cloud Computing (MCC), Mobile-Edge Computing (MEC) and Intelligent Base-Stations.

2.1 Mobile cloud computing

In recent years we are witnessing significant demand for users to have different types of cloud services on their mobile devices. For instance, services in entertainment, social networking, business, news, games or health and well being [16]. However this demand faces mobile devices with issues like low energy, poor resources and low connectivity [32]. To address this, the term Mobile Cloud Computing (MCC) came to light and researchers try to define the boundaries and give proper definitions.

There are several existing definitions for Mobile Cloud Computing. In general, it is a running service on a resource rich cloud server which is used by a thin mobile client [16]. It can also be referred when mobile nodes play as a resource provider role in a peer-to-peer network [24]. Likewise as it's mentioned in Fernando et al. in [16] we can consider MCC as a network with certain characteristics. We can take the need for adaptability, scalability, availability and self-awareness in cloud computing concept [25] and expand it to mobile cloud computing.

Alternatively, MCC could be defined in a more comprehensive way as it is quoted from MCC Forum [2] in Dinh et al. [14] as follows: "Mobile cloud computing at its simplest, refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from

mobile phones and into the cloud, bringing applications and MCC to not just smartphone users but a much broader range of mobile subscribers".

MCC tackles certain challenges of mobile devices in a desirable manner. Energy efficiency is reached by several solutions like intelligent access to disk or screen [30] and mainly by offloading techniques. Various researches demonstrate that offloading the tasks of mobile applications to remote cloud can notably save energy [26] and reduce the energy consumption of the mobile device. In addition, data storage capacity and processing power is improved through storing and accessing big data on the cloud [14]. Also we can have more reliability by storing our data on the cloud on different cloud servers.

However, despite of all improvements by MCC, there are still issues to be addressed. Issues like low bandwidth, high latency, service availability, quality of service (QoS) and service cost [14]. Bandwidth is limited in wireless networks compared to normal wired networks. Users need high availability despite mobile devices lack of connectivity and they demand better QoS and less service cost. Moreover network latency is still a big burden in improving user experience by getting the way of cloud services. These matters are more tangible in applications that offer cognition or virtual reality services which demand low latency and high bandwidth [33].

Therefore, considering the previously discussed weaknesses, utilizing resources in user proximity and improving the locality of services seems to improve the availability, connectivity and network latency. Satyanarayanan in [33] proposes the concept of "Cloudlet"to deal with these objectives which later on leads to Mobile-Edge computing.

2.2 Cloudlets

As it is depicted in Figure 2.1 cloudlet is considered as the middle tier of a 3-tier hierarchy: mobile device, cloudlet and cloud. A cloudlet can also be viewed as a resource rich center at the proximity of users. Cloudlet is connected to a larger cloud server and its goal is to bring the cloud services closer to the end-user [33].

In the cloudlet concept, mobile device offloads its workload to a resource-rich, local cloudlet. Cloudlets would be situated in common areas such as coffee shops, libraries or university halls, so that mobile devices can connect and function as a thin client to the cloudlet [16]. A cloudlet could be any first hop element at the edge of network while it has four key attributes. It has only soft state, it should be resource rich and well-connected, with low end-to-end latency and also it follows a certain standard for offloading (e.g. Virtual machine migration) [1]. In other words, a cloudlet's failure is

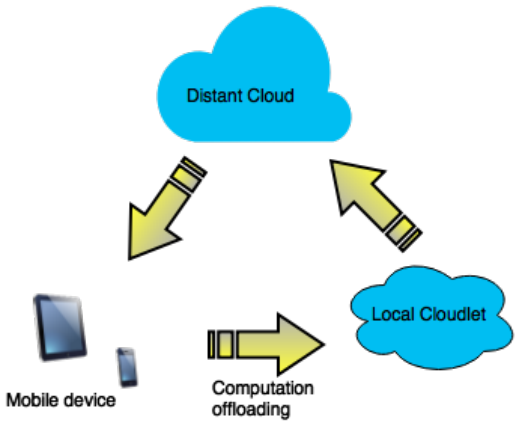


Figure 2.1: Cloudlet's architecture

not critical, it has strong internal connectivity and high bandwidth wireless LAN and it should be in logical and physical proximity of user to reduce the network latency.

There are two main approaches to implement cloudlet infrastructure using Virtual Machine (VM) technology. In both of these architectures it is important that cloudlet could go back to its beginning state after being used (e.g. by post-use clean up). A VM based approach is broadly used since it can cleanly encapsulate and separate the transient guest software environment from the cloudlet infrastructure's permanent host software and it's less brittle than other approaches like process migration or software virtualization [33].

Regarding implementation, it should be possible to transfer a VM state from users mobile application to cloudlet's infrastructure. The first approach is VM migration in which an already executing VM is suspended and its state of processor, disk and memory will be transferred to destination and execution will be resumed from exact state of suspension in the cloudlet host environment. The second approach is dynamic VM synthesis which mobile device delivers a small VM overlay -instead of the mentioned states in first approach- to cloudlet infrastructure that possesses the VM base. The overlay is calculated by mobile device based on the customized image encapsulating the requirements of the application, see figure 2.2 [19]. Then the overlay is executed in the exact state that it was suspended and the result will be returned by the cloudlet [33].

Cloudlets utilize rapidly deployed VMs which the client can customize freely upon their need to make the VM image or VM overlay which has the application and all necessary requirements to run properly [19]. In both types

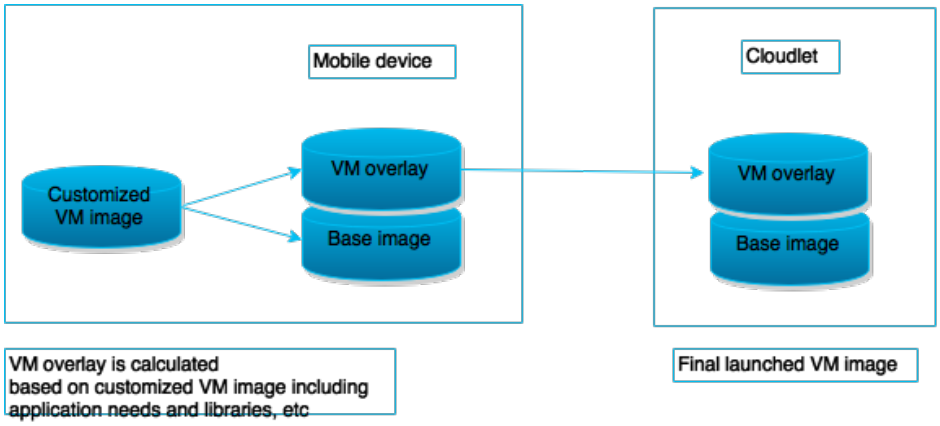


Figure 2.2: Cloudlet dynamic VM synthesis

of implementations the VM image or overlay is created at runtime by user which is quite flexible for offloading the workload to the cloudlet. Nevertheless, despite this flexibility, the procedure of creating an image or a VM overlay and also application status encapsulation could be quite time taking. At the end, it is totally dependant on application design, needs and environment whether to choose using cloudlets as resource rich sources or not.

2.3 Mobile-Edge computing

In Satyanarayanan et al. [33] it is stated that a cloudlet could be any first hop node at the edge of network which resembles a cluster of multi-core computers internally with high bandwidth Internet connection to a distant cloud. Yet after all, it is notable that the business model of cloudlets are not clear. In other words, who's going to play the role of network edge? Is it going to be only dedicated to private clouds which have the possibility of deploying cloudlets in their local network? Is there going to be any standards to increase the simplicity and also to encourage developers to create useful applications?

Realizing the advantages of bringing cloud services and resources closer, the cloudlet concept got the attention of Radio Access Network (RAN) operators, as it is a reasonably valid idea to use RAN base stations as the very first hop of network edge to serve mobile users. Hence, serious efforts have begun to make this idea practical for business environment. As an example, recently Nokia Networks introduced their new generation of intelligent base stations which are considered as edge computing platform enabled base stations (known as:

Nokia Radio Application Cloud Server (RACS)) [3].

Furthermore, ETSI (European Telecommunications Standards Institute) with cooperation of operators such as Huawei, IBM, Intel, Nokia Networks, NTT DOCOMO and Vodafone has formed an Industry Specification Group (ISG) [5] to create a standardized and open environment platform for bringing cloud services closer to the end-users and to formulate a logical integration of mobile applications on such platform between vendors, service providers and third party developers. In other words, the objective is to create an initiative for Mobile-Edge Computing (MEC).

In ETSI's executive briefing [5], MEC is defined as follows: "Mobile-edge Computing offers application developers and content providers cloud-computing capabilities and an IT service environment at the edge of the mobile network. This environment is characterized by ultra-low latency and high bandwidth as well as real-time access to radio network information that can be leveraged by applications."

MEC goal is to transform base stations into high performance customizable intelligent service hubs on the edge of mobile networks while it generates revenue and unique value for operators from offering different value propositions to mobile users, such as proximity of resources, context and location awareness, agility and speed [6].

A MEC-enabled base-station provides developers with ability of running an application on the network edge using a predefined standard platform. This platform might also offer some extra services such as cloud storage, caching, computing, etc to the application. Practically, this turns the base-station into a MEC server. A MEC server can be deployed at different types of LTE (Long-Term Evolution) base stations such as ENodeB or 3G RNC (Radio Network Controller) [6]. Figure 2.3 illustrates the general architecture of MEC.

To conclude, MEC is a new ecosystem which enables MNOs to provide authorized third-parties with a platform to access RAN edge and deploy unique applications based on MEC features. Finally, these advantages enhance quality of experience (QoE) for mobile subscribers and bring value for operators, letting them to play complementary and profitable roles within their respective business models [6].

2.4 Nokia Networks Solution to MEC

To keep pace with the trend of evolution in mobile base-stations, Nokia Networks recently announced their new generation of intelligent base-stations. These base-stations are equipped with Nokia Networks Radio Applications

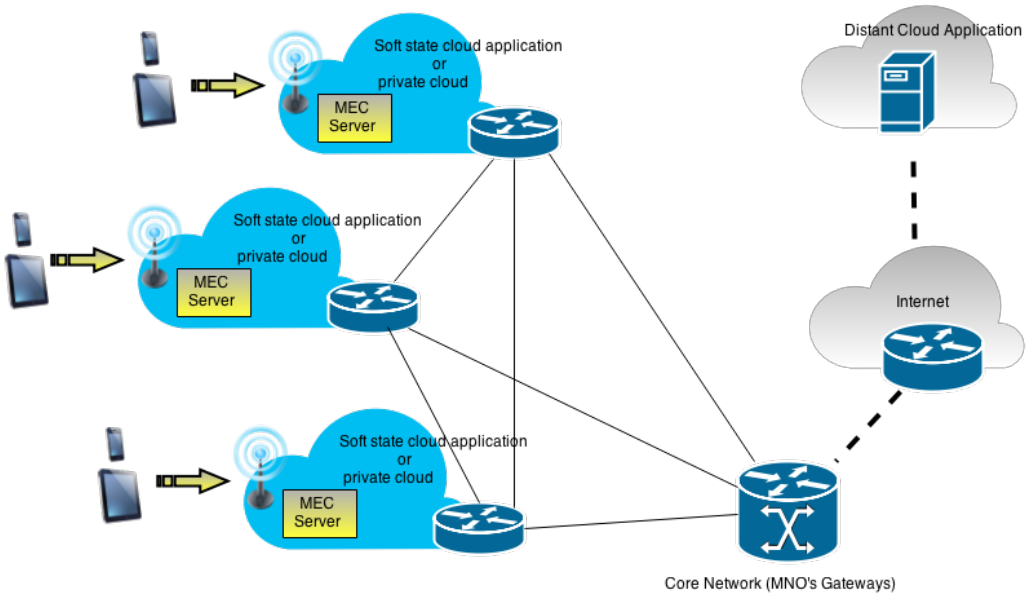


Figure 2.3: Architecture of Mobile-Edge Computing

Cloud Servers (RACS) [3]. RACS has both capabilities of a radio base-station and a MEC cloud server. Cloud applications can be developed on the RACS by third party developers and mobile users can get serviced by these applications while they are connected to one RACS.

On the RACS, applications are encapsulated and run in a virtual machine (VM) on the top of Nokia Networks application engine, see figure 2.4. Each VM is customized according to the requirements the application. When an application is deployed in Nokia Networks network, it means this customized VM is replicated on all the RACS. Each application with certain type of service is encapsulated in a different VM to keep applications from interfering with each others services and also to make the run-time environment more secure. In Nokia Networks terminology these applications are called Liquid Applications.

Regarding developers point of view, Liquid Apps are distributed system applications and all the principles of distributed systems are valid in the environment. Particularly all the distributed system apps can be utilized and deployed on RACS. Nokia Networks has an application life cycle process for the third party developers. This process is called AppFactory. AppFactory consists of verification of the app idea, help and support in the development phase (e.g. test environment and simulation support), validation of developed app, publishing on the Nokia Networks RACS network and at the end,

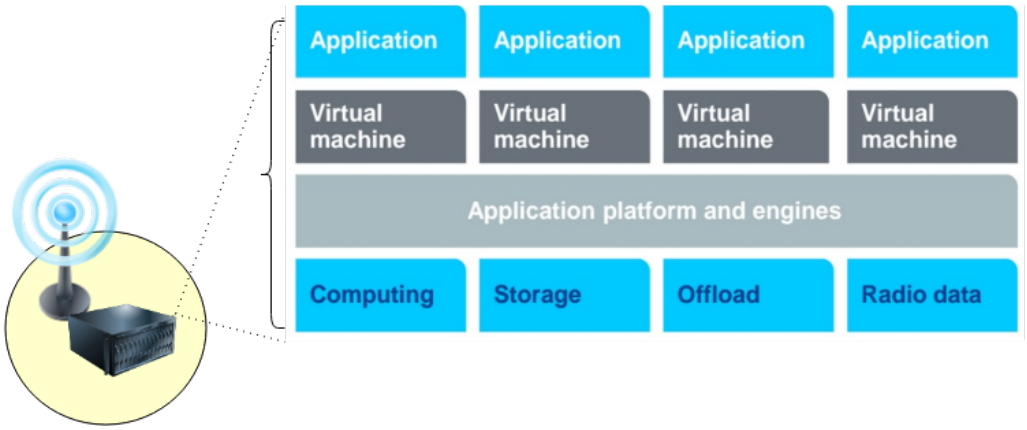


Figure 2.4: Nokia Networks RACS

maintenance. In other words, through the AppFactory process, developers can deploy their Liquid applications on RACS. After verification, development, test and validation phases, their app will be packaged into a customized VM and it will be deployed on all the RACS.

In this master thesis, we use Nokia Networks RACS as the infrastructure to deploy our application on the edge of network. We go through more detail on this in Chapter 5.

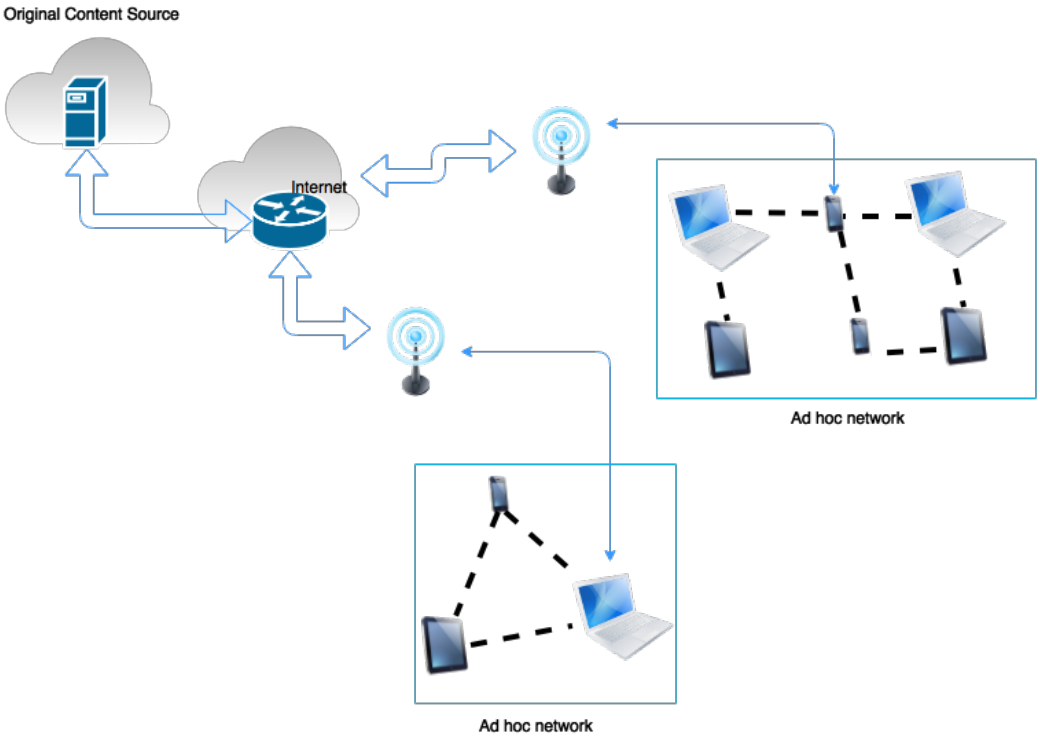
Chapter 3

Cooperative Caching & Content Delivery

The concept of cooperative caching is based on the idea of demanding the necessary data from a neighbour node in the network instead of the original resource. Different approaches have been proposed for cooperative caching, such as, caching on mobile nodes, caching on intermediate or proxy nodes or caching on the edge of network. In this chapter we study different perspectives on cooperative caching in mobile related networks.

3.1 Mobile Cooperative Caching

By technological improvements in smart phones and other mobile devices, mobile clients are capable of sharing data between themselves as peers. In this way they can stay independent from the origin server where the data comes from. Mobile Cooperative Caching is an aggregation of this alternative with the concept of caching for mobile devices [12]. In Mobile Cooperative Caching, mobile devices try to form an ad hoc network with other mobile nodes in the proximity to share the relevant data, see Figure 3.1. To develop this kind of network, one should consider proper policies and select efficient algorithms regarding cache records invalidation, consistency level, cache record placing and searching. Some related studies are reviewed below.



Mobile cooperative caching ad hoc network

There are several desirable outcomes resulting from caching cooperatively in mobile ad hoc networks, such as: improving the access latency, balancing the main server's workload and mitigating the point-to-point channel congestion. However there could be some drawbacks in increasing the communication overhead between mobile devices [11]. Likewise, Content Providers (CP) and Communication Service Providers (CSP) benefit financially from the reduction of bandwidth consumption in network which is achieved by reducing the download request to data source by mobile end-users [35][40]. Some concerns exist regarding implementation and effectiveness of cooperative caching in mobile environments. The system can suffer drastically from user selfishness, non-optimal solution for placing cache records or bad search model [35]. A search model is an approach to find the consistent version of data in the network. Moreover to maintain data consistency, a concrete approach should be used for cache invalidation [8]. Cache invalidation is a process to update values of cached entries across all nodes of network. Yet this process can be different in each case based on system's required consistency level [20][22][31].

There are two main approaches for cache invalidation: push method and pull method. In pulling, mobile device requests the owner of the cached content to verify the validation of the existing cached record. This approach leads to overhead on nodes. In addition, consistency level is dependent on

the other nodes pulling intervals. Nevertheless it's suitable for mobile ad hoc networks since they are not stable networks and any node can leave at any time. In reverse, push method starting point is the original source of the content. In case of any change or invalidation, original source has to notify the cached copy owners. This method is an event based method. In this method consistency level is higher since the event message can propagate fast through the network and nodes are not dependant on each others pulling intervals. On the other hand, the drawback of this method is the overhead on original source for indexing the subscribers. Source has to know all the subscribers to a content, so that it knows where to send the invalidation messages. Push method is suitable for stable networks that node change does not happen often.

Furthermore, cache records placement and searching methods and their performance are another area of study. Taghizadeh et al. in [35] study a cooperative cache record placement policy for cost minimization in mobile network environment with homogeneous content demands. They seek an optimal split between cached objects duplication and uniqueness in each node. A greedy approach for each node is to cache as many distinct records as possible which leads to noncooperation and heavy content duplication. In the other extreme case, a node tries to maximize the total number of unique cached records within the network by avoiding duplications. The study explains that finding an approach in between these two, would result to better network cost for providers. In addition the impact of user selfishness on network cost is simulated and analyzed.

3.2 Cooperative Caching in Content Delivery Networks

The fast growth in sharing of content over the Internet has led to appearance of new network architectures in order to achieve efficient delivery to the end users. Content Delivery Networks (CDNs) are one of the most popular ways to accomplish this goal. CDNs reduce congestion of network and improve the quality of service by caching and replicating contents on intermediate servers placed geographically closer to users at the edge of network [34][17].

CDN is an overlay network which is owned by a service providers (SP). Users request the content from intermediate nodes instead of original data sources. Intermediate servers cache only the most popular contents due to limited storage. In a cooperative CDN network, intermediate nodes cooperate with each other to reduce the transferred traffic and consumed

bandwidth of the network. When an intermediate node receives a request for a particular content, it searches its own cached records and sends back the content if it exists in the records, otherwise the content is fetched from the other intermediate nodes in the network or else from the main source. This approach also conducts cost efficiency for the SPs [37].

There are several studies trying to address cooperative edge caching challenges in CDNs. For example it's necessary to have a proper approach to come up with a decision for the number of intermediate nodes to get closer to the edge network and to the end users, and also the geographical locations where these middle servers need to be placed. In addition an approach is required to organize these nodes in effective cooperative groups. Also finding an adaptive architecture is an issue to handle dynamic contents [28]. Ramaswamy et al. in [29] introduce the concept of Cache Clouds as a framework for cooperation in large-scale edge cache networks. The architectural design of the Cache Clouds includes dynamic hashing-based content look-up and update protocols. The architecture enables Cache Cloud to dynamically balance look-up and update message requests between the caches in the network. In Cache Clouds middle nodes are formed into certain groups. Each one of these groups has a beacon node. A beacon node is a node in which the original cache record of a dynamic content exists. This beacon node works as a reference to this specific content. The rest of the nodes in need of this particular record in the group cooperate with the document beacon point for invalidation updates. Using beacon points helps to alleviate load balancing inside the group and to be failure resilient. Beacon nodes of different clusters can form a group to work with each other as peers and facilitate the cached document transfer and also acknowledge each other about cache invalidation. For example in Figure 3.1 node C can be beacon node for nodes A and B for a specific content, while these nodes themselves are beacon nodes of this content in their clusters.

Considering multimedia contents, they are classified into on-demand multimedia contents like stored audio and video, and live multimedia contents like live radio and television streams. Local CDNs use cooperative caching to serve future possible users. The main problem in live multimedia streams is that it is not possible to cache them in advance but one can use the fact that these live streams are usually synchronized. Hence a CDN server can aggregate user requests aiming for a same live stream and form a multicast group. The multicast group nodes create an multicast overlay network to deliver the content from the origin server to all members. This is known as application-level multicast (ALM) or overlay multicast [27].

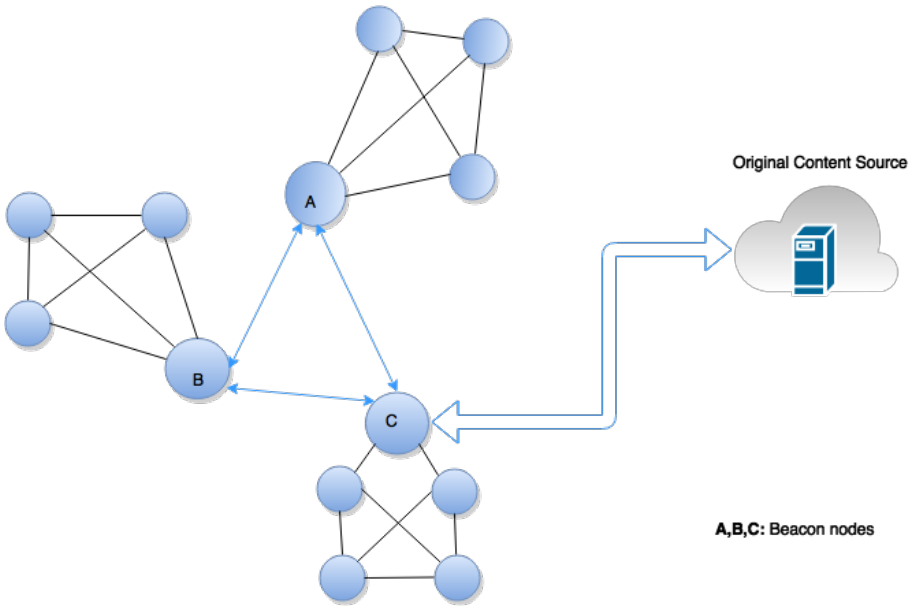


Figure 3.1: Beacon nodes group

3.3 Cloud-P2P Content Distribution

Huge amount of Internet traffic is devoted to content distribution and it grows each day. By significant rise in cloud technologies big content providers try to utilize cloud-based CDN approaches so that they can efficiently satisfy the users need for performance and reliability. In addition, employing cloud infrastructure let providers scale their service as fast as possible. As an example, Netflix employs Amazon's cloud services and third-party CDNs such as Akamai and Limelight [13][21]. There's also peer-to-peer (P2P) approaches for content distribution. Rather than advantages in scalability and end-to-end speed, a large-scale P2P system based on resources of individual users, can reduce the load which needs to be served by data centers. Since there are studies like [36] to optimize resource consumption in such architectures.

Despite all the benefits of P2P content distribution, there are also some drawbacks such as end users dynamic nature and difficulty to find peers with the desirable content. In result, "Cloud-P2P"approach has came to light as a promising alternative [23][41]. Cloud-P2P tries to resolve the issues of previous approaches. End-users find content seeds on the cloud and also they get assisted by different variant methods in finding nearby peers which are usually considered in the same peer groups based on geographical location or bandwidth optimization considerations. In this way Cloud-P2P provoke lower

infrastructure and network bandwidth costs compare to pure Cloud CDN or P2P approaches [21]. For example in [42] Zhao et al. study optimization of inter-ISP traffic in Cloud-P2P CDNs by adding locality-awareness property to peer groups.

Chapter 4

Mobile-Edge Computing Cooperative Caching System

This chapter describes the overall design of the Mobile-Edge Computing Cooperative Caching system. It describes different parts of the system, its goals and the ways to achieve them.

4.1 Overview

Meeting the demanding quality-of-service (QoS) requirements of emerging real-time, interactive and media services has led to edge computing paradigms, where the focus is on delivering high QoS, at low operational costs. In addition the fast growth of mobile users utilizing these services poses significant traffic growth in Mobile Network Operators (MNO) network, which needs to be addressed according to cost efficiency, limited backhaul bandwidth capacity and mobile applications desirable response times. Employing cooperative caching in the edge of network enhances data locality while helping to balance MNO's network workload. In this way, by decoupling the time which a content is downloaded from a distant cloud, from the time which the content is delivered to the mobile user, an edge network cooperative caching solution can boost the QoS experience and reduce the latency while saving the MNO's network backhaul resources. Mobile-Edge Computing Cooperative Caching refers to the use of cooperative caching on MNO's intelligent base-stations which are equipped with MEC server capabilities, to store content at the very first hop of the network from end-users. For instance, if a mobile user request a popular cached content, he will receive the content directly from the relevant base-station rather than relevant CDN server or original distant cloud server- until the session or cache record consistency expires.

Each Mobile Network Operator(MNO) utilizes few gateways to conduct the mobile user traffic out of their inter-network toward Internet backbone. This infrastructure architecture combined with increasing number of mobile users leads to network bandwidth and resource challenges [15][39]. On the other hand, large amount of traffic consumption is according to request popular contents on the web. As an example in Youtube, 80% of views are dedicated only to 10% of popular contents [9]. As matter of fact in this way cellular networks suffer from sending the same content multiple times while it's not necessary [38]. Equally, this congestion causes poor Qos and user experience. Also, as it is mentioned by Chen et al. in [10], "Regardless of MNOs efforts to improve the bandwidth of wireless links by employing approaches at both the physical (PHY) layer and medium access control (MAC) layers in Long Term Evolution (LTE) and LTE-Advanced systems, such as massive multiple-input multiple-output (MIMO), carrier aggregation, and coordinated multipoint (CoMP) transmission, the utilization efficiency of the radio spectrum is notably reaching its theoretical cap".

Rising in the number of mobile users has led to pervasive deployments of high-speed radio frequency base-stations which consecutively resulted in gradual appearance of Mobile-Edge Computing to tackle previously mentioned challenges. MEC cooperative caching helps MNOs to balance the load in the network and reduce the redundant content requests. Likewise, it helps to lower MNO's inter and intra-network traffic [39][10]. To conclude, Utilizing MEC cooperative caching can accelerate cost effectiveness, better user experience and quality of service.

4.2 MEC Co-Caching System Design

Mobile edge computing cooperative caching (MEC co-caching) system is a distributed caching system deployed on a cluster of MEC base-stations. All the base-stations on this cluster run their own MEC servers in order to host their share of the system. The idea is that any arbitrary content delivery application can connect to this system with an application API and benefit from the caching. Some aspects should be considered in such system in order to achieve the desirable objectives. We review each of these aspects regarding the following objectives: low network latency, rich user experience, cache consistency, scalability and less inter and intra-network bandwidth usage.

The overall architecture of the MEC caching system is shown in Figure 2.3. We recognize two different entities in our system. First one is the global cache system which is formed cooperatively on a cluster of MEC base-stations. The second one is the local cache entity in each one of the base-stations. In global

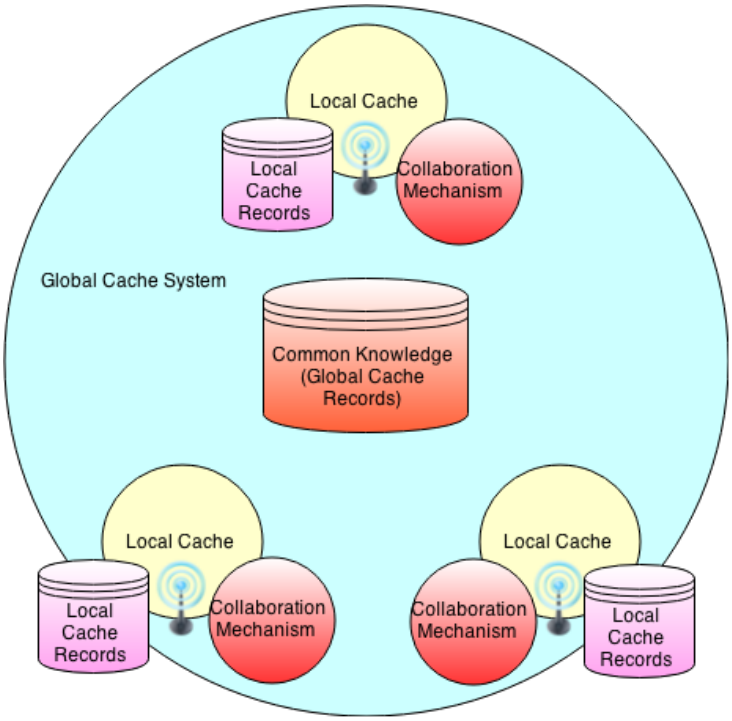


Figure 4.1: Architecture of Mobile-Edge Computing Cooperative Caching system

cache system there's a repository of common knowledge which is practically the index of all the cache records existing in the system. Each base-station in the cluster keeps track of all the records in the cluster through this repository. Also in each base-station itself, there's a local cache records repository that practically is a hash table of all the data content which is cached directly by the base-station or requested from a beacon node. A beacon node in the system is a base-station who has the very first version of cached content from the data source.

In addition to a local cache repository in each base-station, there is a collaboration mechanism module to synchronize all the local caches with the global caching system. There's also a consistency module that takes care of cached records validation and consistency. Figure 4.2 illustrates these different modules of the MEC Caching System that reside in the MEC Server of each of the base-stations in the cluster.

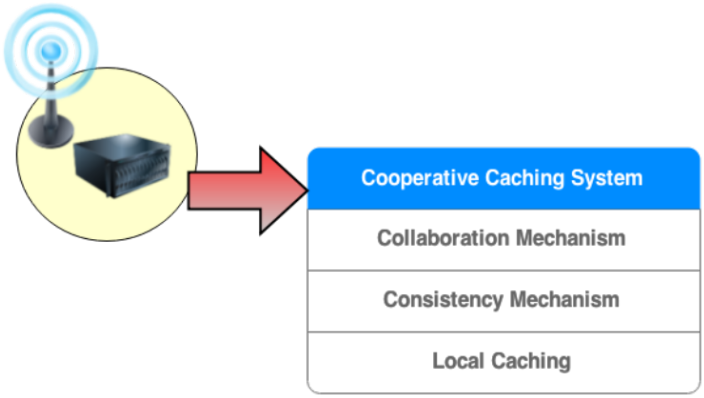


Figure 4.2: MEC Cooperative Caching System parts on each MEC server

4.2.1 Low Latency and Rich User Experience

MEC co-caching is a distributed system that brings the content as close as possible to the mobile user by using the MEC approach. This means that the system is deployed on the edge of the network on MEC base-stations with high connectivity to the Internet. Mobile user gets the cached content from MEC caching system through the connected base-station. This base-station is the first network hop that user is connected to. This helps to reduce the delay of the network for fetching the content and helps to improve the user experience.

4.2.2 Consistency

There are two general approaches to handle a distributed caching system consistency, pull and push methods. In pulling approach, a node with a cached content takes care of validity and consistency of the cached record. It knows who is the beacon node in the higher hierarchy of the cached content and in certain periods checks if still the version of the cached content is valid or it is changed. In this way if the original content is changed the node can have the content from the beacon node or it can request itself from the original source. On the other hand the push approach works with raising some events from the side of original cloud source towards the owners of cached copies to inform them about latest changes. In this way beacon nodes in the content delivery network are responsible to inform the subscriber nodes to the content.

In the MEC cooperative caching we use push approach. This means if any alterations happens to a cached content the meta data inside the common

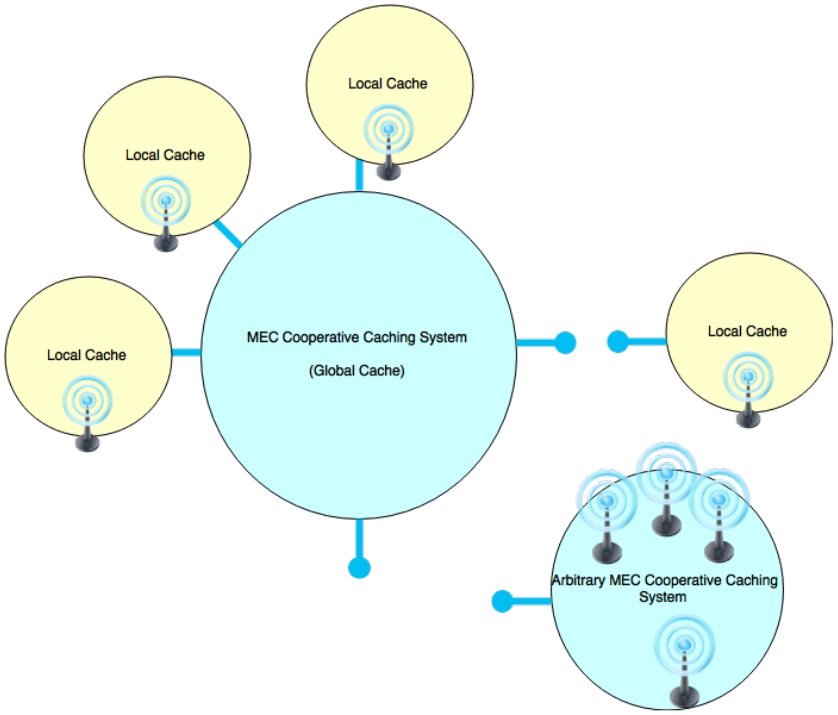


Figure 4.3: MEC Cooperative Caching scalability

knowledge repository will be changed and automatically all the base-stations will be informed about this change and they will update their cached copies or will take a proper action.

4.2.3 Scalability

MEC cooperative caching system is designed to be as scalable as possible. As it is presented in Figure 4.3 this scalability means that it should be possible to add or remove new base-stations to the running cluster. Also it should be possible to merge two different MEC clusters to make a bigger one. Collaboration mechanism module is responsible to provide this flexibility by allowing new base-stations join the global cache system and have access to the common knowledge repository. The other base-stations will be informed of the new base-station arrival. The scenario is the same for merging two clusters. In the implementation chapter more information is provided.

4.2.4 Less Inter and Intra-network Bandwidth Usage

When mobile users are connected to MNO's network through base-stations, application data has to traverse the MNO's network back to the MNO's gateway to reach the WAN. The gateway is connected to Internet backhaul. The bandwidth usage of Internet backhaul is one of the main costs of MNO's data services, so if we reduce this cost, we help MNO's to offer cheaper services to end-users. To achieve less inter and intra-network bandwidth usage, MEC cooperative caching system keeps the caching traffic inside the global cache cluster by using beacon nodes. These beacon nodes are the base-stations that got the cached content for the first time from the original cloud source. Later on if any of the MEC nodes needs to send the same content to a user, they will request it from the beacon node instead of requesting it from the source or nearby CDN server.

Figure 4.4 presents the use case that MEC Co-Caching system tries to avoid to keep the network terrific inside the global cache cluster. In this scenario mobile user is connected to a base-station and requests object X. The base-station forwards the request to the distant cloud that contains the original content while there's a valid cached copy of object X in a nearby base-station (a beacon node). Respectively, Figure 4.5 depicts the scenario that happens inside the MEC Co-Caching system. When MEC node receives the request for object X, first it checks the common knowledge repository in global cache to see if there is any beacon node that previously requested the same content. If it's true, base-station fetches the meta-data (containing the address) of the specific beacon node to connect and to request the beacon node to send him back the cached copy of object X. This copy will be sent back to user by the base-station after receiving from beacon node.

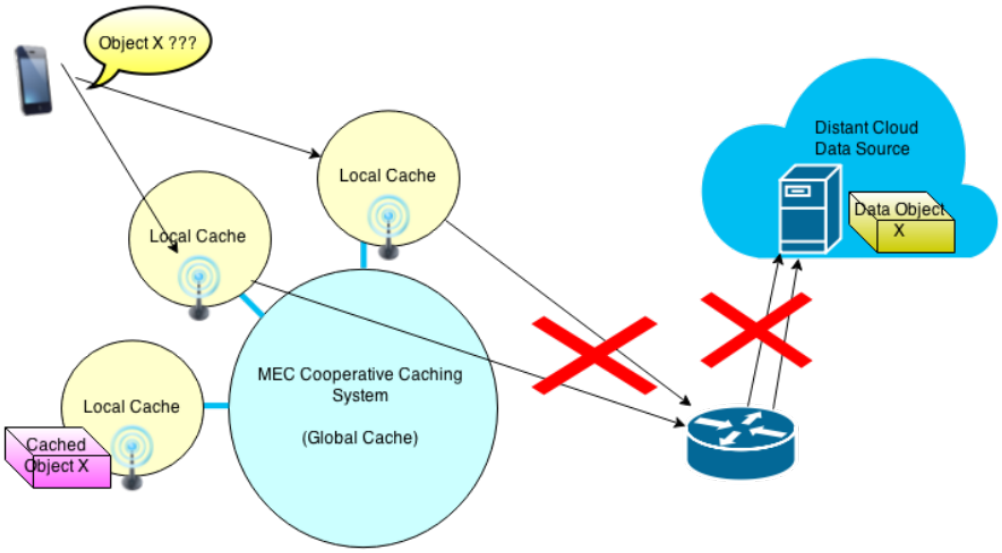


Figure 4.4: Intra-network bandwidth consumption

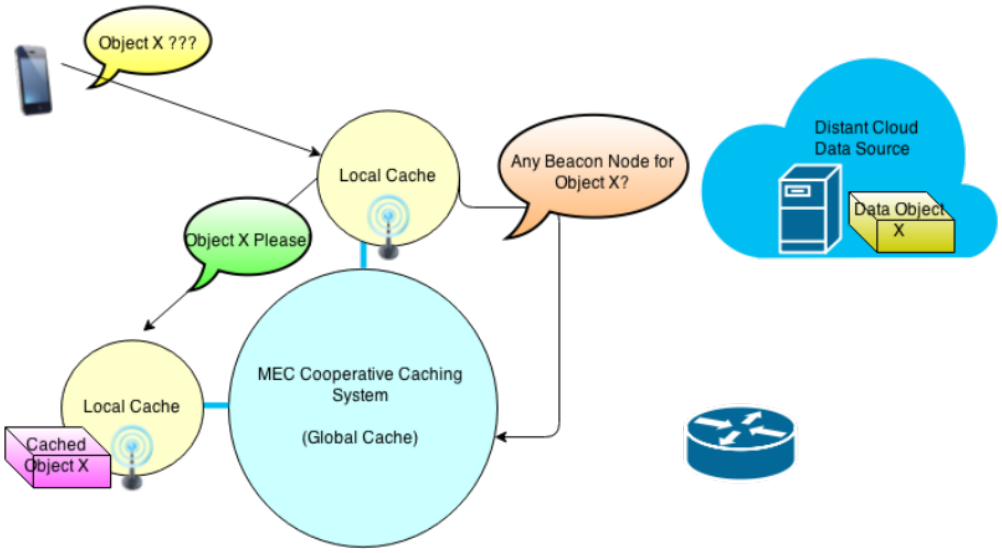


Figure 4.5: Inter and intra-network bandwidth usage saving use case

Chapter 5

Implementation & Deployment

This chapter overviews different implementation aspects of MEC Cooperative Caching System and its deployment considerations. We present the implemented components and utilized technologies alongside their relevance to the system. Also the circumstances of the deployment are reported.

In the system implementation, light weight collaboration and consistency mechanisms exploit Apache ZooKeeper Service. This service is also utilized regarding creation of the cluster of base-stations. Hence, in the first part of this chapter we introduce the ZooKeeper and its functionalities as a light weight and efficient distributed systems coordinator service, then we continue to present the details of the modules in the system as well as local cache and global cache system modules which are implemented using Java.

5.1 Apache ZooKeeper Service

Apache ZooKeeper is an open source distributed coordination service designed to facilitate distributed application development by enabling developers to concentrate on the logic of their program rather than distributed nodes coordination and management tasks. ZooKeeper also takes care of certain levels of failure and recovery in the system, with the help of Zookeeper it's possible to design a highly fault-tolerant application and to avoid the single point of failure in the system. The most common use cases of Zookeeper are service discovery, dynamic configuration management and distributed locking [18].

Zookeeper provides a simple API that enables developers to implement common distributed system coordination tasks. The APIs are available in Java and C and the service component itself is implemented in Java that runs on an ensemble of servers.

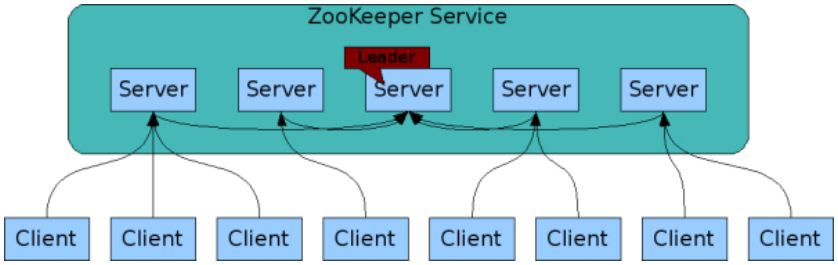


Figure 5.1: Zookeeper service clients and servers [4]

In distributed system programming, separating the application data from control or coordination data is a best practice. This best practice is easily achievable by using ZooKeeper. In MEC Cooperative Caching system, employing ZooKeeper helps to make the coordination task and failure and recovery specific details transparent from the original content source.

According to Apache ZooKeeper technical overview [4], ZooKeeper allows multiple distributed processes to cooperate with each other and coordinate themselves through a shared hierarchical name space. From abstract point of view, this name space is similar to a distributed file system which consists of in-memory data registers called Znodes. Utilizing in-memory approach instead of saving on permanent storage lets ZooKeeper to achieve high throughput and low latency.

ZooKeeper works on an ensemble of servers in which each one of these servers has a replication of ZooKeeper server installed. All the servers in the ensemble are informed about each other. They utilize an in-memory image of state, transaction logs and system status snapshots in a persistent storage. In ZooKeeper terminology, a quorum is the minimum number of servers in an ZooKeeper ensemble that should be up and running so the availability can be guaranteed [18] [4].

As it is depicted in Figure 5.1 clients can connect to a single ZooKeeper server in a time. They establish a TCP connection to the server and send requests, get responses, get watch events and send heart beats. If a failure occurs or server loses the TCP connection for any reason, the client can connect to any other server in the ZooKeeper ensemble [4].

In following, the ZooKeeper properties and concepts are introduced in regard to MEC co-caching system.

5.1.1 ZooKeeper Ensemble

There are two modes of running for ZooKeeper servers: standalone and quorum. In standalone mode, ZooKeeper service runs on a single server and ZooKeeper state is not replicated. In reverse, in quorum mode, there's a set of ZooKeeper servers that state of each server and Zookeeper common data tree (Znodes) are replicated between the participants.

A ZooKeeper in quorum mode guarantees the followings as it is stated in the Apache documentation [4]:

Sequential Consistency "Updates from a client will be applied in the order that they were sent."

Atomicity "Updates either succeed or fail. No partial results."

Single System Image "A client will see the same view of the service regardless of the server that it connects to."

Reliability "Once an update has been applied, it will persist from that time forward until a client overwrites the update."

Timeliness "The clients view of the system is guaranteed to be up-to-date within a certain time bound."

In a ZooKeeper ensemble, a quorum is the minimum number of servers that have to be running and available so that ZooKeeper service works properly and is able to guarantee the previously mentioned factors. This number is also the minimum number of servers that have to store a client's data before client can continue. As an example, for a ZooKeeper ensemble of 5 servers with quorum number 3, as long as all three servers have stored the data, the client can proceed to next task, and the other two servers will eventually catch up and store the data [18].

5.1.2 Znodes

Znodes are the common data structure between the servers participating in a ZooKeeper ensemble. Znodes characteristics provided by ZooKeeper resemble to a distributed file system. There's a path in ZooKeeper's name space hierarchy which identifies a certain Znode. Different branches in the path are separated by "/". Each Znode can have a data associated to it.

The main purpose of ZooKeeper design was to store a unique view of coordination data between distributed servers. For example status and meta data information or configuration parameters. Therefore the stored data at

Figure 5.2: Znodes hierarchy

each Znode should be relatively small, between bytes to kilobytes. Znode takes advantage of a stat structure which is similar to Lamport vector timestamps and it includes version numbers to keep track of each change in the ensemble. Version number changes by any change in data, ACL or any coordinated updates. After each change, the version number increases in the relevant node. Version numbers are sent as meta data to a requested content of a Znode. [4].

Znodes can be persistent, ephemeral and sequential. The persistent Znodes are persisted in the ZooKeeper ensemble even though the session that created them is lost. In reverse, ephemeral nodes will be deleted if the session is not valid anymore. In sequential mode, Znodes are persistent, the only difference is the added automatic sequential number to Znode names at the moment of creation. Through ZooKeeper's API, it's possible to create, update and delete a node's name and data[4]. In the Figure 5.2 a sample Znodes hierarchy is presented. Any of these nodes can be ephemeral or persistent.

5.1.3 ZooKeeper Watches

ZooKeeper watches are events that can be set on Znodes. When a watch is set, with any alteration in a Znode the watch will be triggered and removed. This means if it's needed to keep an eye constantly on a certain Znode, a watch should be set again after each trigger. When a watch is triggered a message will be send to inform the client. This events also will be triggered if a connection establishes or breaks. If a client wants to constantly monitor changes on a Znode, it has to set a watch again after receiving the notification. Several clients may submit watches to a single or several Znodes [4].

5.2 Implementation

MEC Cooperative Caching implementation is done using Java language. The main part of the program consists of a manager application running on the MEC server of each Base-station listening to predefined ports for mobile client connections. This application is practically the coordinator between local cache and global cache. In addition in all of the base-stations a ZooKeeper server is running and each base-station is part of the ZooKeeper ensemble.

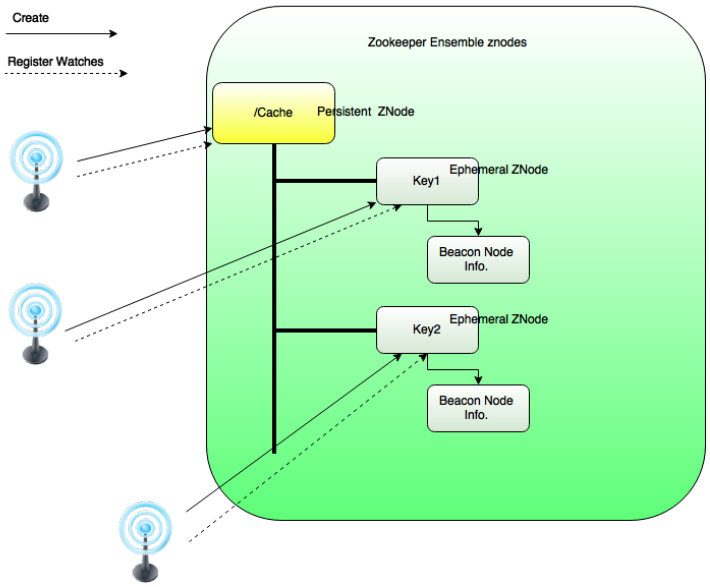


Figure 5.3: MEC Co-Caching implementation

5.2.1 Local & Global Cache

Local cache is implemented using Java array lists for keeping the cached content identifiers. Each identifier matches with a content on the local file system. If a content exists in local cache it means that it has been requested before by at least one mobile client connected to this base-station.

Global cache keeps the cached content identifiers and their related beacon nodes for all the base-stations participating in the Zookeeper service ensemble. Global cache uses ZooKeeper Znodes to implement the cache records structure which is the common knowledge repository. Each global cache record matches a unique content on one beacon node. As it is presented in Figure 5.3 the records structure consists of a persistent Znode with the name "/root"and all the other records will be added under root as ephemeral Znodes. The creator of each ephemeral record is a beacon node that is requested a content for a first time. If this beacon node fails, the record will be removed automatically by ZooKeeper because it was ephemeral. It is possible to consider more than one beacon node for each cache content identifier but for simplicity our assumption is based on having only one beacon node for each record.

5.2.2 Collaboration & Consistency Mechanisms

Collaboration and consistency mechanisms are implemented using Znodes and ZooKeeper watches. Every base-station which detects it is the first node in the ensemble who requests an specific content, register itself as the beacon node for that content in the common knowledge repository. That's how everybody else can be aware of not requesting the same content again. Instead they request the content through a direct TCP connection to the beacon node. This is how the collaboration mechanism works.

Likewise, base-stations coordinate with each other by constantly registering watches on relevant Znodes in common knowledge repository. If a new record is created by a beacon node, a watch will be set on that record constantly till that record will be deleted or expired. This is also the case for the node that request a content directly from a beacon node. In this way, if a cache record changes in any possible way or some base-station or the main source alter the data every other base-stations participating in the ZooKeeper ensemble will be notified. Regarding the cache invalidation, if an original content changes, the only thing that the main cloud source application needs to do, is to create a handle to the relevant ZooKeeper ensemble and alter the record identifier related to its content. In this way any of the concerned base-stations will be notified because they all have a previously registered watch on that specific content cache record. In Figure 5.3 setting watches on different cache records is depicted. In other words, this is practically the implementation of cache invalidation push method (consistency mechanism) which we mentioned in Section 3.1.

5.2.3 Mobile Client Content Request Use Case

As it is illustrated in Figure 5.4 in this part we review the MEC Co-Caching system process flow after simple content request from a mobile client which is connected to one of the base-stations in the ZooKeeper ensemble. This process diagram contains all the main processes implemented in the system.

A mobile application is implemented to connect to MEC CO-Caching system and to request a content. This application sends the content identifier as request for the content to the main Java program -we call this main server for simplicity- running on the MEC server of the base-station through TCP. Main server checks its local cache for the content, if it's a hit mobile user will receive back a cached copy of the content, if not, main server uses Java ZooKeeper API to create a handle to the ensemble and checks if the content identifier exists in Znode structure. If yes, this means there's a beacon node nearby that already has requested the same content, so it has a cached copy.

Then main server reads the meta data of this beacon node from the relevant Znode and sends a request to receive the cached copy directly from beacon node. After receiving the cached copy, main server sends the content back to mobile user, in addition it registers a watch on this specific Znode so that in the future gets notified of any manipulation or cache invalidation. Also it adds the content to its local cache records.

On the contrary, if there's no beacon node it means the base-station itself is the first one requesting this content in the ZooKeeper ensemble. So it requests the content from the original source directly and after receiving the content and sending it back to the user, main server adds the base-station information as the beacon node for the content to ZooKeeper Znodes. Also a copy of the content will be added to the local cache. At the end, it's obvious that if the base-station has a cached copy of the content in its local cache, it will be immediately sent back to the client.

5.3 Deployment

In this Thesis, the implemented MEC Co-Caching system is deployed on Nokia Networks Radio Application Cloud Servers (RACS). RACS is equivalent to MEC base-station environment. As it is presented in Figure 2.4 there's a server for each base-station which is responsible for all MEC functionalities. These RACS are situated in the campus of Aalto University in Otaniemi. The network is called NetLeap Network and it's a collaboration between Nokia Networks and Aalto University to boost innovation, creation and verification on telecommunication-cloud related applications and services.

For each base-station, MEC Co-Caching main server and ZooKeeper service applications are packaged into an specific virtual machine (VM). Then the VM runs on top of RACS application platform. This VM is replicated on the other RACS, although there might be obligation to change some configurations. We practically use two different RACS instances for the evaluation purpose. One is used as our main node and the other one to test the functionality of a beacon node. Also there's a mobile device which uses a special SIM card to connect to the main RACS. The client side application is installed on the mobile device and sends content request through a TCP connection to the main server application on the main RACS.

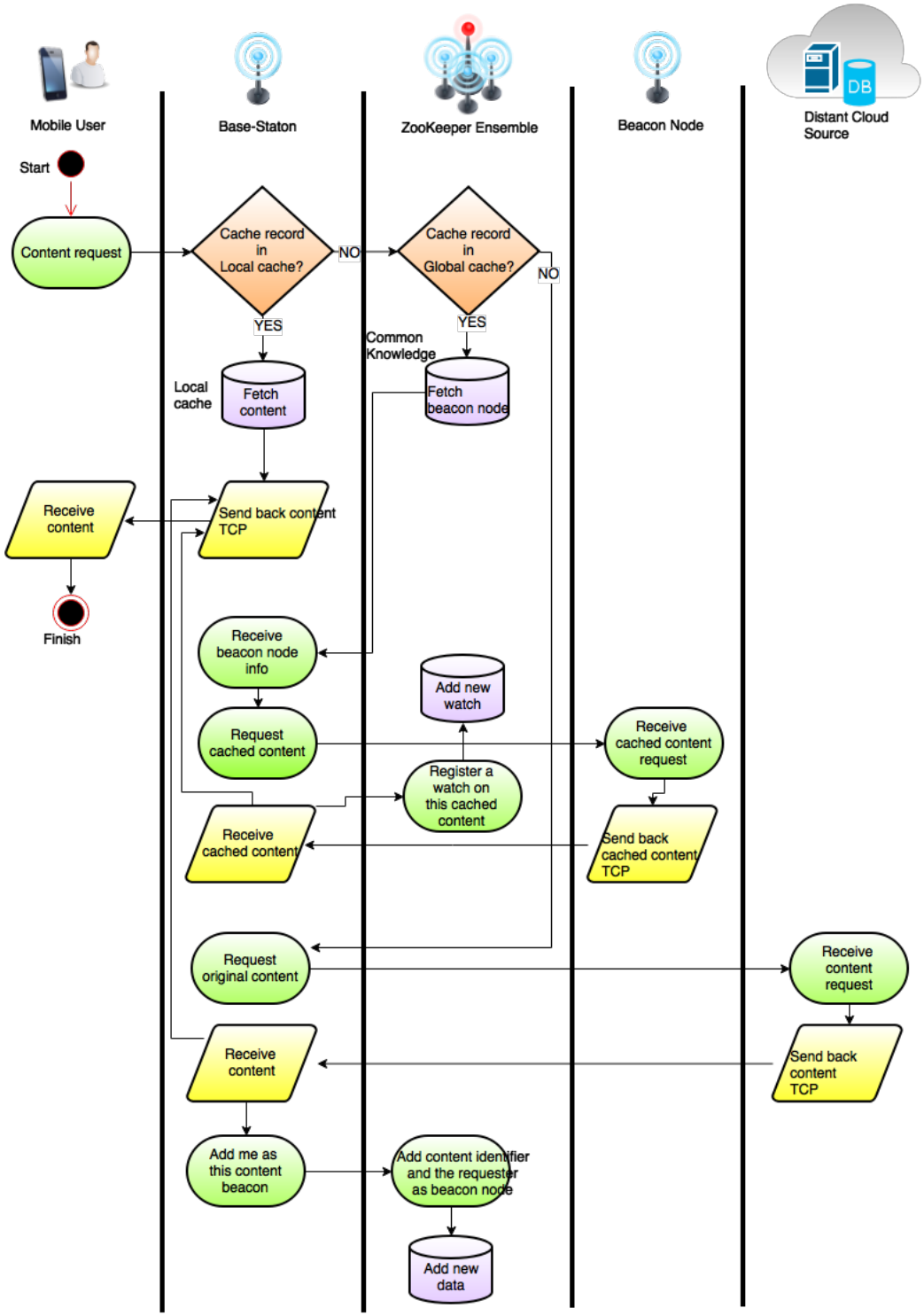


Figure 5.4: Content request process diagram

Chapter 6

Evaluation

This chapter analyzes the results of using MEC Co-Caching system and the influence on network delay and user experience.

6.1 Test Environment

The test environment consists of two RACS instances, one AWS server and one smart phone. The RACS are configured as a Zookeeper ensemble and they also run the MEC Co-Caching system main server program. The smartphone runs an app to request desired content by sending the content ID through TCP to main server on RACS. The content benchmark resides in AWS server and an HTTP application is run to serve the content to RACS. The properties of test environment are listed in Table 6.1 and the benchmark details in Table 6.2.

Environment Object	**Network/Location**	**Connection Type**	**Role**
Nokia Networks RACS	NetLeap Network, Otaniemi, Helsinki	TCP, HTTP	MEC base-Station
Smart-Phone	NetLeap Network, Otaniemi, Helsinki	TCP, HTTP	client
AWS Server	Frankfurt	TCP, HTTP	content source distant cloud

Table 6.1: Test Environment

Content ID	Type	Size (Byte)
1	jpg	175730
2	jpg	2916838
3	mp3	9997627
4	avi	25840890
5	flv	52804376
6	flv	110917557
7	flv	165007211
8	ova	881933824

Table 6.2: Benchmark Details

6.2 Test Scenarios

To test the MEC Co-Caching system, three internal scenarios are considered and all these three cases have been tested on the benchmark. These three caching scenarios and their details are presented in Table 6.3. The objective is to evaluate MEC Co-Caching system network latency, bandwidth saving and user experience improvements.

6.3 Latency Time

In this section we present the main response delay test results of the defined scenarios performing on MEC Co-Caching system. All the presented results are the average outcome of 10 different iterations in each test scenario. Also the main source of content in all the scenarios is the AWS server which is introduced before. All presented delays are end-to-end delays, from the exact moment of sending a request from a relevant base-station/smartphone till the exact moment of receiving back the complete response of the request which it means till the end of complete transfer of the content to the requesting client.

In Figure 6.1 results from scenario 1 and 2 are compared to direct request of smartphone to AWS server through Netleap LTE network. Delay reduction in both scenarios are easy to observe and the difference drastically increase by the increase in size of the contents. In average in the benchmark in scenario 1 the total delay is improved by 87.38% and in scenario 2 by 80.16%. Though it should be mentioned that the relatively good result is dependent on the increasing size of contents in the benchmark. For example for the smallest content (index 1) the rate of improvement is less than 9% in average in both scenarios and on the other hand for the largest content (index 8) the rate of improvement is more than 88%. The detailed values of Figure 6.1 are

	Scenario	Details
1	Request exists in local cache	Mobile device requests a content from the RACS(1) that it is connected to. A copy of content is already cached on the RACS.
2	Request exists in global cache and not in the local cache	Mobile device requests a content from the RACS(1) that it is connected to. Local cache doesn't have a copy but some RACS(2) in the ensemble has requested before the same content. A cached copy exists in global cache. RACS(1) requests a copy of the content from RACS(2) and sends it back to client.
3	Request is not cached at all	Mobile device requests a content from the RACS(1) that it is connected to. Local cache doesn't have a copy neither any other RACS in the ensemble has requested before the same content. RACS(1) requests the content from the original source and sends it back to client.

Table 6.3: Test Scenarios

illustrated in Table 6.4.

On the other hand it can be seen from the presented results in Figure 6.2 that in case of scenario 3, the routine scenario of requesting by a smartphone through LTE network has the better total delay in all the benchmark contents while it's expected for them to have delays almost at the same range. We investigated this problem and found this is the effect of using a proxy server for transferring the requests from RACS to Internet. Nevertheless this slows down the MEC-Co Caching system but in general doesn't influence the good results of two other scenarios. Exclusively that the value proposition of the system is based on the presumption of reducing the negative outcomes of excessive user requests for the same popular contents.

In Figure 6.3, four main phases of the MEC CO-Caching system are evaluated. The presented figures are average percentage of relevant terms in all scenarios. It can be derived from the figure that the main time taking phase in all scenarios is request from RACS to AWS server which was expected as we mentioned the phenomenon before. It is also interesting to see that requesting from RACS to RACS, smartphone to RACS and the overhead of processing time delays are in the same range. Processing time is the spent time on handling the distributed environment between the local cache, global cache and ZooKeeper service-see Chapter 4&5-. Hence it can be concluded that the processing time wastes a relatively small time of the system.

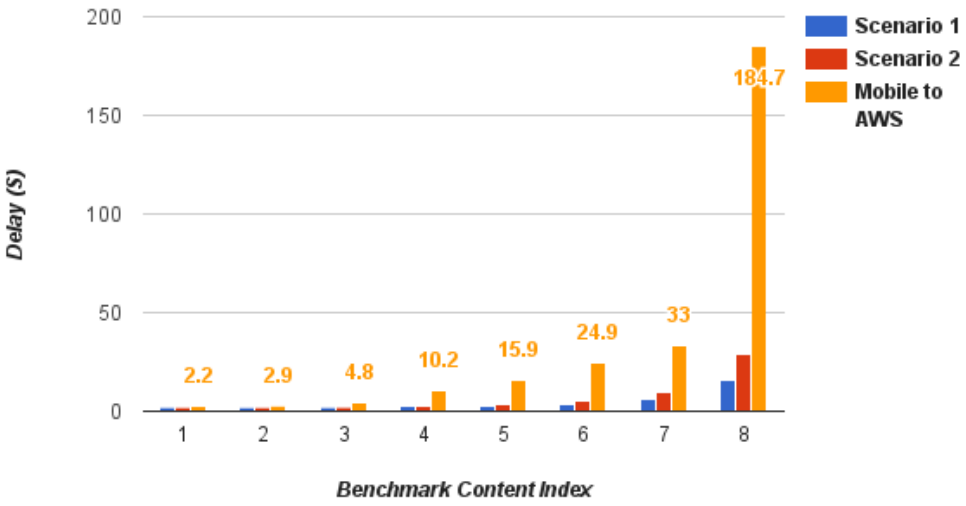


Figure 6.1: Total delay

Content ID	Scenario 1	Scenario 2	Mobile to AWS
1	2.002	2.006	2.2
2	1.94	1.98	2.9
3	2	2.1	4.8
4	2.21	2.51	10.2
5	2.4	3.1	15.9
6	3.27	4.87	24.9
7	5.9	9.6	33
8	15.5	29.1	184.7

Table 6.4: Total delay table (s)

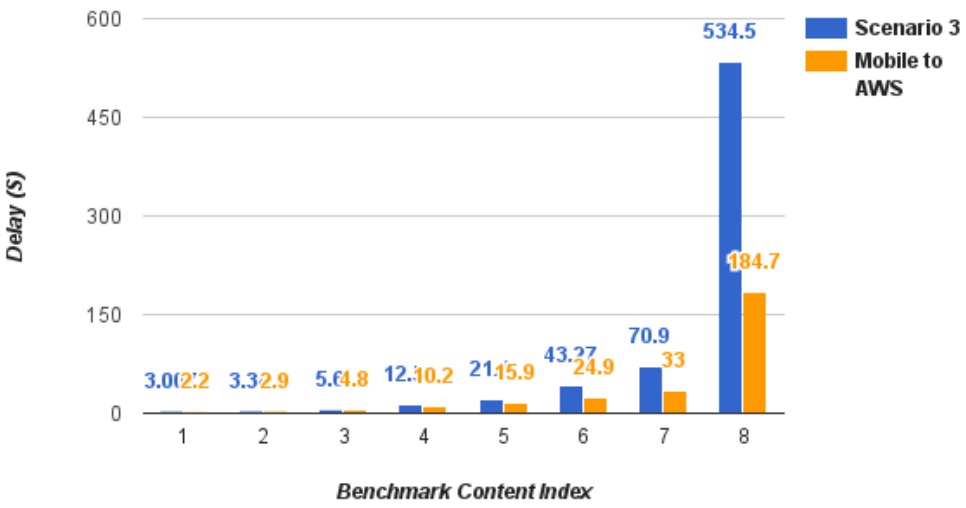


Figure 6.2: Total delay

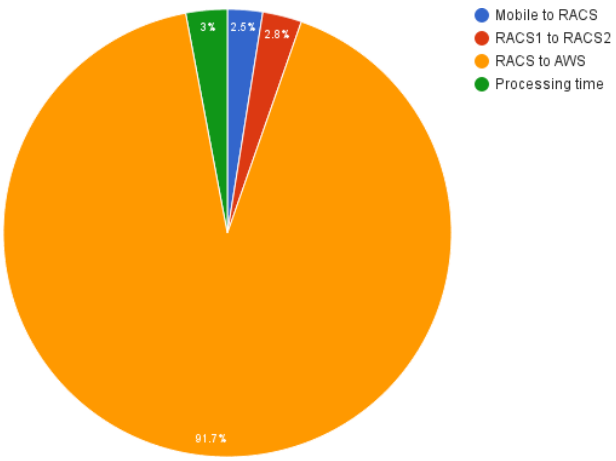


Figure 6.3: Specific delay

Scenario	Inter-network	Intra-network
1	HL*XM	HL*XM
2	-	HG*XM
3	-	-

Table 6.5: Average bandwidth saving

6.4 Bandwidth Saving

If a content X is cached in the system - scenario 1 and 2 - no data will be transferred through the main gateway of Nokia Networks (MNO's gateway) toward the operator's intra-network. In second scenario the amount of inter-network transferred data is equal to size of the content. On the other hand in third scenario content X should go through the Nokia Networks gateway. Is this going to be useful or not for MNOs is dependent on the MEC Co-Caching system hit rate. If we consider HL and HG as number of local and global cache hits consecutively, and XM as the average size of content in our benchmark, in each one of our scenarios the total amount of saved bandwidth can be calculated as they are presented in the Table 6.5 .

As it is mentioned in previous chapters, most of the requests of contents on the web are dedicated to popular contents and this is when caching becomes beneficial. Likewise in MEC Co-Caching system, as higher the popularity of a content (or benchmark), the amount of saved bandwidth is higher too. In other words, when mobile users request popular contents, the probability of hit rate is higher hence the amount of saved bandwidth is bigger. In the Figure 6.4 the amount of total saved bandwidth -inter and intra network- for the test benchmark is presented for each cache hit that occurs in scenario 1 and 2. There's no saving in scenario 3 since all data goes through the gateway after traversing the inter-network. It can be concluded that for small files bandwidth saving is not enough beneficial. So the total amount of saved bandwidth is worthy when the system has high amount of requests for large popular contents.

To elaborate more the bandwidth gain from utilizing the system, we presume a set of 100000 requests from mobile users in a way that 80% of requests are dedicated to 20% of popular contents. If we set the average size of popular content on the benchmark to the average size of the benchmark itself (156199256 bytes) then the average amount of consumed and saved bandwidth for the operator will be as it is depicted in Figure 6.5 in scenarios 1&2&4. Scenarios 1&2 are the same as previously defined ones and scenario number 4 is been considered in a way to be closer to real world mobile users

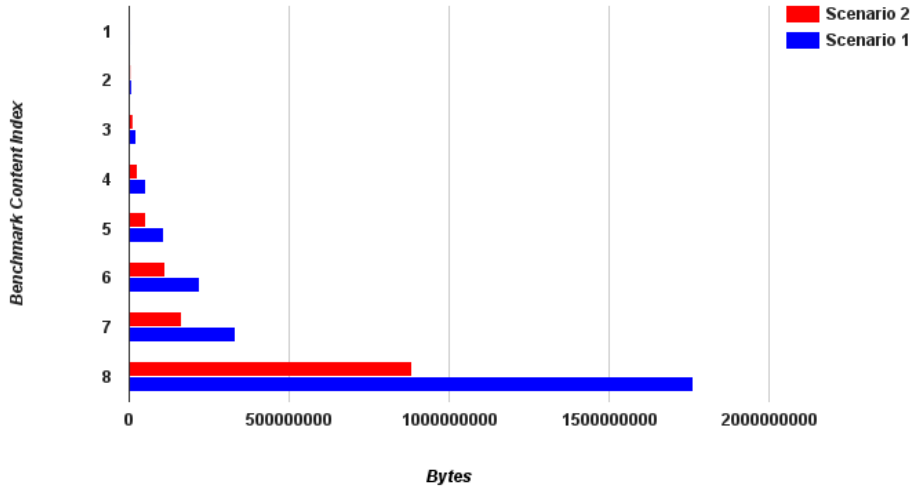


Figure 6.4: Total saved bandwidth

requests. In scenario 4, it is assumed that it takes on average 10% of requests so that all the popular contents be cached in global cache and other 10% of requests so the cached contents move from global cache to local caches (we assume there's no capacity restrictions on RACS).

6.5 User Experience

The context of user experience in transferring non-streaming content is not different than total delay of receiving the content on the mobile device which is equal to the delay time from requesting a content till the moment it is received. Hence in our benchmark we can consider the user experience is also improved by the factors given in 6.3 in scenarios 1&2.

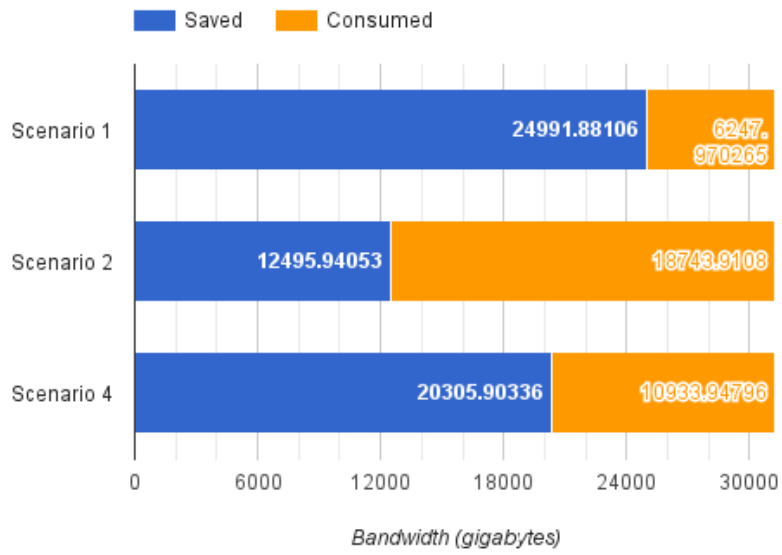


Figure 6.5: Average saved and consumed bandwidth

Chapter 7

Discussion

This chapter gives an informal view of what have been done so far in this thesis. In addition some improvement possibilities, challenges and future works are discussed.

7.1 MEC CO-Caching System

The MEC Cooperative Caching system is a distributed cooperative content caching system designed to work on NSN MEC base-stations as the distributed nodes and to test the possible improvements. A content delivery application is implemented which uses the system to benefit from MEC Co-Caching system. Mobile devices use the application to send a request for a content through the system. If the content is cached in the system and the cached copy is still valid they will receive the copy right away and if not, the content will be requested from the original source and gets cached for further usage. In other words, the idea is to evaluate reducing latency time (application delay) and saved bandwidth in a content delivery application using the Co-Caching system.

The design tries to take into consideration the desirable improvements in network latency, user experience, cache consistency, scalability and network bandwidth usage. System consisted of three components: global cache, local cache and main server. All of the components are coordinated using Apache ZooKeeper service through the MEC base-station ensemble. A cached content is stored physically in a beacon node which is the first node who has requested the content. The meta data about all the beacon nodes in the system is saved in global cache common knowledge repository. When a mobile client requests for an specific content the base-station in which it is connected to, starts looking for the content in the global cache repository to find the relevant

beacon node. If a beacon node is found having the cached content it will be requested by the base-station to send back the content directly. Consecutively the base-station will send back the content to the mobile user.

The implementation of the MEC Co-Caching system is done using Java and Apache ZooKeeper service. System is deployed on NSN RACS(the base-stations) in Aalto university NetLeap network in Otaniemi area. All the VMs running on NSN RACS are configured to be connected in a ZooKeeper ensemble. This also helps the system to be more flexible toward adding or removing a RACS to the system. Any RACS willing to join the system only needs to be configured as a ZooKeeper node in the ensemble and run the MEC Co-Caching system main server.

The evaluation of MEC Co-Caching system shows desirable results in the defined scenarios in reducing latency time, bandwidth saving and improving user experience. However there are still other parameters that can be considered to improve the performance of the system. For example using an optimized cache record placement policy in MEC servers based on mobile users behavior or making the file transfer parallel between the RACS can drastically raise the efficiency of the system. There are also challenges to take into consideration. Challenges like the possibility of interrupted RACS service or changes in the RACS connection of mobile client due to user movements. However this can be mitigated by working on hand-over and fail-over strategies specially in the moment of content transfer.

7.2 Future Works

To extend this work for future studies, there are further areas that can be explored. The MEC-Co caching system is supposed to work efficiently with popular content requests, hence it will be intriguing to simulate user requests arrival under heavy load, using a suitable distribution to observe the efficiency of the system. A distribution like Zipf [7] which is discussed in some studies to be applicable to model the popularity of objects on the Internet. Next possibility is to study optimal cache record placement algorithms to find a powerful solution based on MEC base-stations storage limits. Finally, the MEC Co-Caching system could benefit from a complete p2p approach between base-stations. For implementing this point of view, one can consider more than one beacon node for each cached content where the beacon nodes play the role of content seeds in p2p content delivery networks. Also this idea can be extended to use connected mobile devices to each base-station as an ad hoc network which participates in the created p2p network both with base-stations and other mobile clients.

Chapter 8

Conclusions

Recent expanding demand of mobile device users for cloud services leads to resource challenges in Mobile Network Operator's (MNO) network. This poses significant additional costs to MNOs and also results in poor user experience. Studies illustrate that large amount of traffic consumption in MNO's network is according to request the same popular contents by mobile users. Therefore such networks suffer from delivering the same content multiple times through their connected gateways to Internet backhaul. On the other hand, in content delivery network(CDN), the delay caused by network latency is one of the biggest issues in the way of efficient delivery and desirable user experience.

In this Thesis an aggregation between Cooperative Caching and MEC concept has been considered whereas MEC offers data locality and a resource reach environment to the cloud applications while cooperative caching aims to reduce the extra posed traffic by mobile users. Also the proximity of MEC alleviates the MEC Co-Caching total content delivery delay by mitigating network latency time and this all leads to a better user experience.

This Thesis proposes a design for Mobile-Edge computing Cooperative Caching system with the goal of efficient content delivery to mobile users using MNOs networks. The design is targeted to have some properties to achieve an efficient system. The main part of the design is dedicated to reduce network latency and total delay of content delivery, to reduce bandwidth usage in inter and intra MNO's network, to obtain better user experience, to be scalable and finally to reach a good level of cache consistency.

Furthermore, the proposed design is implemented and deployed on Nokia Networks Radio Application Cloud Servers(RACS) as intelligent base-stations supporting MEC platform. These RACS are located in Otaniemi, Aalto University area and connected to Aalto NetLeap Network. Each RACS runs an image which contains the MEC Co-caching system. In addition a mobile application is implemented on a mobile device for evaluation purposes. To

evaluate different aspects of the system all the tests have been done on a generated benchmark of contents.

In conclusion, the evaluation of MEC Co-Caching system shows desirable results in the defined internal scenarios, in reducing total content delivery delay time, bandwidth saving and improving user experience. The results illustrate that in a content delivery application using MEC Co-Caching system, previously mentioned factors are ameliorated. In addition it can be concluded that such applications can benefit more from the designed system, whatever the requested contents are larger and the popularity of the contents are higher.

Bibliography

[1] Elijah, cloudlet-based mobile computing group. http://elijah.cs.cmu.edu/.

[2] Mcc forum. http://www.mobilecloudcomputingforum.com/.

[3] Nokia Networks Intelligent base stations white paper. Tech. rep., Nokia Solutions and Networks Oy, 2012. http://networks.nokia.com/sites/default/files/document/nokia_intelligent_bts_white_paper.pdf.

[4] ZooKeeper Overview: A Distributed Coordination Service for Distributed Applications. Tech. rep., Apache ZooKeeper Open Source Project, 2014. http://zookeeper.apache.org/doc/trunk/zookeeperOver.html.

[5] Executive Briefing - Mobile Edge Computing (MEC) Initiative. Tech. rep., ETSI - European Telecommunications Standards Institute, 2015. https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/MEC%20Executive%20Brief%20v1%2028-09-14.pdf.

[6] Mobile-Edge Computing - Introductory Technical White Paper. Tech. rep., ETSI - European Telecommunications Standards Institute, 2015. http://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf.

[7] Adamic, L. A., and Huberman, B. A. Zipfâs law and the internet. *Glottometrics 3*, 1 (2002), 143–150.

[8] Cao, J., Zhang, Y., Cao, G., and Xie, L. Data consistency for cooperative caching in mobile environments. *Computer* (2007).

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

[10] Chen, M., and Ksentini, A. Cache in the air: exploiting content caching and delivery techniques for 5g systems. *IEEE Communications Magazine* (2014), 132.

[11] Chow, C.-Y., Leong, H. V., and Chan, A. Peer-to-peer cooperative caching in a hybrid data delivery environment. In *Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on* (2004), IEEE, pp. 79–84.

[12] Chow, C.-Y., Leong, H. V., and Chan, A. Peer-to-peer cooperative caching in mobile environments. In *Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on* (2004), IEEE, pp. 528–533.

[13] Cockroft, A., Hicks, C., and Orzell, G. Lessons netflix learned from the aws outage. *Netflix Techblog* (2011).

[14] Dinh, H. T., Lee, C., Niyato, D., and Wang, P. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing 13*, 18 (2013), 1587–1611.

[15] Erman, J., Gerber, A., Hajiaghayi, M., Pei, D., Sen, S., and Spatscheck, O. To cache or not to cache: The 3g case. *Internet Computing, IEEE 15*, 2 (2011), 27–34.

[16] Fernando, N., Loke, S. W., and Rahayu, W. Mobile cloud computing: A survey. *Future Generation Computer Systems 29*, 1 (2013), 84 – 106. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[17] Garroppo, R. G., Nencioni, G., Tavanti, L., and Gendron, B. The greening potential of content delivery in residential community networks. *Computer Networks 73* (2014), 256–267.

[18] Junqueira, F., and Reed, B. *ZooKeeper: Distributed Process Coordination.* " O'Reilly Media, Inc.", 2013.

[19] Kämäräinen, T. Design, implementation and evaluation of a distributed mobile cloud gaming system, 2014.

[20] Kumar, N., and Lee, J.-H. Peer-to-peer cooperative caching for data dissemination in urban vehicular communications. *Systems Journal, IEEE 8*, 4 (2014), 1136–1144.

[21] Li, Z., Zhang, T., Huang, Y., Zhang, Z.-L., and Dai, Y. Maximizing the bandwidth multiplier effect for hybrid cloud-p2p content distribution. In *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service* (2012), IEEE Press, p. 20.

[22] Lim, S., Lee, W.-C., Cao, G., and Das, C. R. Cache invalidation strategies for internet-based mobile ad hoc networks. *Computer Communications 30*, 8 (2007), 1854–1869.

[23] Liu, F., Shen, S., Li, B., Li, B., Yin, H., and Li, S. Novasky: Cinematic-quality vod in a p2p storage cloud. In *INFOCOM, 2011 Proceedings IEEE* (2011), IEEE, pp. 936–944.

[24] Marinelli, E. E. Hyrax: cloud computing on mobile devices using mapreduce. Tech. rep., DTIC Document, 2009.

[25] Mei, L., Chan, W. K., and Tse, T. A tale of clouds: Paradigm comparisons and some thoughts on research issues. In *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE* (2008), Ieee, pp. 464–469.

[26] Miettinen, A. P., and Nurminen, J. K. Energy efficiency of mobile clients in cloud computing. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (2010), USENIX Association, pp. 4–4.

[27] Ni, J., and Tsang, D. H. Large-scale cooperative caching and application-level multicast in multimedia content delivery networks. *Communications Magazine, IEEE 43*, 5 (2005), 98–105.

[28] Ramaswamy, L., Liu, L., and Iyengar, A. Cache clouds: Cooperative caching of dynamic documents in edge networks. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on* (2005), IEEE, pp. 229–238.

[29] Ramaswamy, L., Liu, L., and Iyengar, A. Scalable delivery of dynamic content using a cooperative edge cache grid. *Knowledge and Data Engineering, IEEE Transactions on 19*, 5 (2007), 614–630.

[30] Ranganathan, P. Recipe for efficiency: principles of power-aware computing. *Communications of the ACM 53*, 4 (2010), 60–67.

[31] Safa, H., Artail, H., and Nahhas, M. A cache invalidation strategy for mobile networks. *Journal of Network and Computer Applications 33*, 2 (2010), 168–182.

[32] Satyanarayanan, M. Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing* (1996), ACM, pp. 1–7.

[33] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE 8*, 4 (2009), 14–23.

[34] Spagna, S., Liebsch, M., Baldessari, R., Niccolini, S., Schmid, S., Garroppo, R., Ozawa, K., and Awano, J. Design principles of an operator-owned highly distributed content delivery network. *Communications Magazine, IEEE 51*, 4 (2013), 132–140.

[35] Taghizadeh, M., Micinski, K., Ofria, C., Torng, E., and Biswas, S. Distributed cooperative caching in social wireless networks. *Mobile Computing, IEEE Transactions on 12*, 6 (2013), 1037–1053.

[36] Tan, B., and Massoulié, L. Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM Transactions on Networking (TON) 21*, 2 (2013), 566–579.

[37] Vo, P. L., Dang, D. N. M., Lee, S., Hong, C. S., and Le-Trung, Q. A coalitional game approach for fractional cooperative caching in content-oriented networks. *Computer Networks 77* (2015), 144–152.

[38] Wang, X., Li, X., Leung, V. C., and Nasiopoulos, P. A framework of cooperative cell caching for the future mobile networks. IEEE.

[39] Woo, S., Jeong, E., Park, S., Lee, J., Ihm, S., and Park, K. Comparison of caching strategies in modern cellular backhaul networks. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services* (2013), ACM, pp. 319–332.

[40] Yadgar, G., Factor, M., and Schuster, A. Cooperative caching with return on investment. In *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on* (2013), IEEE, pp. 1–13.

[41] Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., Zhang, H., and Li, B. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *Proceedings of the 17th ACM international conference on Multimedia* (2009), ACM, pp. 25–34.

[42] Zhao, J., Wu, C., and Lin, X. Locality-aware streaming in hybrid p2p-cloud cdn systems. *Peer-to-Peer Networking and Applications* (2014), 1–16.