移动边缘计算网络的自适应缓存策略研究

ZIKPI AYAWA MATHILDE

北京科技大学

密　　级：公开

论文题目：移动边缘计算网络的自适应缓存策略研究

学　　号：S20191491

作　　者：ZIKPI AYAWA MATHILDE

专 业 名 称：信息与通信工程

2022 年 04 月 22 日

移动边缘计算网络的自适应缓存策略研究

Adaptive Caching Strategy in Mobile Edge Computing

研究生姓名：玛吉

指导教师姓名：马忠贵

北京科技大学计算机与通息工程学院

北京 100083，中国

Master Degree Candidate： ZIKPI AYAWA MATHILDE

Supervisor： Zhonggui Ma

School of Computer and Information Engineering

University of Science and Technology Beijing

30 Xueyuan Road，Haidian District

Beijing 100083，P.R.CHINA

分类号：TN92　　密　　级：公开

ＵＤＣ：　　单位代码：１０００８

北京科技大学硕士学位论文

论文题目：移动边缘计算网络的自适应缓存策略研究

作者：ZIKPI AYAWA MATHILDE

指 导 教 师：马忠贵 副教授　**单位：**北京科技大学

指导小组成员：　**单位：**

　　单位：

论文提交日期：2022 年 04 月 22 日

学位授予单位：北 京 科 技 大 学

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisor Dr. Zhonggui Ma for his continuous support, patience, motivation, and immense knowledge. The advice of my supervisors helped me during my periods of research and writing the thesis. I couldn't have hoped for a better mentor. Mr. M. Zhonggui Ma helped me work well by correcting my mistakes with guidelines that allowed me to improve the quality of my thesis work. A sincere thank you to my supervisor for his fruitful comments that were useful to me during my document preparation.

I am so proud and grateful to be accepted into the University of Science and Technology Beijing for the comprehensive education it gives me. Training at USTB really helps me to work on my thesis project. I would like to thank all the faculty for transmitting knowledge during my four years, I would like them to know that this training gives me strength and a very beneficial future.

My deepest thanks go to my father Mr. ZIKPI KOMLANVI MITRONUNYA, who always taught me that with a courageous heart, nothing is impossible and that discouragement is not Togolese. My mum Miss WILSON ANIVI for her advice and support. To all my family members, your encouragement and support have contributed to my university career at the USTB.

摘 要

移动边缘计算（MEC） 指通过将计算和存储资源放置在网络边缘（即网络中任何类型的基站），使计算和存储资源靠近最终用户，目标是减轻移动核心和减少因极端接近而导致的移动用户延迟，MEC 服务器能够托管移动应用程序并提供 Web 内容。边缘缓存系统是一种新兴技术，是网络频谱边缘的内容检索解决方案，也被认为是移动边缘计算的一种支持技术，它提供了执行缓存服务 MEC 的特殊性在于服务器可以直接在基站上实现，实现边缘缓存并确保部署在靠近移动用户的位置。然而，边缘缓存仍存在延迟的问题，同时没有为 MEC 选择正确的缓存策略，这也是他们不够为移动用户提供优质服务器体验的根本原因。

为了解决 MEC 中的延迟问题，本文提出一种基于缓存能适应 MEC 框架的解决方案，即边缘设备的自适应缓存策略 ARC。它是一种结合了 LRU（最近最少使用）和 LFU（最不常用）的方法，提出的算法能够在不删除网络用户最少请求内容的情况下，提升用户请求最多的内容。在实验阶段，我们使用一小部分视频请求，可以从 MEC 服务器或主数据库提供服务，以便能够在有序列表和 ARC 之间进行对比；将 LRU 策略作为起点，对比 ARC 算法与 LRU 算法。通过使用 ARC 算法代码，并以 MATLAB 作为辅助工具 ，确定了影响边缘缓存的参数。此外，使用客观评估方法评估了 ARC 作为 MEC 的自适应缓存策略，并使用指数图以数值显示统计数据。结果表明，本文提出的 ARC 算法能够更好地解决延迟问题，并且可以被 MEC 用来提供高质量服务。

关键词： 缓存, 移动边缘计算, LRU, LFU, ARC, QoS, Zipf

Adaptive caching Strategy in Mobile Edge Computing

Abstract

Mobile Edge Computing MEC is a concept that proposes to bring computing and storage resources close to the end user by placing these resources at the network's edge, which could be any type of base station in the network. The objective is to alleviate the mobile core and reduce latency for mobile users caused by extreme proximity. MEC servers can host mobile applications and serve web content. The particularity of MEC is that servers are implemented directly at the base stations which enables edge caching and ensure deployment in close-proximity to mobile users. However, Edge Caching still has issues reducing latency.

To solve the latency problem in mobile edge computing (MEC), we propose a cache-based solution adapted under Mobile Edge Computing (MEC) framework. In this thesis we propose an adaptive selective replacement (ASR) caching strategy in which the least frequent and least recent cache object is replaced in which cold caches and temporal frequency issues are taken into account, an adaptive predictive prefetch caching algorithm introduces runtime associative patterns prediction of time-varying popularity and uncertain contents in individual MECs, finally, a greedy collaborative algorithm is implemented which eliminates data redundancy and improves cache data exchange among all MECs. Focusing on real-world performance, we have employed the use of virtualized Linux Docker containers as computational and caching nodes in our MECs network which in turn is simulated with GNS3, furthermore, we have quantitatively evaluated the performance of several popular algorithms against our proposed approach and found a significant performance increase compared to conventional baseline caching algorithms (LRU, LFRU, FIFO, ARC) and is more efficient compared to contemporary algorithms (OPT, LFHH, MQ) under varying Zipf distributions, we then combine our proposed methodologies in MECs and evaluate the systems overall performance under a varying number of MECs (5,10,15) in the network using the MovieLens 20M datasets with hit rates approaching 90%.

Key Words： Caching, Mobile Edge Computing, LRU, LFU, ASR, QoS, Zipf

Contents

1 Introduction

At present, mobile phones have been widely used worldwide, and as time goes by, customers have more and higher requirements and expectations of mobile phones, such as performance, efficiency, environment, safety, and service quality. The rapid development of the Internet era has accelerated the transformation of user needs and promoted the formation of diversified mobile terminal services. By 2020, global mobile data traffic will reach 30 billion gigabytes per month. The vast majority of global mobile data requests are for content video. With more than 100 videos posted per minute, YouTube is poised to become mainstream, according to data. This mobile computing scheme is based on a two-layer client-server model. Compared to the usual offload computing technology, mobile cloud computing takes into account many factors, such as equipment, energy, cost of using bandwidth, mobility, context sensitivity, and location awareness [1].

Although mobile cloud computing has made many optimizations based on offloading the computing base, it still has some defects due to the distance between mobile users and the cloud. To deliver cloud services and resources more closely, MEC(Mobile Edge Computing) further deployed cloud servers in base stations as a promising solution for proximity and low latency requirements, creating a high performance, low latency, and high broadband Internet environment for users. Moreover, MEC is one of the key technologies in the current 5G network.

The rapid development of diversified mobile terminal services has brought higher challenges to related industries, such as bandwidth and storage capacity upgrade and optimization. In the traditional core mobile network architecture, mobile clients request services from content creators across the mobile network, and MEC provides cloud computing services to deliver popular content to the edge of the network, bringing the content closer to the end user. Mobile edge computing servers are not affected by latency and congestion. This is because through content caching on the server, the amount of data sent to the center for processing is greatly reduced and the impact of data traffic on the network is minimized. The cache was originally designed to allow fast and very fluid communication, as the distribution of content between peripheral servers and end users allows content to be propagated only over the network, regardless of user mobility.

The old version of the Internet had many restrictions. The old Internet also provided users with a wide variety of content, but they were not interconnected. In the process of realizing content interaction, CDN (Content delivery network), TCP/IP, and P2P are the means for reliable and fast content delivery. The common

relationship between content and location retrieval is a key point that affects the quality of service. Every time there is a content request, it needs to link to the address of the given location, fulfilling the main requirement of retrieving the content as quickly as possible, and explicitly finding an effective solution to this problem requires replacing the location with information. Based on this, ICN (Information-centric Networking) is a shift from the host-centric model, which focuses on what to offer by having named topics as the focus regardless of location.

Today, Internet users consume vast amounts of online content, much of it video. This can lead to a serious problem of runaway network congestion, requiring new systems to meet the needs of users. To address this problem, temporary content stores exist on different or multiple servers on the network to meet user needs, but this temporary storage works on caching systems, and large-scale deployment can increase the management complexity, storage space, and high availability of the cache. There are several caching strategies to determine when and where to cache content, but popularity is something to look up. The broad growth of user numbers and Internet paradigms has further increased user needs, so we need the ability to manage multiple perceptions of cache and select the best strategies. In the context of big data, storing popular content nearby is a promising solution that can meet user needs while minimizing backhaul congestion. However, local cache groups for mobile edge computing servers should be used with caution. Recent technological advances in MEC systems and their development are known to have led to increased data traffic, and their services work based on different types and very large amounts of big data. MEC servers are precisely equipped with computing, analysis, and storage tools to handle large volumes of content requests, thereby improving the network's quality of service (QoS) and quality of user experience (QoE). The local cache is part of the MEC appliance that reduces request traffic and improves response time. Caching, as a special feature, can recover data very quickly and reduce network traffic because user needs can be satisfied by edge nodes very close to the end user. It also means that edge caching can ensure connectivity requirements by decoupling the receiving and producing sides. However, due to the requirement of computational clarity, caching has advantages in mobile networks, so caching is the bottleneck system in the development of MEC systems. Thus, the peripheral cache is located at the edge of the network, closer to the end user. There are different cache policies due to storage limitations. Edge caches use different placement and replacement strategies just to allow popular content to travel over the network. In addition, mobile edge computing (MEC) servers provide the opportunity to implement edge caching, and cache placement, and create rewrite policies. A solution is available or network requirements are met. The user is the

apex of the MEC and is used for dynamic computing uninstallation, which may cause network congestion. It reduces service quality, and increases decision-making and overall system improvement issues. So the question is, how do you combine the stochastic dynamics problems of the network to make caching decisions better?

To optimize the cache system, scholars at home and abroad have carried out a series of studies. However, although they have made some optimizations on a previous basis, they still face the lack of intelligent systems to adjust decisions based on the evolution of the network, such as insufficient investment, policy combination, dynamic conditions, and temporary isolation. To solve the shortage of cache solutions, this paper designs an intelligent cache decision system and applies it to the MEC system to prove the effectiveness of our solution and meet the requirements of MEC.

1.1 State of The Arts

This section examines the most common forms of caching systems as well as a review of the academic literature on these caching systems.

1.1.1 Proactive Caching

The reactive caching policy controls whether or not to cache a certain piece of content after it is requested. It often occurs during peak traffic hours when the network is unable to efficiently handle the peak traffic. A proactive caching policy, on the other hand, calculates which contents should be cached before they are requested based on user demand predictions [96]. To improve caching efficiency and ensure QoS requirements, proactive caching typically makes use of estimations of request patterns (e.g., user mobility patterns, user preferences, and social relationships). With the advancement of machine learning and big data analytics, it is desirable to cache popular items locally before the queries come [32], [49]. Proactive caching boosts caching effectiveness by downloading popular content during off-peak hours and serving expected peak-hour demand. Bastug et al. [32] presented a proactive networking model that takes advantage of social networks and content popularity distributions to increase caching performance in terms of satisfying requests and offloaded traffic. They showed that proactive caching outperforms reactive caching. Tadrous et al. [97] investigated a technique for forecasting the popularity of services. Cache nodes can cache services proactively during off-peak hours based on their popularity. They investigated the proactive caching technique by considering resource allocation to maximize the cost reduction associated with the offloaded traffic caused by proactive caching. To boost the performance of proactive caching even further, it is preferable to optimize

caches across numerous nodes at the same time. To maximize the cache hit ratio, Hou et al. [98] used a learning-based strategy for proactive caching. Different caches in their system model can share information and contents. They used a learning method to measure the popularity of the content and then created a greedy algorithm to find suboptimal content distribution strategies. The fundamental disadvantage of proactive caching is that caching efficiency is greatly dependent on prediction accuracy. Prediction mistakes can significantly reduce caching performance [99].

1.1.2 Distributed Caching

To determine caching techniques, centralized caching employs a central controller with a global view of all network conditions. By extracting and evaluating received requests, the central controller typically tracks information about user mobility patterns and channel state information (CSI). As a result, centralized caching can achieve optimal caching performance with optimal caching options (e.g., content placement). Obtaining complete network information, on the other hand, is difficult, especially in the context of dynamic 5G wireless networks, which are expected to support an increasing number of mobile users [100]. Furthermore, the central controller must process a huge volume of traffic, putting a strain on both the controller and the linkages between the controller and network entities. In that instance, the central controller may become the mobile caching system's bottleneck. Cache nodes in distributed caching, also known as decentralized caching, make decisions (e.g., content placement and update) solely based on their local information and information from nearby nodes. [49] employs distributed caching, in which adjacent base stations (BSs) are collaboratively optimized to maximize the cache hit probability. The total cache size visible to the user can be raised by requesting contents from numerous surrounding caches. The believe propagation (BP) method has been proposed as an efficient method for distributing resource allocation problems in wireless networks. In BP, a complex global optimization problem is typically split into several subproblems that can be tackled successfully in a parallel and distributed fashion. Li et al. [102] addressed the file location issue to reduce the average file downloading delay. Their network design consists of a macro base station (MBS) and cache-enabled small base station (SBSs) to which UEs' queries are preferentially sent. They organized the files in the file library into file groups and anticipated that each SBS could only cache one of them. A factor graph, a bipartite graph consisting of factor nodes and variable nodes, was used to suggest a distributed BP algorithm. A factor node represents a user's utility function and is related to the average file download delay. Each variable node represents an

SBS's cache status vector. Only when a UE is covered by an SBS can an edge connect the UE (factor node) to the SBS exist (variable node). The BP algorithm is then implemented by transferring messages between the factor and variable nodes iteratively. In each iteration, the message is represented by a probability mass function based on the UE's utility function; each variable node updates its message to be delivered to linked factor nodes and each factor node updates its message to be transmitted to connected variable nodes. When the messages do not change, the BP algorithm quits. Unlike [102], which assumed that each UE could only be served by one BS, Liu et al. [92] developed a distributed BP algorithm to minimize average download delay in cellular networks with many cache-enabled BSs. The data transport technique is determined by the cache location. If only one BS caches the requested file, the BS will send it directly to the user; otherwise, numerous BSs can send it via cooperative beamforming. Each BS in their BP model collects local information (e.g., user requests and CSIs), executes computations, and exchanges messages with neighboring BSs iteratively until convergence. They demonstrated that the distributed BP algorithm is less computationally intensive than the centralized one.

1.1.3 Cooperative Caching

Because the caching area in a BS is generally tiny, establishing a caching policy for each BS separately may result in insufficient cache use. This occurs when certain caches are overcrowded while others have many empty places. To solve this issue, cooperative caching policies to boost caching efficiency have been proposed. BSs can share cached contents in cooperative caching [99]. However, the delay in searching for and obtaining content from other caches may be significant and should be considered. To enable cooperative caching, network nodes must be aware of the caching status of other nodes via information exchanges, which may result in considerable signaling overheads. As a result, we must discover a way to convey the caching status with the least amount of overhead. Jiang et al. [103] created a cooperative caching policy for HetNets that allows users to retrieve content through femto-base stations(FBSs), device-to-device(D2D) communications, or macro base stations (MBS). To reduce average downloading latency, they structured the cooperative content placement and distribution problem as an integer linear programming (ILP) problem. The original problem was then decoupled into two smaller subproblems that could be addressed more efficiently using a Lagrangian relaxation approach. The Hungarian algorithm was also used to formulate and solve the content delivery problem. Most studies on cooperative caching assume static popularity; the joint consideration of cooperation and learning of time-varying

popularity requires additional research. [104] Song investigated the content caching problem with an unknown popularity distribution. They incorporated popularity distribution learning and then jointly optimized content caching, content sharing, and content retrieval costs.

1.1.4 Coded Caching

In a traditional switching network, packets are forwarded one after the other: two packets are present in the node at the same time; one of the two packets is forwarded while the other is queued, even if they are both headed to the same destination. This classic packet forwarding system necessitates separate transmissions, reducing network efficiency. Network coding is a technique that combines two independent communications into a single coded message and sends it to its destination. The network node divides the coded message into two original messages after receiving it. To enable network coding, transmitted data is encoded at network nodes and decoded at destinations. As a result, the network coding technique uses fewer transmissions to send all of the data. This technique, however, necessitates coding and decoding processes, resulting in higher processing overheads for network nodes. Efficient packet transmissions can reduce the complexity of network coding [105]. Files in the file library are typically separated into coded packets in network coded caching, and any linear combination of these code packets can rebuild the full original object [43]. The file library, for example, contains the file C, which is separated into $C_1 \oplus C_2$. Because cache storage is limited, a user who requests file C for the first time only caches packet C_1 after receiving file C. When a user requests the same file C for the second time, the BS merely needs to send C_2. File C, on the other hand, must be transmitted both the first and second times in uncoded caching. As a result, coded caching reduces network traffic ($C + C_2 < C + C$). Maddah-Ali and Niesen [106] combined optimized caching and programmed multicast delivery and showed that when demand for cached information is consistently distributed, the joint optimization issue can improve caching gain. They also demonstrated that a random caching technique achieves near-optimal performance for coded caching [107]. In [108], they also demonstrated how caching gain can be obtained via coded multicast transmissions. In [108], they proposed a decentralized coded caching scheme and discussed how to deal with asynchronous user demands, nonuniform content popularity, and online cache updating scenarios. Most works only investigate single-layer coded caching; however, Karamchandani et al. [109] suggested a hierarchical-coded caching technique that takes a two-layer hierarchical cache into account. They first used coded caching methods in each layer, then connected the two layers by allowing coded multicasting across various layers.

1.1.5 Probabilistic Caching

Unlike wired networks with fixed and established topology, wireless networks confront uncertainty regarding which user will connect to which BS because of indeterminate user locations and the variability of user requests. When a user moves from one cell to another during content delivery, caching in wireless networks becomes more difficult. A probabilistic caching policy, in which content is placed in caches based on some random distributions, is one way to solve this problem. To account for the uncertainty, Blaszczyszyn and Giovanidis [85] treated user positions as a spatial random process. They optimized the likelihood of each material being cached at each BS to maximize the cache hit probability. They also demonstrated that the widely used greedy approach, which caches the most popular files, cannot always guarantee optimization in a general network unless there are no BS coverage overlaps. Ji et al. [43] developed the random caching approach in D2D networks where UEs are uniformly dispersed in a grid network and can communicate contents with each other. They pointed out that the disadvantage of deterministic caching is that optimal cache placement cannot always be accomplished without errors, especially when D2D caching is considered. They demonstrated that their random caching strategy, in which users make arbitrary file requests, performs better as the network size grows.

1.1.6 Game Theory based Caching

Multiple parties cohabit in wireless networks, including service providers (SPs) who provide content, mobile network operators (MNOs) who operate radio access networks (RANs), and mobile users who consume various contents. When using a given caching approach, the benefits of multiple parties may conflict. Bringing more material to BSs, for example, benefits consumers while raising MNO costs due to higher storage and power usage. Because each party is simply concerned with its profit, competition among them is unavoidable. To effectively compete and provide a high overall customer experience, game theory is used to examine the interactions between these parties. An auction game is a good way to describe the competitiveness among SPs. In this configuration, cache storages are treated as auction items, with SPs paying the winning bid to MNOs. The MNO should be in charge of the auction process. Hu and colleagues [110] used game theory to examine how individual parties' selfishness affects overall caching performance by taking into account the relationships and interactions between them. They looked at two scenarios: SBS caching and D2D caching. The former proposes an auction game to overcome the problem of several SPs attempting to cache their contents into SBSs

with limited cache storage. They used a coalition game for the latter to examine how a cooperative group may be established to download content jointly. They expanded their work by presenting the notion of caching as a service in [12], where they used wireless network virtualization technology and each SP was required to pay for the MNO's SBS cache storages. To define competition among SPs, a multi-object auction technique was proposed. Because all SPs cache more content to increase service performance, they intended to compete for limited cache storages as bidders. The utility function is associated with the typical content download file. Their mechanism involved a series of auctions that were solved using the market matching algorithm [111]. Hamidouche et al. [112] hypothesized that all SBSs in a cache-enabled small cell network may choose between wired links, mmW, and sub6 GHz bands for backhaul. They devised a backhaul management minority game in which the SBSs are the players and independently choose their backhaul link types and the number of files to download and cache from the MBS without compromising the quality of service (QoS) of current requests. A minority game is distinguished by the fact that participants prefer the action chosen by the minority group. The existence of a single Nash equilibrium was then established. Hamidouche et al. [113] used the game theoretic approach to identify the content placement strategies for SPs by taking into account the social links among UEs. Between SPs and SBSs, a many-to-many matching game was developed, in which each file in SPs can be matched to a set of SBSs. SBSs prefer to store more popular files, whereas SPs express their preferences based on the average file download time. The stable solution can be found by iteratively updating the matching solution based on the preferences of the SPs and SBSs until neither of them can find a better preference.

1.2 Innovations of the thesis

- A cache replacement technique that chooses victims based on popularity, recency, and network cost. Through its selective caching strategy, the scheme may identify cold cache objects. Furthermore, the issue of temporal frequency has been addressed.

- An adaptive predictive caching method based on association rule mining has been developed to forecast cache behavior patterns based on historical occurrences. Cache objects are also prefetched when necessary to maximize cache storage and reduce network latency.

- A greedy collaborative caching approach has been proposed for sharing and retrieving cache among MECs while lowering cache redundancy in the collaborative space

2 Overview of the Mobile Edge Computing

2.1 Introduction

ETSI (European Telecommunication Standards Association) first published mobile Edge computing MEC in 2014 [2]. Mobile Edge computing is an environment that provides IT services and cloud computing power at the edge of the mobile network, thus bringing end users closer to their network. Its goals are to achieve minimal latency, provide an improved user experience, and ensure that service delivery meets the requirements of the network [3]. MEC increases edge response time and speeds up content, business, and applications to optimize the mobile user experience on the network and provide efficient services. Different wireless systems or applications may have different performances, and MEC networks can meet users' latency requirements. What differentiates MEC from traditional mobile networks is that it provides cloud computing, so it can bring popular content to the edge of the network with zero distance between the content and the end user. Applications and analysis at the mobile edge computing server level are not affected by latency and congestion [4]. By performing caching or analysis on the MEC server, the size of the data sent to the core for processing is greatly reduced, minimizing the impact of data traffic.

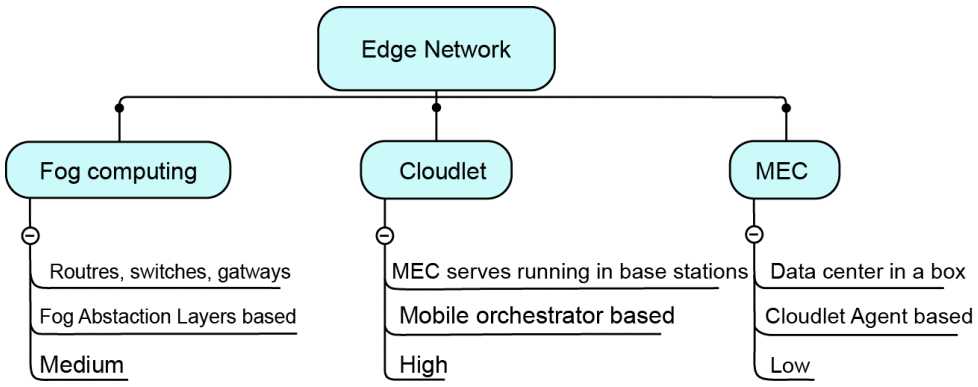

Figure 2.1: Diagram of the MEC

2.2 Mobile Peripheral Network

The primary goal of mobile peripherals is to bring network components and users closer together, and the virtualization systems (NVF) and programmable networks (SDN) used by mobile peripherals enable them to achieve their goals. Network resources mainly consist of computing, caching, storage, and communication. It is worth noting that in some pieces of literature, the cache is

included in computing resources [5]. This article discusses caching strategies in moving edge computing. An advanced calculation scheme is also proposed.

2.3 Mobile Edge Computing

Mobile edge computing aims to reduce latency by enabling nearby computing and storage and allowing services to be hosted on edge. The main idea of mobile computing at the edge arises internally we note, for example, the improvement of the quality of experience (QoE) and the improvement of the quality of services (QoS) of the users by reducing latency and providing personalized services, will therefore serve to optimize network efficiency. It also offers important services for machine-to-machine effects, large-capacity data analysis, and offloading. Close integration with the environment facilitates an understanding of network and user characteristics. Illustrating MEC's external scenarios, including technology integration and support use cases that MEC can enable are: Connected Vehicles, Healthcare, Augmented Reality, Gaming, Distributed Networks, Services IoT, etc. The elements representing the architecture of the MEC are:

(1) Mobile devices, connected to the internet through the backhaul;

(2) MEC servers there are two different kinds such as the server directly integrated into the base station (BS) and the server not directly integrated into the base station (approval site);

(3) The Edge cloud is responsible for controlling network traffic and accepting and hosting various mobile edge applications;

(4) the public cloud is a cloud infrastructure hosted on an internet network.

2.3.1 MEC

MEC servers provide the following services:

(1) Offloading: through offloading, the MEC increases storage computing limits, device battery, and bandwidth, capacitating it and minimizing power consumption;

(2) Edge Content Delivery: MEC servers offer content services to edge networks. They act as distributors of local content nodes and use cached content;

(3) Aggregation: instead of sending all the data on a case-by-case basis to the base routers, the MEC servers can aggregate similar traffic and therefore reduce routing;

(4) Local connectivity: as traffic flows through MEC servers, the servers can split the traffic flow and redirect it to other endpoints.

(5) Augmentation: When there is the availability of additional data at the base station site, this data may be broadcast to optimize the quality of experience.

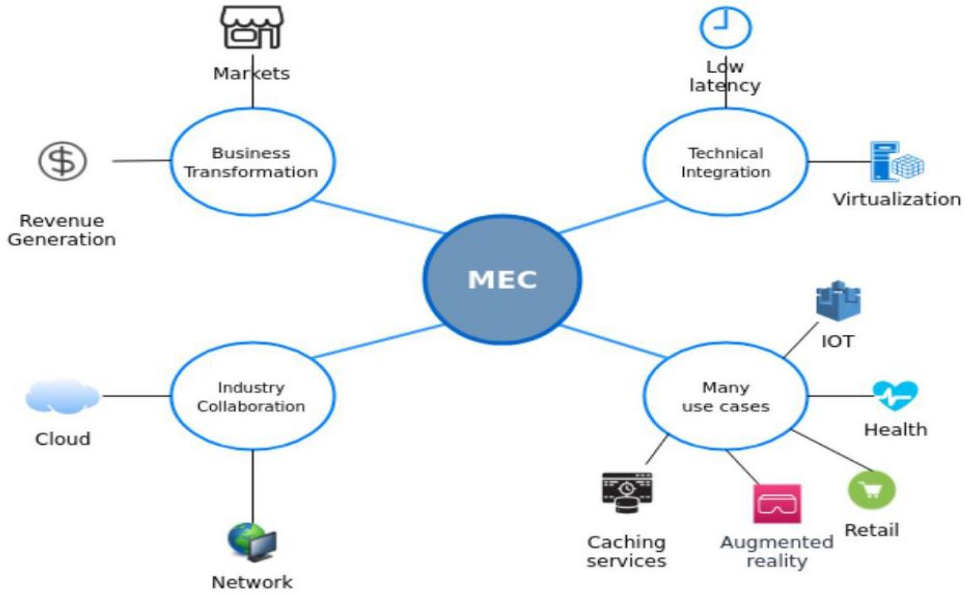

Figure 2.2: Exterior scenarios of MEC

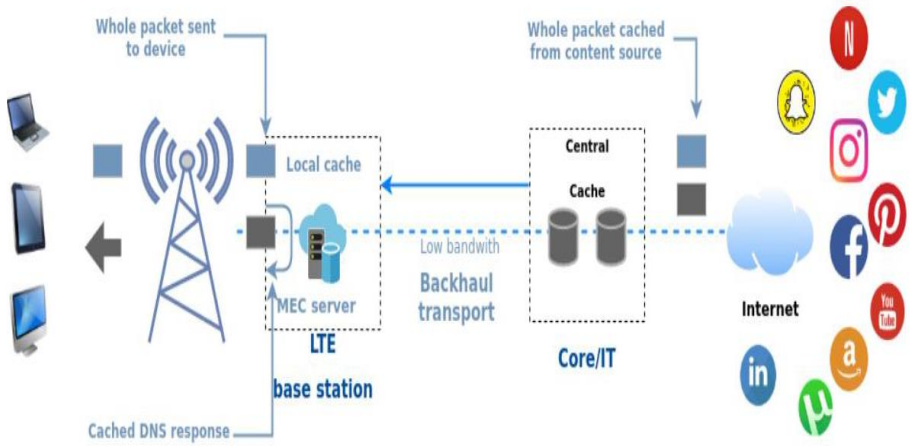

Figure 2.3: MEC mobile edge computing architecture

In short, edge computing is local and isolated, allowing direct access to local resources that work independently of the network. In addition, the MEC node is very close to network users and information sources, so it has the following functions :

(1) it can easily obtain the information for analysis, to analyze user behavior;
(2) Provide context-aware services, such as edge services near terminal devices;
(3) Due to its proximity to the edge and being part of a wireless network, it can also reduce latency and increase bandwidth.

Mobile Edge computing exposes capabilities that allow MEC servers to implement and host many different types of applications that cannot run on end-user devices, depending on different obligations, especially at storage capacity levels. The implementation of MEC systems requires support for various flagship technologies such as servers and virtualization, without forgetting components and all functional elements. Two researchers proposed a classification of moving edge computing. [6] The figure above shows us that the taxonomy shows that the MEC system has many properties and contains many users and organizers who represent a different number of participants with different roles. In terms of services, MEC offers a wide range of applications in all areas, aiming to improve the quality of the experience. Use an available access technology to allow devices to access mobile devices that are part of an edge computing environment. The biggest feature of the MEC system is its goal. As mentioned above, the main goal of MEC is to reduce latency and energy consumption. Finally, implementing MEC systems requires the support of many enablers, represented by different types of technologies, who will help MEC provide good services to users of mobile networks.

Table 2.1 The Potential Need for a Mobile Edge Computing Server

	Without MEC servers	**With MEC servers**
Content Optimization	Traditional content optimization is done to meet the expectations of the user. It is based on the user's web history stored in the database	The content optimizer can be stored on the MEC server to optimize the performance of the quality of experience, the quality of the network, and new services which can be added in the future
Unloading and aggregation	Traditional calculations such as aggregation and offloading are not performed in the device, and to provide a solution to this problem, the applications are separated into small patches and performed in the central network	Offloading tasks to the peripheral server without transferring tasks to the central network will certainly reduce latency.
Big Data Analytics	The data collection procedure is done from the edge and then transferred to the core network and takes high bandwidth and good latency	MEC, Mobile Edge Computing servers can perform the big data analytical procedure, and the results are sent to the network core. As a benefit, there will be a reduction in bandwidth consumption and improved latency.

2.3.2 Edge Computing

In truth, the benefits of edge computing are many, from workspace to productivity and security, especially for businesses. Some advantages are:

(1) The efficiency of operations: the MEC ensures the optimization of operations continuously by processing in a short time the data of large volumes onboard or on the local sites where the various data collected are located. Compared to other operations where data is collected from centralized clouds, this operation is more efficient;

(2) Employee productivity: the edge network makes it easier for companies to provide data very quickly. It is the data that employees need to perform their tasks as efficiently as possible. The work is done in very intelligent workplaces that provide maintenance and automation peripheral computing plays a very important role in the proper functioning of tools or equipment without any fault or errors;

(3) Very fast response times: without going through the centralized location of the cloud and data centers, companies manage to process data very quickly and, above all, in a very reliable way, on time. Instead of addressing data latency, network bottlenecks, and data quality failures that are often faced when sending different information from thousands of sensors simultaneously to a central network, IT edge other uses the devices available onboard the network to immediately alert the personnel and the basic equipment to ensure the security and other failures that could face, to take the possible measures as quickly as possible.

(4) Improved safety in the workplace: in the workplace where faulty equipment or proposed changes to working conditions can spoil everything or even cause worse damage, edge computing and IoT sensors can ensure in terms of personal safety. For example, on remote industrial sites, predictive maintenance and data analyzed in real-time near the equipment site can surely help to perform worker safety and minimize environmental impacts.

(5) Functionality in remote locations: MEC ensures easy use of data received at remote sites where internet connectivity is faulty or bandwidth is often limited. Operational data such as water quality can be monitored by the sensors, and the latter can be processed if necessary. Once the internet connection is available, the data concerned will be transmitted to a data center for analysis and processing.

(6) Reinforce security: in any company, the security risk of adding a large number of sensors and devices connected to the Internet to their network is a big problem. The MEC reduces the risk by encouraging companies to process data locally and save it offline. This decreases the rate of data transmitted over the network and allows businesses to be less exposed to issues related to security threats.

(7) Reduced IT costs: the MEC allows companies to improve their IT expenses by processing each data locally than processing it in the cloud. Aside from reducing

processing and cloud storage costs for companies, the MEC lowers sending costs by eliminating data that is not useful at the location where it is collected.

(8) Data sovereignty: during the collection, processing, storage, and use of user data, companies must follow the security rules, especially the confidentiality of the data of the region or country where it is collected and stored their data. Moving data to the cloud or main data centers across national borders often makes adhering to data sovereignty regulations very difficult again MEC allow companies to be reassured that they are following the local directives on sovereignty given by processing and recording the data at the same level where they were collected.

2.3.3 Edge Computing Uses cases and Examples

Smart devices and MECs could easily change the way data is handled in different industries around the world. We'll mention some use cases in the following sections.

(1) Branches: Smart sensors and devices decrease the total resources needed to operate a company's secondary offices. Let’s not forget the controls for heating, air conditioning, and ventilation connected to the internet, the sensors that signal when necessary to repair the photocopiers, and surveillance camera security. By simply sending an alert to the necessary devices at the main data center of a company, the MEC makes it possible to avoid the problem linked to the servers' congestion and the level of delay while increasing the time to a large level of response to installation problems.

(2) Manufacturing: Sensors mounted in large factories can be used to monitor equipment to detect routine maintenance issues and operational problems. They can also ensure the safety of thousands of workers. Otherwise, the various intelligent equipment present in a factory and a warehouse considerably increases productivity, ensures good quality, and reduces production costs. In short, keeping the data and analyzing it in the factory instead of transferring it to a centralized data center avoids the problem of costly and potentially dangerous delays.

(3) Power: Power companies increasingly use IoT sensors and edge computing to increase efficiency, automate the power sector, minimize maintenance, and supplement network connectivity delays in remote areas. Utility towers, wind farms, oil plants, and many remote power sources can also be equipped with IoT devices capable of withstanding harsh weather conditions and other challenges caused by the environment. These devices can process data on the energy site or at the edge and transfer only the data required to the core data center. In the oil and gas industry, peripheral computing and sensors generate timely and important safety alerts that alert important personnel to key repairs and unsafe equipment operations that can lead to catastrophic hazards such as fire.

(4) Agriculture: MEC also contributes to efficiency performance and energy efficiency yields. Weather-assisted IoT drones and sensors can provide temperature monitoring and equipment enhancement, analyze environmental data, improve the quality of water and plants used on plants, and provide very efficient time collection. The MEC offers a very economical use of IoT technology, even in remote locations where network connectivity is limited.

2.4 Information-Centric Networking

In the 1960s, the distribution of resources was one of the network's key objectives. The advent of the internet was already gaining momentum and helping to achieve this objective. Computers and broadband devices were hosted and distributed on sites. Nowadays, the Internet Protocol (IP), which defines the point of connection, has been established to help the sharing and offering of data. Internet designers had not imagined its enormous growth and the different ways of using the Internet, especially in the case of large-scale content using the cloud and Edge computing. On the other hand, resources are shared everywhere, while the end-user only cares about data, not locations. The other Internet protocol takes care of the location (where) and what data to provide (what). Many applications such as games, IoT, big data, etc., don't care about the location rather they care about data. Therefore, to solve the problems mentioned above, it is necessary to abandon the intellectual priority, a paradigm that has been set up for the distribution of resources on a limited number of sites over 50 years old. There are replacements for shipping bits. Hence the proposal of the network to focus on information to directly name the data, which solves the problem of the limitation of the semantic point-to-point IP [7].

Information-Centric Networking (ICN) is a concept used to refer to all classes of internet architectures that rely on the content as the focal entity instead of host-centric network architecture. The ICN can be defined as a paradigm shift in networking that promotes applications, services, and networks to interact thanks to the primitives of the network centered on the information to be shared. As shown in the figure below, the design of the ICN is given to the hourglass, followed by the architecture of internet protocol: the network layer in the form of an hourglass is very transparent, which favors all applications to run fairly simply on top of it, so it can run on most link-layer technologies.

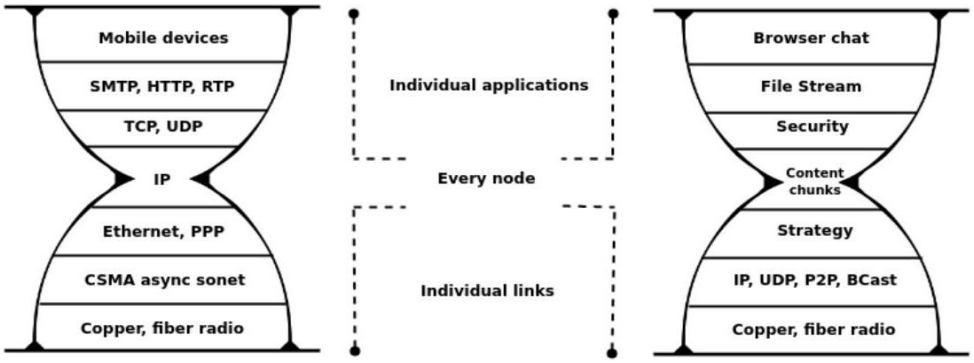

Figure 2.4: ICN's hourglass approach (right) versus the current internet (left)

ICN contributes to many projects to solve problems of the architectures we have today. As a solution, we can quote the security, the occupation, and the use of the resources without forgetting the mobility and the evolution of a network. Despite its success, the ICN still faces many challenges, such as caching, routing, security, and naming [8]. Out of all these challenges, caching is considered the major challenge because it needs a good strategy to make the decisions like how to cache through the network. It is very necessary to switch to a new network paradigm to solve the problems and remove the current limitations of the host-centric communication model. In the ICN, the data no longer depends on its location, application, and storage, promoting caching. The following table summarizes the DCI's key concepts and principles and shows how each seeks to address some current Internet issues.

Table 2.2 Key concepts and principles of ICN

Key concept	Description
Name the data	ICN primitives are focused on naming data. The name of each data must be persistent and unique. Name identity must also be independent of transmission path and storage. Either in a self-certification scheme or a hierarchical scheme.
Data recovery	The retrieval of requested data is divided into two tasks: the first step is discovering the content, which means routing requests to the location of the data. The second stage is the reception of the content, which is the system of transmission of the data to the requester. To satisfy future requests, the content caching system along the path is used.
Data security	There is a division between the security in the ICN and the data containers and the security is based on itself, which favors the verification of the content wherever it is stored and whoever wants to do the recovery while including a signature directly.

2.5 Summary

The current Internet is riddled with problems that plague users, such as spam and ignorance of information within networks, as well as various denial-of-service attacks that pose new challenges to the industry's security, mobility, quality of service, and caching requirements. For now, users are more focused on the information they need. As a result, there is a growing need to refocus communication on content access rather than host-to-host interaction. Some researchers claim that mobile edge computing and the ICN information center network may be the key systems for the general Internet. More and more people see data content as a selected entity on the Internet. Network architecture, and the ability to cache server content from various locations on the edge of the network. The next chapter introduces fuzzy logic and reinforcement learning. We use these two concepts as examples to build a cache system.

3 Theoretical Basis of Mobile Edge Caching

3.1 Introduction

Caching at the network edge is a very emerging technology suitable for content retrieval. Edge caching is also an enabling technology, much like the head of mobile edge computing, which gives important opportunities to perform caching services. Specifically, MEC servers are deployed directly in BS base stations, ensuring deployment close to end-users and enabling edge caching. This chapter shows a general overview of the caching system and mentions its different techniques and principles. It also contains a description of performance measures and compares existing caching mechanisms.

3.2 Cache Location and Cache Strategies

he caches simply stores the data in a temporary database through which we can receive files without reporting back to the original server, thus saving extra network traffic and recovery time. The load on Web servers and the Internet drives caching technology. Caching is very valuable in the network industry. Three features of web caching make it very attractive to all Web participants without forgetting users, content creators, and network administrators [11]:

(1) Caching minimizes network bandwidth usage;
(2) Caching reduces mobile network latency;
(3) Caching reduces loads on the origin server.

3.2.1 Caching Locations

In the mobile device network, caching occurs in the following places:

(1) SBS caching: small base stations should be widely deployed in the heterogeneous networks that will emerge. The difference with MBS is that caching in SBS makes content even closer to users. It will respond faster to user queries. The texts [12] [13] and also [14] [15] have studied the performance of caching in small base stations (SBS) [16].
(2) MBS caching: caching and the edge server are deployed in the micro base stations of heterogeneous networks. At this level, caching can be performed reactively and proactively. With caching based on the micro base station and edge server, system capacity can be significantly improved, and data transmission delay could be reduced since contents will now be available at the edge, ready for users. In [17], the performance of MBS caching was investigated.
(3) Device-to-device (D2D) communication: D2D is an architecture also included in the 5G mobile networks. In D2D, the end edges can be used for content storage and caching. This will greatly reduce transmission times. D2D caching can be done collaboratively by forming clusters or individually between end-

users. A scheme of D2D communication based on caching has been proposed in [18] taking into account the relationships between different users and their common interests. In the literature [19], an opportunistic cooperative D2D transmission scheme exploiting the caching capability at the device level is proposed.

3.2.2 Caching Strategies

A caching policy is a policy for managing temporary storage resources. Typically, a caching strategy decides when what, and where information should be stored. This part will present the principles of caching techniques and then the proposed caching strategies.

principles of caching strategies

For a broad understanding of caching strategies, some concepts must be understood. We will place our work where the need for an algorithm or a caching policy will be. Everything is summarized in the following part:

(1) The conventional caching policy: in [20] [21], the content replacement policies are adopted in a large number of caching policies so it is the less frequently used content policy (LFU) and least recently used (LRU). These strategies are very effective and simple with objects of the same size. Which makes these policies ignore download latency and object size? There is also an MPV policy, it is a proactive caching policy used in Content Delivery Networks. MPV policy caches the most viewed videos based on the worldwide popularity distribution of the video [18]. The cache size of the RAN is more limited than that of the CDN, so the probability of success obtained by the MPV policy could be too low for the RAN cache.

(2) Policies based on the User Preference Profile: a caching policy taking into account the user preference profile has been proposed in [22]. So, there is a big difference between the popularity of a national video and the popularity of a local video, and users can show strong preferences for specific video categories. So, we can say that the UPP is the probability that a user requests videos from a specific video category.

(3) Policies based on learning: generally, it takes time to know if the content is Popular or not. It is important to follow the evolution of content called Popular content closely. Learning-based caching policies are proposed while relying on machine learning technology [23]. The authentications described in [23] solve the problem of grant caching in SBS from a reinforcement learning view. With coded caching, the caching problem is reduced to a linear program which, taking into account network connectivity and the coding scheme, performs better than the coding scheme. The solution in the literature [24] solves the cache replacement problem with a Q-LEARNING-based strategy.

(4) Non-cooperative caching: Some caching policies directly cache content to each base station regardless of the relationship between the base stations. The scheme proposed in [17], decides to cache lUPP active uses in a specific cell without considering the impact of caches in other cells. The literature [24] shows us that the cache replacement problem models as the Markov Decision Process (MDP) are solved in a distributed manner using the Q-Learning method, without sharing additional information about the cached data between databases station. This strategy is above conventional strategies like LRU, LFU, and randomized strategy.

(5) Cooperative caching: this caching is not to be neglected; the expectation is that many works have been carried out to study the cooperation between caches when designing caching policies. A lightweight cooperative cache management algorithm is developed to maximize the volume of traffic served from the cache and minimize the cost of bandwidth [25]. The [26] shows the cooperation between the user equipment and the base station for caching and content delivery has been studied. The cooperative caching problem is the integer linear programming problem solved using the sub-gradient method. The content delivery policy is defined as an imbalance attribution concern and resolved using the hungria n algorithm. In [27], the authors explore the exploitation of the SVC scalable video coding technique in collaborative video caching and inter-cell scheduling to improve caching capacity and user QoE. The authors study in [28] the cooperation between the caches in the RAN and have as a result, the optimal redundancy ratio of the content cached in all the base stations. The [29] literature shows us how cooperation between D2D users is exploited for cached D2D communication. The method of placing content on the network encoding is seen in [30]. The strategy multiplies the capacity of data available to users and results in equal sharing of data at the same time. In the [31] literature, the authors jointly set up the caching and routing scheme to maximize the content requests offered by small base stations under the BS bandwidth constraint. Therefore, the concern is reduced to a placement localization problem and is solved using bounded approximation algorithms. Given the evolution of caching, we will be using caching of different file types, which means they are points not to be overlooked.

Large data files or multimedia data: many existing pieces of literature [17] [32] and [27] have studied how large multimedia data is cached, specifically video files. What characterizes multimedia data is the large number of users who have a common interest in popular videos. So, caching popular files in the RAN can be beneficial with a high hit rate.

(1) IoT data: The low-throughput monitoring, automation, and measurement data that IOTs generate and run on many devices should be cached to better (reduce) the total traffic load. IoT data has a short lifespan which is largely the opposite of media files. Therefore, this requires different types of caching[33].

(2) The most popular contents have the privilege of caching. in popularizing content, large user demand is cached in the periphery of networks.
(3) Static model: Recent work on caching assumes that the popularity of content is static and adopts IRM, which is an independent benchmark model. The demand for content is generated following the free fish process, whose rate is linked to the popularity of the content modelizer thanks to the power law [34]. The ZIP model observed in web caching is the most popular model to use [35].
(4) Dynamic model: Considering the blockade that the static MRI model, in other words, cannot give the real popularity of the content, which varies over time see [34], another dynamic model of popularity is proposed in [36]. This model is called the model of SNM shooting noise. SNM uses a pulse of two parameters to model each content: this is in the form of a diagram (the height represents the immediate popularity, and the duration represents the lifetime of the content). The [32] literature analyzes the statistical properties of user-generated content (UGC) popularizing distributions and reviews the probabilities of taking advantage of "long-tail" video requests.

Performance measures

The measures taken for the evaluation performance of caching mechanisms are:

(1) Cache hit rate: hitting a cache means that the content request can be successful. Otherwise, if the cache cannot resolve a request, a cache miss is noted. Hence the cache hit or hit ratio is defined as the ratio of cache hits to the sum of the number of cache misses and the number of cache hits. Therefore, a very high cache hit rate can result in reduced access latency and server load.
(2) Remote caching: let D(s) be the time or the distance between the MEC server and the requestor with the presence of the cache that will serve the request. The hit distance is defined as the distance D(s) average overall requests. The shorter access time is better for a good user experience.
(3) Cost: cost is defined as the appropriate time to retrieve content. Therefore, the operational cost should be low to help the caching mechanisms apply on large-scale networks.

Typically, caching at the edge of the network means that popular content is cached to peripheral nodes, such as MBC(macro station), SBS(small base station), and even UE(user device), as shown in Figure 3.1. Edge caching can significantly reduce redundant data traffic and effectively improve user service quality (QoE)[37].

In terms of cached content, as Internet penetration increases, so does the amount of content that needs to be stored, such as video, audio, and other Internet data. However, not everything can be cached due to the storage space limitations of peripheral nodes. The popularity of content affects or answers the question of what content to cache. It represents the probability that content can be requested within

a certain time. This very interesting property should be a major factor in the caching mechanism. Much recent work on edge caching assumes that the popularity of content followed the release of static Zipf[38].

However, because the number of users connected to a single peripheral node varies, the end user's wishes can change from time to time, as only the most popular content is not sufficient as a parameter to reduce hidden risks. The cache itself that moves the edge selects what to cache with the support of the cache system policy. Caching strategies will achieve different goals. Examples of goals include traffic offload, experience quality, energy consumption, and many other goals, the only goal being to maximize cache rates. Different caching policies fall into different categories based on their characteristics:

Cache coordination

- Coordinate: caches are important in the exchange of information to know concretely where to cache content, and at the same time, this will avoid unnecessary storage of too much content of the same content. The coordination cache is defined as a shared session function that allows many instances of a session to broadcast content changes between them so that all caches are updated;
- Uncoordinated: In the uncoordinated sector, each of the caches works in different ways.

Cache size

- Homogeneous: in the base station (BS) all caches have the same size;
- Heterogeneous: each cache with its size and sizes differs according to the corresponding cache.

Cooperation between caches

- Cooperative: the different caches cooperate by producing cache states by other caches to know the different kinds of states like in [39].
- Non-cooperative: Each cache makes its caching decision without broadcasting cache state information.

Where the content is cached

- Along the path: only caches content that traverses the entire download path;
- Out of Path: Caches captivating content outside of the download path.

In general, conventional caching policies are separated into two main phases: the placement phase and the replacement phase. The placement phase is the phase that decides if caching should be done and when to cache content. And the replacement phase, others determine which data to remove when there is a problem with free storage space.

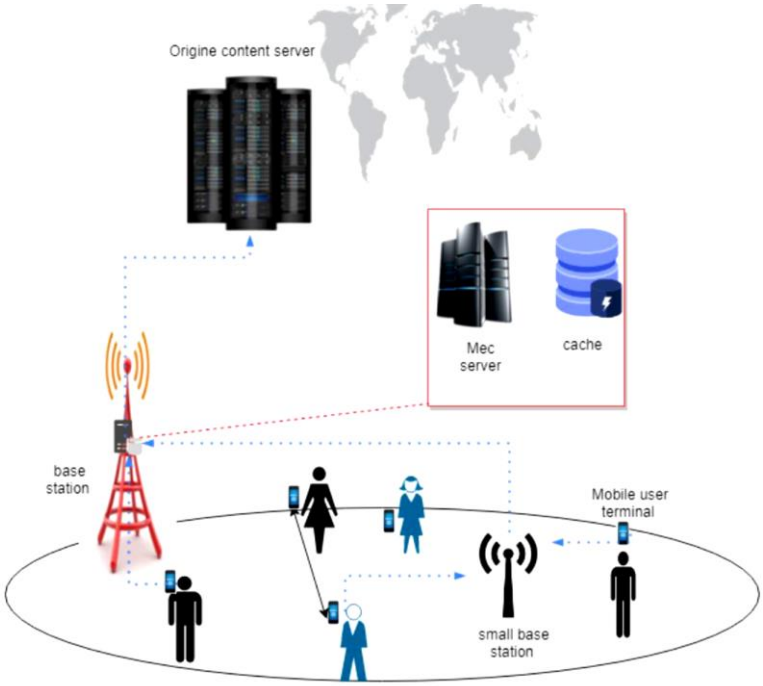

Figure 3.1: Caching Architecture

State of the art on caching strategies

To know the best sector of mobile edge caching, different researchers have used the classic web, caching algorithms for the replacement and replacement phases characterized by the popularity of the content. In this literature, we note different placement strategies like (LCE) copy everywhere, a default caching system in almost all cache designs. The LCE minimizes upstream bandwidth demand and reduces latency [40]. In this mechanism. Popular content is cached in all available caches along the route. LCE is a non-cooperative homogeneous caching system. Leaving copy down (LCD) and moving copy down (MCD) are different methods to implement both heterogeneous and cooperative caching to reduce overhead. Probability mechanisms have been used enough, such as the prob(p) policy. This has been used as a fallback system in the [41] [42] [43] works of literature. This mechanism grants each content a probability p in advance and does not cache the content for a probability that is equal to (1-p). Therefore, it is a question of a heterogeneous system with a non-cooperative strategy. We do not forget. The prob-cache is on the way, heterogeneous, and the cooperative probabilistic caching mechanism. It augments the fields at each request and content of Time Since Inception (TSI) and Time Since Birth (TSB) to allow cached content to be available to neighbor caches, to enhance cooperation [44]. Al and Jason Min in the [45] literature, have suggested clear cache cooperation called the intra-domain cache in cooperation. To be more specific, it takes care of two cache

summary tables that save the content information currently cached by the content router and replace it with other caches. In-network edge caching, classic web caching has been well used. As a result, many researchers have proposed new edge placement strategies, for example, in [46]. They show the edge buffer as a caching and reloading strategy, leading to the insufficiency of strategies that rely on the historical past of the device. Unlike the other case, they instead suggest a prediction model based on statistical aggregate data on the network. Instead of these blind popularity decisions, the researchers and authors of others suggest a mobility-focused probabilistic placement scheme (MAP) [47]. This mechanism caches content at the edge of networks with connected vehicles while considering the time required and trajectory predictions to finally serve the content. Authors S. Zhang et al. in [48] investigated maximum delay cooperation for large-scale edge caching, with high cluster size optimization and content placement based on stochastic information of network topology, traffic sharing, file popularity, and channel quality. CCPNC is a caching strategy that considers content popularity and distribution means of routing nodes. It caches the most requested content to the parent routing node with the centrality of the largest node and its next-hop node on the return path of the content, and the least requested content is cached in the non-routing route. main with the nodes defined by the coefficient cache probability link to the query length [49].

As explained above, the replace phase consists of deciding what content to eliminate from the cache. This ranks by, frequency-based on Least Recently Used (LRU) [50] and Least Frequently Used (LFU) [51], and semantically as FIFO (First In First Out) the FIFO replaces the oldest contents of the cache [52]. Not long ago, an alternative to Age-only caching was explored in [53]. To not feel the replacement of the cache, a penalty caused by the deletion of old contents is suggested in Max-Gain In-Network Caching MAGIC [54]. It also helps to maximize the local cache gain of new cached content. In [55] the Cache Less for More strategy further aims to cache in the best place in the download routing using the intermediate centrality mechanism [56].

Some research provides an architecture for the best selection of caching strategies [57]. The general idea is to switch between replacement strategies that already exist based on the change in user behavior with the help of an intelligent agent (ICA). Authors Matha Deghel and Al have worked on the benefits of caching at the edge of the multiple-input, multiple-output (MIMO) interference channel [58]. In [59] authors Al and Qinghua Ding presented a collaborative Edge Caching system and designed a Sequential Auction Mechanism (SAM) to allocate cache space per sequence. For Data Networking Caching, in [60] Noor Abani et al.

considered in their research a Locate vehicle network on the NDN where they displayed a proactive caching based on mobility prediction using the Markov model to remove insert cache redundancy by caching edges. In the [61], the authors propose a work based on comparing the activities of the caching at the edge compared to the traditional caching on the way on a geometric topology and choose the best strategy. There have been works that have had to put together cloud processing and caching capabilities mentioned in the following literature [62] [63], [64] [65]. The Literature [62], proposes new caching and data processing environment in the cloud through the information-theoretical model for (Fog Radio Access Network) F-RAN which is a wireless network assembly developed which uses the caching capabilities in the nodes of the wireless peripheral network. In [64], and [65], the authors also propose another cooperative hierarchical caching environment in a Cloud Radio Access Network (C-RAN) system. In this context, a new cloud-cache a Cloud Processing Unit (CPU) is reserved for addressing the lack of storage capacity/performance lag in the core and edge caching paradigms. However, the previous caching systems have been developed with great success in classic web caching, on the mobile side, on-board caching is still lacking due to the negligence of certain properties and the being focused only on popularity while forgetting the characteristics of the end-user and environments such as the uncertainty of the topology of the mobile network, the mobility of the subjects, the cost limit, and the storage.

3.2.3 Comparisons of Caching Strategies

The comparison is made on four different caching parameters in several simulation environments. The most popular caching strategies parameters are popularity model, topology, cache size, and catalog.

(1) Popularity Model: Popularity is a statistic that shows end-user requests and content value, popularity changes over time [66]. A content popularity model is defined by a function that outputs the popularity of different content, i.e., how many times different users request the same content. Documents [67] and [68] show us how content popularity is modeled by the probability of a distribution function like Zipf or MZipf.
(2) Topology: several topologies were used to evaluate caching strategies. At the level of Internet service providers, topologies vary from K-aire trees.
(3) The size of the cache: the size of a cache makes it possible to know the space available in the various MEC servers to store web files temporarily. Generally, the size is determined with a given value or a radio compared to the size of the catalog. In [69], studies were done on the influence of multiple object sizes web

the performance and overhead of web caching and show how size awareness could serve caching efficiency.

(4) Catalog: the catalog is the part that represents all the content of the network. The number of requests for some content varies directly from the popularity. A variety of values for the size of a catalog are used, for example, there is YouTube which uses catalogs of more than 100 contents.

3.2.4 Differentiated Caching Services

In the literature of Lu [70], we can find the different importance of getting a variety of caching services, as the importance we can mention:

(1) At the content level: the caching of the most requested content is improved for the end customer's satisfaction. The web contents are important to define the satisfaction of a customer suddenly. The caching mechanisms are obliged to treat with particular attention the contents and need a specification between several ranges to optimize the QoE of the final subjects.

(2) At the user level: the cache is not to be neglected for mobile users since it promotes the improvement of the QoE quality of experience and rapid access to the network. In the case where the cache is located in the center of the network, there is a large improvement in the transmission delay. In the opposite case where the cache is at the edge of the peripheral network, it will also have a good reduction in the transmission delay. As a benefiting end, users will have fast access to the network.

3.3 Caching Network Optimization Models

There are many studies on caching network optimization. The literature [79] tas about how software-defined networking-enabled caching was investigated for the wireless network. The authors of [80] have worked on energy-efficient caching in a wireless ad hoc network to use to achieve the desired trade-off performance between energy use and latency. The cooperative caching solution was studied in [81] to maximize system performance and minimize expected delay. The authors of [82] propose a cluster-based caching scheme for divergent wireless networking systems that were investigated to improve performance.

3.4 Overview of Cache Replacement Algorithms

This chapter will go through the algorithms that will be subsequently used in the next chapter to compare with our approach, where applicable a “strength” equation is provided this is based on Anderson and sSchooler's [91] study that suggests a good estimator of need probability, Each item in memory is given a

"strength" value. the suggested strength function in their 1991 study on the probability of word occurrences in the New York Times headlines was:

$$S_{AS91} = A\sum_{I=1}^{n} s(t_i) \quad (1)$$

Where A is a constant, n is the number of occurrences of the item, t_i is the time of the ith occurrence, $s(t_i) = t_i^{-d_i}$ and $d_i = max[d_1, b(t_i - t_{i-1})^{-d_1}]$. being the decline of the strength from the ith occurrence over time d_i is a model parameter that can be changed. The model was able to reproduce the practice, recency, and spacing effects with d_1 = 0.125[17]. This is used to prioritize items in memory, here items are searched in decreasing order of strength and stop when strength falls below a certain threshold, because of recency effects imposed the strength of recalling the items in memory gradually drops, and hence this is akin to a cost function that determines the likelihood of finding an item in memory, where applicable this strength function will be provided below as we go through popular algorithms that we used to compare with our approach.

3.4.1 LRU

LRU [7] is one of the most popular algorithms. When the cache is full, the policy replaces the cache object that hasn't been accessed in the longest time. The strategy is based on the idea that recently referenced blocks are likely to be used again in the future.

LRU works effectively with workloads that have a high temporal locality. However, according to [20], LRU is ineffective with file server caches. LRU's time complexity is O (1).

A strength function that is compatible with LRU is:

$$S_{LRU} = \frac{1}{t_{current} - t_n} \quad (2)$$

Where $t_{current}$ is the current time and $t_n$is the time of the item's nth occurrence.

3.4.2 LFU

LFU is yet another popular cache replacement algorithm. Because LFU keeps track of all cache objects' reference counts, when the cache is full, it replaces the cache item with the lowest reference count. The algorithm's logic is based on the fact that some cache blocks are more frequently accessed than others. As a result, the frequency count provides an accurate assessment of the likelihood of a cache object being requested.

LFU has two major disadvantages. To begin, if two cache objects have the same frequency, there may be a tie. Second, even if a cache item is no longer active, it may acquire a big reference count and never be replaced. Many changes have been proposed to mitigate the disadvantage of LFU. The aged LFU is one of these enhanced varieties. This guideline assigns varying weights to modern and historical references. Aged LFU outperforms the initial LFU [8]. LFU has a temporal complexity of O(log(n)).

The strength function of LFU is:

$$S_{LFU} = n \tag{3}$$

where n is the number of occurrences of the item, The more times an item is used, the stronger it becomes.

3.4.3 FIFO

In terms of time complexity and implementation, this is one of the simplest replacement strategies. Cache objects are placed at the bottom of a FIFO queue. If a cache object needs to be replaced, it is removed from the head until there is enough capacity for the incoming request. This algorithm has a time complexity of O(1).

3.4.4 Combination of LFU and LRU (LRU-K & LFRU)

This algorithm is a hybrid of the LFU and LRU strategies. It originally appeared in database disk buffering. LRU-k's [10] primary idea is to keep track of the times of the last K references to popular cache items. On a request-by-request basis, this information is then used to statistically estimate the interarrival periods of references. The decision to replace is based on the reference density recorded during the previous K references. Cold cache objects are discovered faster when K is small because they have a larger period between the current time and the kth-to-last reference time. LRU-K has a temporal complexity of O(log(n)).

The strength function is equivalent to:

$$S_{LRU-k} = \frac{1}{t_{current} - t_{n-k+1}} \tag{4}$$

Where ti is the ith occurrence's time Because simple LRU does not compensate for frequency, LRU-k is a technique for adding frequency into an algorithm that is also sensitive to recency. As k grows, behavior approaches LFU, and as k approaches 1, it approaches LRU.

LFRU also combines LFU and LRU [9]. This algorithm's strategy is to replace cache objects that are infrequently and infrequently used. Each cached object has a Combined Recency and Frequency value assigned to it (CRF). The cache object

with the lowest CRF value is removed and replaced. Each cache object's request adds to its CRF. The time complexity of LRFU is between O(1) and O(log(n)).

the strength function is:

$$S_{LRFU} = \sum_{I=1}^{n} \frac{1}{2}^{\lambda(t_{current}-t_i)} \tag{5}$$

where λ is a variable parameter with the value λ = 0.001.

3.4.5 Optimal Algorithm (OPT)

Sometimes referred to as Belady's algorithm, OPT, or MIN, in this algorithm the item that will not be used for the longest time in the future is replaced by this algorithm. It entails anticipating future queries to determine which item in the cache will be required again. The Optimal algorithm produces the fewest page faults, but it is difficult to implement since it requires knowledge of future requests. [9 comp cache vid]

3.4.6 2Q & MQ

As an improvement to LRU-k, the 2Q algorithm [12] has been proposed. The goal is to reduce access overhead and promptly delete cold cache objects. The 2Q employs two LRU queues, A1out and Am, as well as an additional FIFO queue, A1in. When the cache objects are first accessed, they are originally stored in the A1in. When a cache is removed from A1in, it is added to A1out. When an A1out cache item is accessed, it is relocated to Am. For second-level buffer caches, the 2Q outperforms LRU and LFU [8]. 2Q has an O time complexity (1).

MQ [8] uses an approach similar to 2Q. The goal is to develop an algorithm that has a frequency-based priority and supports temporal frequency for cache objects. MQ employs multiple LRU queues (denoted as m), where m is a variable parameter. Cache items in specific queues have longer lifetimes than others, depending on where the object is in the queue. MQ additionally stores recently evicted cache objects in a history FIFO queue Qout of restricted size. MQ evicts the cache object in the LRU queue's tail with the least frequency. MQ [8] outperforms 2Q, LRU, and LFU. the time complexity is O(1).

3.4.7 ARC

ARC [11] is comparable to 2Q but can adapt to its settings. The maximum size of the LFU queue in ARC changes as data is received. The algorithm keeps track of the items removed from each queue. When ARC encounters a cache miss (i.e., a

failed retrieval) for an item that was just evicted from one of the queues, it expands that queue since it got rid of an item that it should have kept.

4 Our Approach, Simulation Results, and Analysis

4.1 Network Architecture

MEC intends to minimize core network traffic by putting computational and caching capabilities closer to users. Mobile operators and content providers would benefit from caching popular content in the MEC local cache to boost user QoE. Consider the generic system design depicted in Figure 4.1. In this case, a cluster of MECs defines a collaborative area to support the core network. Let C denote a collection of collaborative spaces $C = \{C_1, C_2, C_3 \ldots C_n\}$. Each collaborative area has a collection of MEC servers $C_i = \{M_1, M_2, M_3 \ldots M_n\}$. To offer computing and caching resources, it has been assumed that the MEC is co-located with the base station.

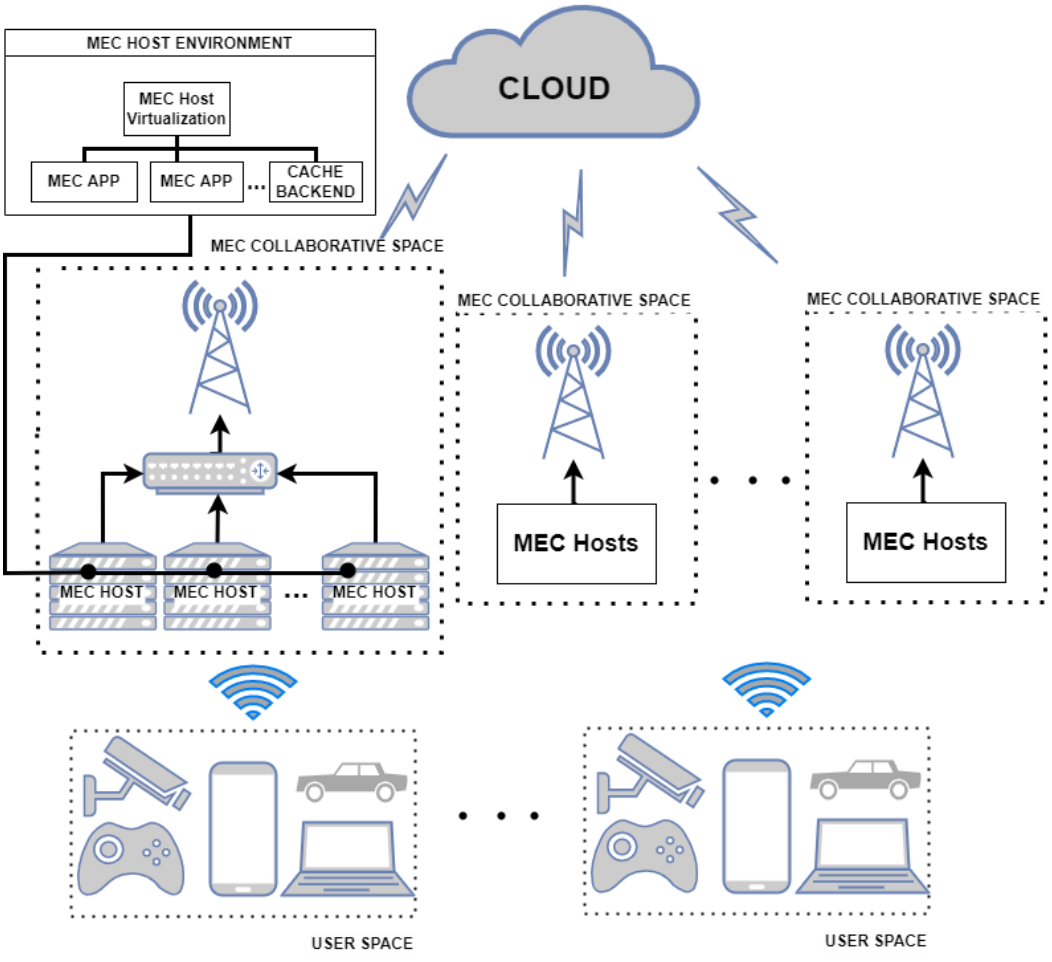

Figure 4.1: Overall Network Architecture Concept

The collaborative space is expected to be determined by the network administrator based on the hop count distance between the MEC nodes [27]. This is done to reduce communication delay and communication overhead within the collaborative environment. Users are linked to the MEC in the collaborative space

that is nearest to them. The MEC and the UE have a skewed one-to-many cardinality. In this case, an edge node is linked to several UE, but no UE is linked to many edge nodes. Let us define $U_i = \{u_1, u_2, u_3 \ldots u_n\}$. as a finite non-empty set of UEs connected to a MEC M_i. Each u_i request data r_i to be obtained via the MEC, where $r_i \in R$. $R = \{r_1, r_2, r_3 \ldots r_n\}$ is a finite non-empty set of data that can be accessed from C or P, where P is the cloud platform. Requests that cannot be fulfilled by C are routed to P via the core network.

4.2 Proposed Methodology

The issue of temporal frequency has been addressed here. In addition, an adaptive predictive caching method that learns user request patterns anticipates queries and prefetches them is proposed. Finally, a collaborative approach for effectively utilizing global MEC cache storage is developed. The details of the proposed plans are summarized below.

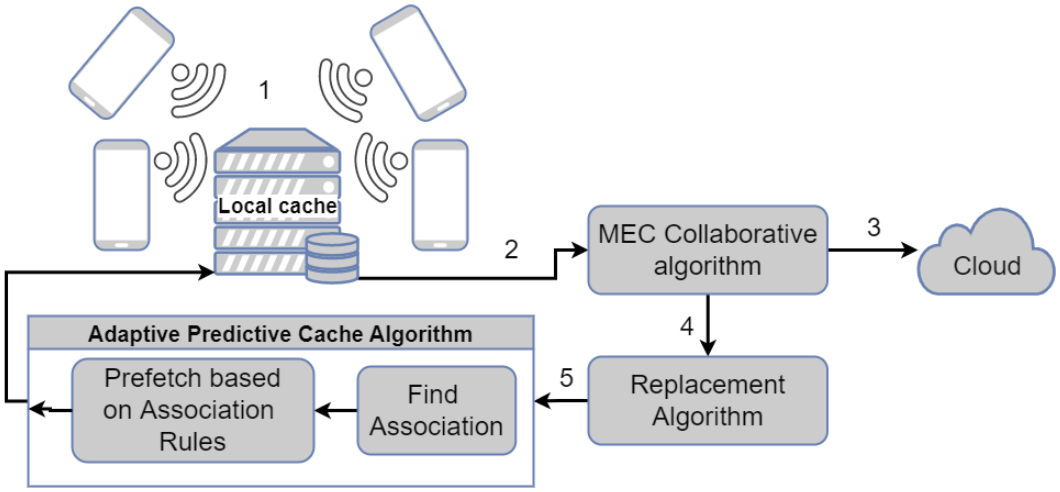

Figure 4.2: Caching System Diagram

Figure 4.2 depicts the suggested caching system. The algorithm is initialized with a fresh user request after the numbered items in the figure. If the requested cache object is not found in the local cache, the MEC Collaborative Algorithm is invoked to get it. If it is not already in the collaborative cache, it is obtained from the content server. When the cache is filled, the Replacement Scheme handles the identification and eviction of the cache victim. Finally, depending on the received request patterns, the Cache Prediction Scheme constructs cache sequential association rules. When there is a match based on the developed rules, the cache items are prefetched and placed in the local cache, which then Adaptively adjusts to new inputs based on the hit rate. These three systems are detailed in depth below.

4.2.1 Replacement Algorithm: Adaptive Selective Replacement

The reasoning behind the proposed content caching algorithm is to develop an efficient caching algorithm with competitive time complexity. The basis of ASR is in the combination of LRU and LFU algorithms, in which the least frequent and least recent cache object is replaced. This algorithm, however, has undergone two changes:

Selective Caching:

The issue of cold cache objects, which are requested once and not requested again for an extended period and hence are unsuitable for usage as a cache store, is addressed. To address this, a selective caching strategy is utilized, in which the requested cache object ri is only cached under one of two conditions. Because the first requirement is based on temporal locality, cache objects with a higher temporal locality are prioritized. A history queue was used to do this. The second criteria prioritizes cache objects with a higher cost of retrieval. This is done to lower the cost of user access. Equations 1 and 2 show how the two conditions are represented.

$$r_i^{(R-1)} \leq r_J^{(R-1)} \ \forall \ r_i \in H \tag{6}$$

$$r_i^c > r_J^c \tag{7}$$

H denotes the history queue, while $r_i^{(R-1)}$ and $r_J^{(R-1)}$ denote the previous recency of the new cache object r_i, as well as the least often and recently used object, respectively. r_i^c is the retrieval cost of r_i, while r_j^c is the retrieval cost of r_j.

$$r_i^c = \frac{size}{t_{rate}} \tag{8}$$

The size of r_i in bits is represented by size, and the transmission rate is represented by t_{rate}. The missed penalty, r_i^c, is assumed to be the delay in obtaining the cache object if there is a cache miss. The history queue H is a finite-size FIFO queue (of size h_{size}). It saves the information about recently removed blocks but not the cache data. When the MEC cache store M_i^{store} is full, the selective caching decision is activated.

$$|M_i^{store}| \geq cache_{size} \tag{9}$$

When either equation 1 or equation 2 is true, r_i is deleted from H and stored into M_i^{store}.

Temporal Frequency:

A frequency count limiting approach is utilized to handle the problem of temporal frequency. The highest distance between two consecutive cache objects

r_j and r_i an LRU queue is organized in this manner by its frequency count, which is limited by f_{max}. The increment of the frequency count of a cache object r_i^f is defined by the function $FreqCount(r_i^f)$ specified in the equation below, with r_i being the lead cache object in M_i^{store}.

$$FreqCount(r_i^f) = \begin{cases} r_i^f + 1, & r_i^f - r_j^f < f_{max} \\ r_i^f, & r_i^f - r_j^f \geq f_{max} \end{cases} \quad (10)$$

A similar technique utilized in FBR [11] is also used to prevent the problem of overflow associated with the implementation of frequency counts. The sum of all frequency counts, f_{sum}, is dynamically maintained in this manner. As a result, anytime the following situation occurs, every frequency count in M_i^{store} is lowered.

$$\frac{f_{sum}}{|M_i^{store}|} \geq A_{max} \quad (11)$$

A_{max} is a specified maximum value that is used as an algorithm parameter. In this case, a small A_{max} indicates frequent updates. According to [11], the frequency count of cache objects in H and M_i^{store} is lowered using the following equation.

$$\left\lceil \frac{r_i^f}{2} \right\rceil \quad \forall \ r_i \in \{M_i^{store}, H\} \quad (12)$$

Using this method, fsum would be between $|M_i^{store}| \times \frac{A_{max}}{2}$ and $|M_i^{store}| \times A_{max}$ in a steady state. It is worth noting that in this reduction, a count of one remains one, a count of two remains two, a count of three remains two, and so on. To achieve LRFU, the ASR algorithm employs several LRU queues Q, each of which contains cache items with the same frequency count.

$$Q = \{Q_1, Q_2, Q_3 \ldots Q_n\} \ s.t \ \forall \ r_i \in Q_I, r_i^f = i \quad (13)$$

There is a dynamically preserved reference to the LRU queue containing the cache objects with the lowest frequency Q_L. As a result, when a cache needs to be replaced, the cache object in Q_L's the tail is the victim.

Algorithm I: ASR

Input $cache_{size}$, A_{max}, f_{max}, h_{size}
Output *None* **Initialization** $M^{store} \leftarrow \varnothing$

$cache_{decision} \leftarrow True$
// Procedure to be invoked upon reference to cache object r_i
1: **if** $r \in M_i^{store}$ **then**
2: $Q_i.pop(r_i)$
3: **else**

```
    D1 ← do eq. (1)
    D2 ← do eq. (2)
    if D1 or D2 is True then
        Victim ← QL.pop()
        H.push(Victim)
        if ri not in H then
            H.push(ri)
            cache_decision ← False
        else
            update r^recency
            cache_decision ← False
        end if
    end if
end if
k ← FreqCount(r_i^f)
if cache_decision then
    if Qk not in Mstore then
        Qk ← ∅
    end if
    Qk.push(ri)
end if
if eq(6) is True then
    do eq(7)
end if
```

4.2.2 Cache Prediction: Adaptive Prefetch Caching Algorithm

If the clients' future request pattern is known, the cache store can be maintained more efficiently, since OPT outperforms other algorithms[11]. Studying history is the best approach to predicting the future. This adaptive caching technique is motivated by the desire to offer a predictive caching strategy based on understanding the association patterns between content requests in a MEC environment where content popularity is time-varying and uncertain. To detect content requests with close relationships, association rule mining approaches are used. In this scenario, for a given MEC M_i and time t_n, request history Req is used to produce rules ($Rules$) that map antecedents $rule^{ant}$ to consequents $rule^{cons}$. Rather than providing forecasts for t_{n+1}, this strategy is used. It is expected that users frequently request particular content requests con_i and con_j simultaneously or sequentially, where con_i and con_j are sets of various lengths. As a result, the goal is to categorize coni and conj as being either $rule^{ant}$ or $rule^{cons}$, so that

$$con_i \rightarrow \text{conj} \neq \text{conj} \rightarrow con_i \tag{14}$$

Let S denote the FIFO request sequence with length m received by a MEC M_i.

$$S = \{r_1, r_2, r_3 \ldots r_k\} \tag{15}$$

This is an association sequential pattern mining problem. The order of the requests is preserved here, but no duplicate items may appear in the sequence. Two pruning strategies are used to minimize the algorithm's processing time. First, the dimension of the mined request sequence si is limited by k.

$$s_i = \{r_1, r_2, r_3 \dots r_k\} \;\; s.t \;\; s_i \subseteq S \tag{16}$$

The number of distinct elements u_{no} for a particular window size w_{size} is dynamically calculated, so that the sequence is restricted by the window size s_{ws} is a subset of S, akin to how ARC uses its ghost lists to adapt to new changes[12]. As a result, the following equation yields k.

$$k = (u_{no})^2 \tag{17}$$

This is to ensure that there is enough training data to create useful insights. [15] has taken a similar strategy. The training dataset D in this case is a matrix derived from S with a dimension of $u_{no} \times u_{no}$. As a result, the following requirement must be met for association mining to be done.

$$k \leq m \tag{18}$$

Second, the minimum level of support The $support_{min}$ a function is used to decrease the number of item sets that must be considered as candidates during the association mining process. The number of rows n_{row} in D that include con_i is the support of a given sequence set $con_i \;\; s.t \;\; con_i \subset D$.

$$support(con_i) = \frac{n_{row}}{|D|} \tag{19}$$

With $support_{min}$, a set Scon containing all coni and satisfying $support(con_i) \geq support_{min}$ is sorted after.

$$S_{con} = \{con_1, con_2, con_3 \dots con_n\} \;\; \forall \;\; support(con_i) \geq support_{min} \tag{20}$$

S_{con} is used to construct $Rules_{all}$, which comprises rules with antecedents that are a subset of S_{con}.

$$Rules_{all} = \{rule_1, rule_2, rule_3 \dots rule_n\} \;\; \forall \;\; rule_i^{ant} \subseteq S_{con} \tag{21}$$

The created rules $Rules_{all}$ are then ordered to determine which rules are the fittest by utilizing rule support ($Rule_{support}$) and rule confidence ($Rule_{confidence}$) [40]. $Rule_{confidence}$ of ($con_i \rightarrow con_j$) refers to the percentage of transactions in D that include both con_i and con_j.

$$Rule_{confidence}(con_i \rightarrow con_j) = \frac{Support(con_i \cup con_j)}{Support(con_i)} \tag{22}$$

Given the sorted ranked rules $Rule_{ranked}$, the top p rules are selected to be used for prefetch caching.

$$Rule_{output} = \{rule_1, rule_2, rule_3 \dots rule_p\} \ \forall \ rule_i \in Rule_{ranked} \quad (23)$$

Every $Rule_{timeout}$, $Rule_{output}$ is modified to ensure that only the most relevant rules are preserved. After each user request is completed, the $rule^{cons}$ of the $rule^{ant}$ that matches $Req((t-l) \rightarrow t)$ are prefetched. If no match is found, no prefetch is performed. Here, t represents the current position in Req, and l represents the number of elements in $rule^{ant}$. The rules in $Rule_{output}$ are stored in a hash table with the key being the number of elements in $rule^{ant}$ to enable speedy search. The value is also a hash table, with the $rule^{ant}$ as the key and the $rule^{cons}$ as the value. $Rule_{output}^{max}$ limits the number of rules in the $Rule_{output}$. This is the maximum number of $rule^{ant}$ items that can be saved in $Rule_{output}$. As a result, $Rule_{output}^{max}$ is the maximum look-up performed to discover matches?

Apriori [41] and FP-Growth [42] are two key algorithms that can be utilized for association rule mining. Apriori produces better associations with sparse datasets, but it uses a lot of memory due to its breadth-first strategy. FP-Growth is faster and employs a depth-first strategy. FP-Growth, on the other hand, does not work well with sparse datasets [43]. By using both approaches and then deciding which one to utilize at runtime depending on the sparse density and a given memory threshold mem_{max}, the best of both worlds has been blended. The default algorithm in this technique is FP-Growth. However, the average amount of memory used mem_{avg} for updating $Rule_{output}$ is saved. As a result, Apriori is employed if the following equation is true: Mmemi is the MEC's current memory utilization, and $D^{density}$ is the sparse density of the dataset D.

$$Apriori \mid M_i^{mem} < mem_{max} \ \forall \ D^{density} > 0.5 \quad (24)$$

Algorithm II: APCA

Input: *S, w_{size}, m, $support_{min}$, mem_{max}, $Rule_{output}{}^{max}$, p*
Output: *$Rule_{output}$*
Initialization:
$Rule_{output}$ ← ∅
From *s_ws obtain u_{no}*
k ← $(u_{no})^2$
if *k* ≤ *m* **then**
    $S_i = \{r_1, r_2, r_3 \dots r_k\} \ s.t \ S_i \subseteq S$
    obtain D from s_i s.t dimension = $u_no \times u_no$
    using eq.(15) obtain S_con
    if eq.(19) is *false* **then**
        $Rule_{all}$ ← *FPGrowth*(*D*)
    else
        $Rule_{all}$ ← *Apriori*(*D*)
    end if
    $Rule_{ranked}$ ← *sort*(*$Rule_{all}$*)
    $Rule_{output}$ ← *slice*(*$Rule_{ranked}$* , *p*)
end if

4.2.3 MEC Collaborative algorithm: Greedy Collaboration

This algorithm aims to efficiently manage the cache in the collaborative space by eliminating data redundancy and improving cache data exchange among MECs. The goal here is to improve the efficiency of the individual edge node to boost the efficiency of the global collaborative cache. Assume content-centric networking is used to share cache data within the collaborative space. As a result, the contents are obtained using the specified identifier n_i. Furthermore, let us assume that each collaboration space C_i has a name resolution server (nrs). After each content retrieval, the MECs in C_i populate the contents in nrs. The contents of the nrs are recorded in a blockchain to maintain security and integrity. To increase efficiency, M_i stores $M_i^{names} | M_i^{names} \subset nrs$ locally. M_i^{names} is a FIFO queue with a finite capacity. For the sake of simplicity, assume that each MEC M_i has identical storage capacity and is part of a collaborative space C_i. C_i^{store} represents the total amount of content kept in the collaborative store. As a result, $M_i^{store} \subset C_i^{store}$. MECs send event-driven updates to the C_i^{store}, which populate it. A binary cache function $Cf(cache_{store}, n_i)$ is also defined, which determines whether a cache item is available in a cache store.

$$Cf(cache_{store}, n_i) \in \{0,1\} \tag{25}$$

According to equation (20), 1 indicates that n_i is cached in a specific $cache_{store}$ and 0 indicates that it is not. If a named material is not kept in either M_i^{store} or C_i^{store}, it is fetched from the cloud P's content provider. If n_i is in more than one MEC in C_i, it is fetched from the MEC with the shortest network latency. $M_{i \to j}^{delay}$ denotes the network delay between two MECs M_i and M_j. If M_i^{ni} is a set of MECs with n_i in the cache and M_d is the MEC with the shortest network delay. The collaborative cache retrieval function $Retrieve(n_i, M_d)$ is defined in equation (21).

$$Retrieve(n_i, M_d) \leftarrow \ \forall \ n_i \in M_i^{ni} \ \ s.t \ \left| M_i^{ni} \right| > 1 \tag{26}$$

The amount of MECs that can hold n_i is limited to w percent to minimize cache redundancies. As a result, equation(22) represents the total number of MECs (TNM) that can store n_i.

$$TNM = w \times |C_i| \tag{27}$$

Algorithm III: Greedy Collaborative Caching Algorithm

Input *w*
Output *None*
$M^{names} \leftarrow \emptyset$
$M^{store} \leftarrow \emptyset$
$C^{store} \leftarrow \emptyset$

$TNM \leftarrow w \times |C_i|$
// Procedure to be invoked upon reference to cache object r_i

if $r_i \in M^{store}$ *not True* **then**
  $n_i \leftarrow M_i^{names}(r_i)$
  $q = Cf(C^{store}, n_i)$
  if $q \in \{1\}$ **then**
    if $|M_i^{ni}| > 1$ **then**
      r_i *Retrive*(n_i, M_d)
    else
      r_i *Retrive*(n_i, M_j)
    end if
    if $|M_i^{ni}| < TNM$ **then**
      $M_i^{store}.push(r_i)$
    end if
  end if
else
  use Algorithm I to fetch r_i
  Send push request to nrs with r_i
  send cache update to C_i
end if
Use Algorithm II to prefetch cache

4.3 Experiment Results and Simulation Analysis

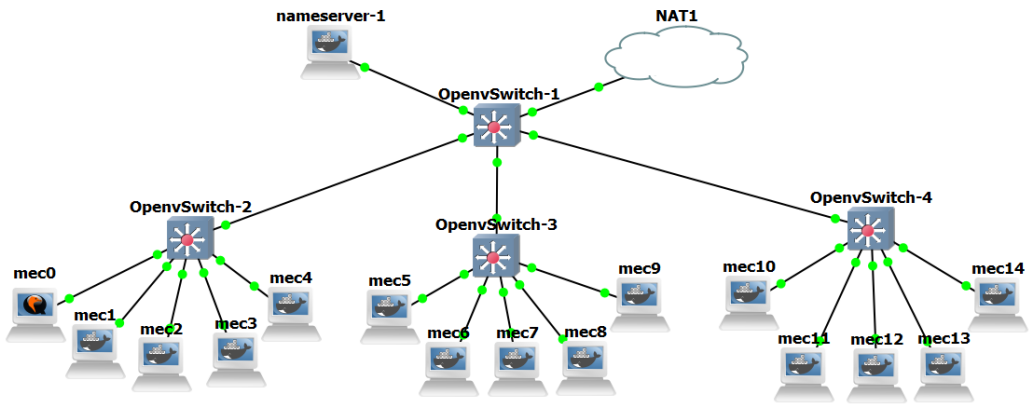

Figure 4.3: MEC Network & Infrastructure Setup in GNS3

To assess the efficacy of the suggested method, an emulation environment with varied numbers of MECs (5, 10, 15) and a content server was implemented. The content server was built using Netlify [44], while the MECs were built on a GNS3 platform. Each MEC is a Linux server with a docker virtualization infrastructure. Mosquitto (mqtt) is used as a messaging broker to communicate among the MECs. The proposed approach was compared against a number of the most frequently used algorithms. The MovieLens 20M datasets were used while running on MECs,

and data with varying Zipf distributions was used in the comparison. The emphasis was on movie IDs used in [35] and movie IDs ranging from 1 to 1070. Python was used to implement the algorithms. The experiment assumed that the arrival time of user requests on the MEC server follows the Poisson distribution $\lambda = 1$. TABLE 4.1 summarizes the parameters used in the experiment.

Figure 4.3 depicts the structure of the experiment's environment, the graphical network simulator 3 (GNS3) is a software widely used to simulate complex networks composed of real and virtualised devices, the namerver is a simple HTTP server that aims at simplify at providing a single point of fast communication between the MECs and the content server in the cloud it was implemented using Flask mainly due to it having minimal bloatware and speed, the switches are an open-source implementation of a distributed virtual multilayer switch, it was found that having more than 7 MECs on one switch incurred a large hike in latency, hence why 5 MECs where used on each switch, each MEC is a linux(Ubuntu 18) running as a docker container, all of them are similar being that they are minimal Ubuntu installs except 2, but even those run the same caching functions as the others, the first MEC(mec0) this has a GUI and is used to visually monitor the progress, run the control center and each MEC saves it's experimental data on it via FTP, the other is the second MEC(mec1) which hosts the MQTT server(mosquito), which the MECs use to communicate between themselves in order to check or update cache fields using hashes, update their adaptive rules and receive commands from the command MEC among other things, MQTT is s popular lightweight publish/subscribe messanging network protocol, the limit of 15 MECs was a hardware limit since the experiment was done on a 4 Ghz 16GB RAM windows computer with Hyper-V turned for running the GNS3 custom virtual machine (on VMare workstation pro), the MECs, their data and their virtualization software takes upto 80 GB, running the experiment took between 12 hours and 3 hours depending on the number of MECs used and the state of the WIFI network used, with a maximum 216 and 180 Mbps in download and upload speeds respectively, at the time of writing it reaches an average of 40 and 60 Mbps in download and upload speeds respectively, following are parameters used in the experiment with the various algorithms and environment variables used.

Table 4.1 parameters and variables used in the experiment

Parameters	*Value*	*Meaning*
General Setup		
$\|C_I\|$	(5,10,15)	MECs in the collaborative space
$\|R\|$	20000	Total requests have been done per experiment
No of content	1070	Unique content
λ	1	Poisson mean
Adaptive Selective Replacement Algorithm		
A_{max}	100	Max avg frequency
f_{max}	20	Frequency count bound
$cache_{size}$	50	Cache size
h_{size}	$4 \times cache_{size}$	History size
Adaptive Prefetch Caching Algorithm		
w_{size}	$\frac{cache_{size}}{2}$	Window size
m	${w_{size}}^2$	History request length S
$support_{min}$	0.45	Min support
mem_{max}	70%	Max ram threshold
$Rule_{output}^{max}$	4	Max length of $rule_{ant}$
p	10%	Top rules to select
Greedy Collaboration Algorithm		
w	10%	Redundant MECs that holds a cache
Comparison Parameters		
α	(1.01,1.20,1.35)	Zipf distribution parameter
$cache_{size}$	(10,20,30)	Cache size
$\|R\|$	1500	Total requests
No of content	40	Unique contents
MQ Algorithm		
$\|Q_{out}\|$	$4 \times cache_{size}$	History FIFO queues
m	8	Number of LRU queues

4.3.1 MECs Experiment Setup

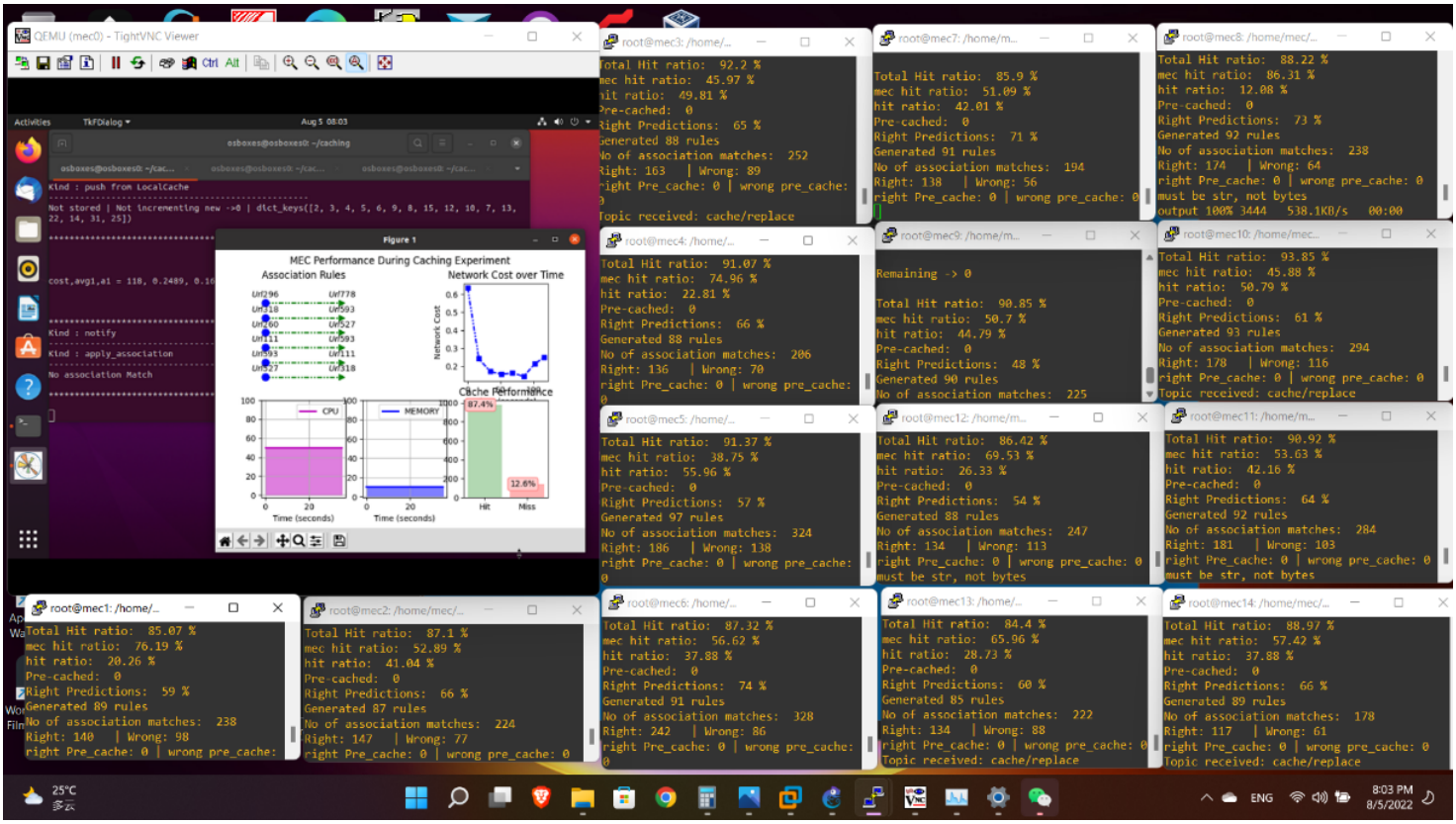

Figure 4.4: cache algorithms running on 15 MECs in virtual machines

Figure 4.4 shows when an experiment had finished, as mention in the section above here we see all the 15 MECs experiment summary, where one of the MEC's has a GUI that shows in realtime what is going on, it can also be seen that all the MECs have different result summaries, this is mainly attributed to different data they have to gather, some are larger than others, others might be uncommon and so it takes time to adapt and learn new association rules, it should be mention that all the MECs are self contained an error doesn't affect another except losing to their association rules and the cache hash made, also requests are divided the first time they run and so the controller MEC subdivides them and after that they are not tied to each other, this makes this solution good for places where internet and power cuts are a possibility, following will be a review of the results obtained from the 3 experiments conducted, which are simply running a different number of MECs on the same data (5,10,15 MECs).

4.3.2 Multiple MECs Experiment

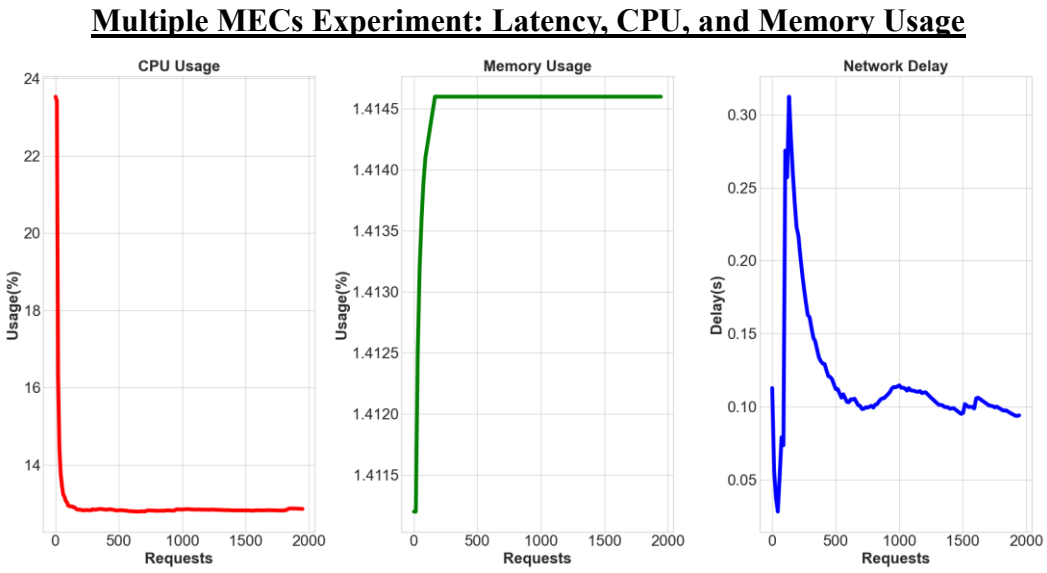

Figure 4.5: CPU, Memory and Latency

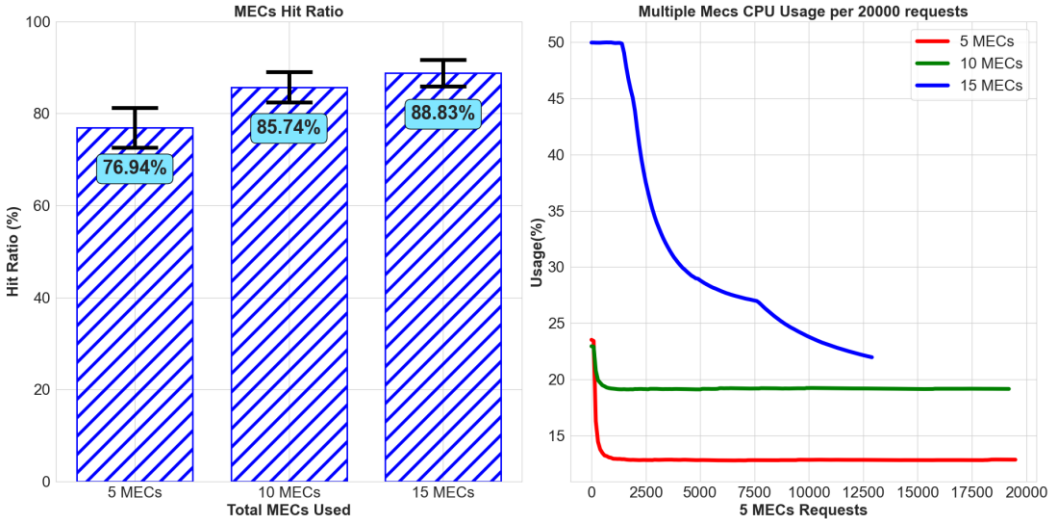

Figure 4.6: Perfomance comparison between 5,10 and 15 MECs

Figure 4.5 depicts the access delay. It can be deduced that our proposed Adaptive caching algorithm achieves a lower access delay relatively fast. This is related to an increase in the hit ratio, as access delay is affected by the hit ratio. When a user request is supplied from the local cache or MEC cache, there is less access delay than when the request is obtained from the content server in the cloud. As a result, a higher hit ratio would result in a shorter access delay. This reduction would result in improved QoE for end users and would bring us one step closer to realizing the envisaged 5G ultra-reliable low latency communications (URLLC).

Looking at Figure 4.5 it can be seen that our approach uses very little memory (<1.5%), although many of the parameters must be stored in memory, these parameters are capped to prevent overflow and hence achieve lower memory utilization. Due to the restricted computing resources of MEC nodes, low memory

consumption is critical in practical MEC systems. The experiment's memory use demonstrates that the suggested framework may be applied in a real MEC context.

Our adaptive algorithm approach has a relatively high CPU usage, Figure 4.5 shows that when they start their operation, it’s high but drops to an almost constant amount after a short time, in figure 4.6 we see that using many MECs increases the hit ratio and reduces the amount of time of doing the same number of requests, it can also be seen that the speed increase is not linear, we see a large increase in speed but also a larger increase in CPU usage from when we use 5 to 10 MECs than when 10 to 15 MECs were used, this high CPU utilization is because our predictor for what to prefetch is an online algorithm, which means that prediction functions keep running throughout the process as a result, the association prediction is done during runtime, resulting in a higher CPU consumption. The obtained CPU use is steady and predictable, as a result, they can be planned for during deployment.

4.3.3 Comparison of our Approach to Other Algorithms

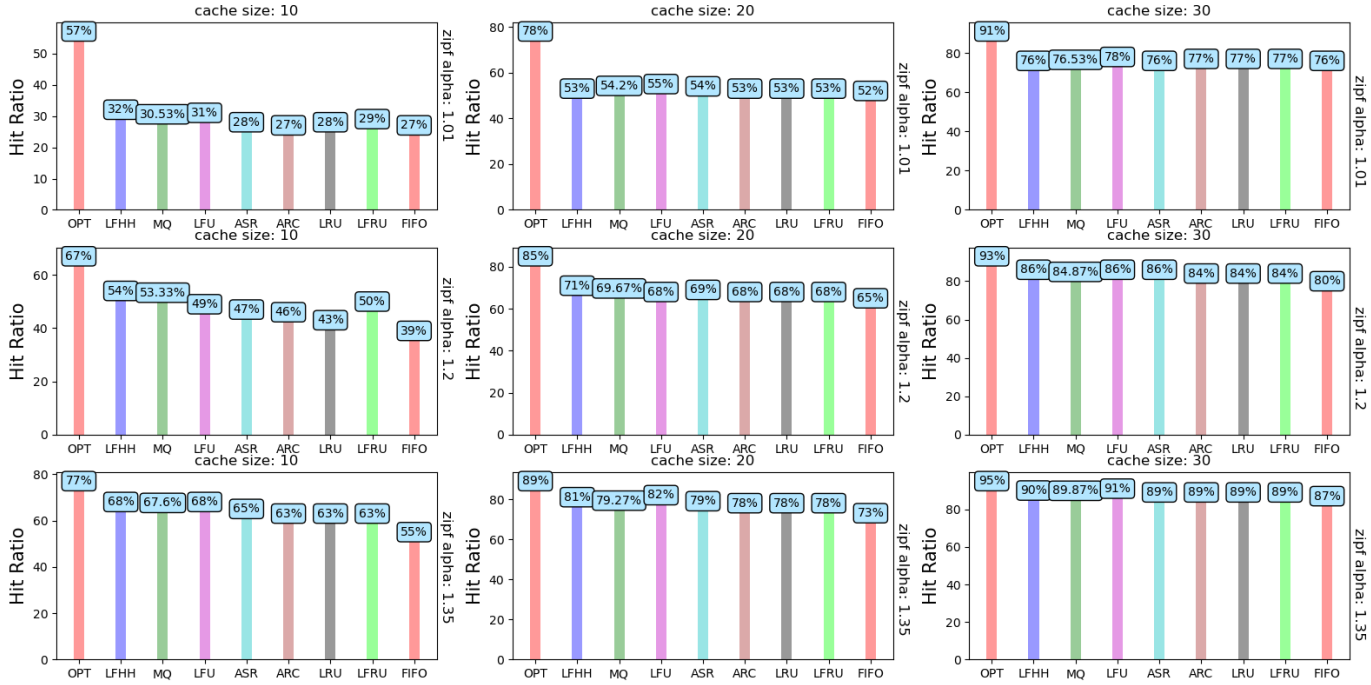

Figure 4.7: Comparison of ASR Perfomance with Contemporary Algorithms with Different Zipf and Cache Size Parameters

From Figure 4.7 our adaptive approach can be seen to outperform conventional algorithms and even outperforms ARC by 1% to 2% but surprisingly it doesn’t outperform LFU, even though it doesn’t outperform the more advanced algorithms it is not far behind and in Figure 4.8 it can be seen that it uses fewer resources that these more advanced algorithms, this experiment was not run on multiple MECs, here we are comparing the base caching system, which means we are comparing ASR with other algorithms, and so collaborative caching and prefetching based on

predicted association rules does not work in this instance, but still it perfoms quite well, it can also be seen that as the cache size and Zipf parameters go high, so does the hit ratio, this is because the algorithms can save more data and in the case of the Zipf parameter it increases the likelihood that more and more data is similar and so the data saved is even more effective.

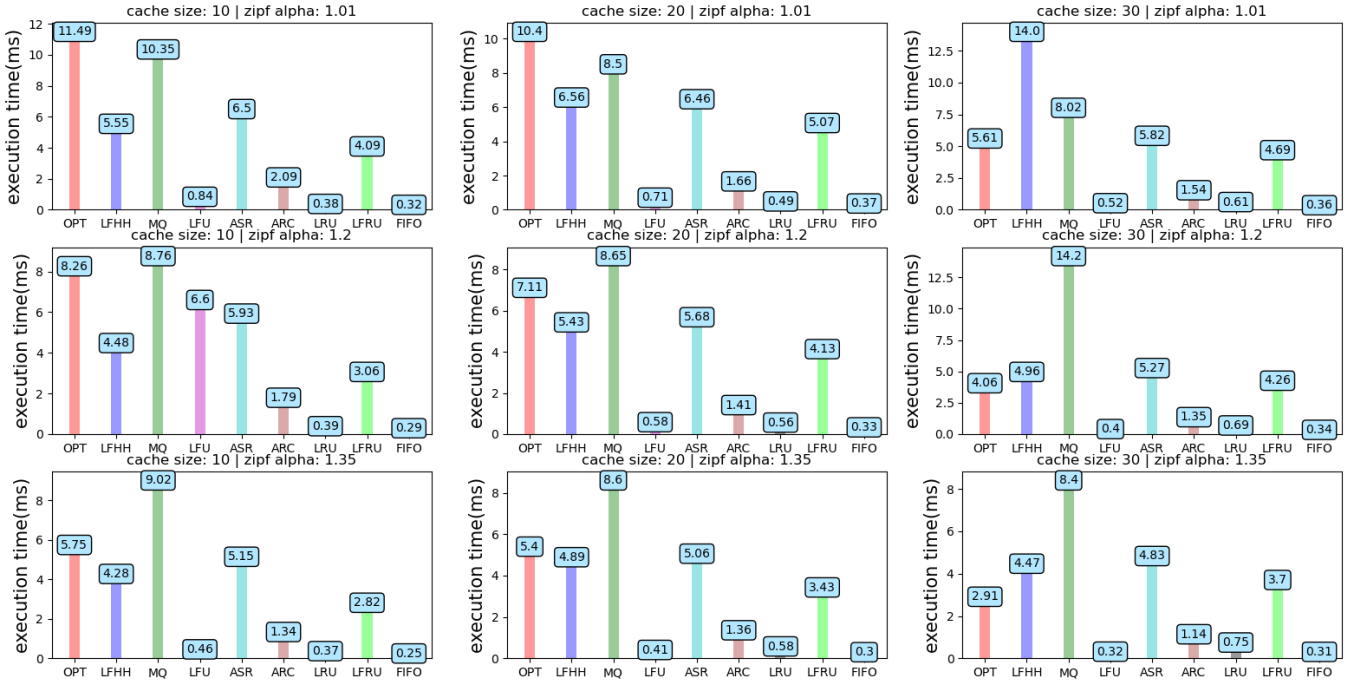

Figure 4.8: Comparison of ASR Efficiency with Contemporary Algorithms with Different Zipf and Cache Size Parameters

Figure 4.8 shows the time takes for each algorithm to perform a single request it can be seen that ASR is more efficient in caching than advanced algorithms like MQ but is less efficient than other more conventional algorithms including ARC when used on its own with no MECs that run the other solutions like our greedy collaborative caching and prefetching based on association rules, it can be quite inefficient but again it's in the middle of the more advanced algorithms and conventional ones, and so it is a middle ground between high performance with lower performance and aim to balance both when combined with the aforementioned solutions it's hit ratio goes higher.

5 Conclusion and future challenges

Nowadays, smartphones have become indispensable mobile terminals in everyone's daily life. In this paper, the adaptive cache strategy in mobile edge computing MEC is studied closely, which is of great significance to solve the current challenges faced by mobile networks. This study is based on user perception and performance of a mobile network, that is, multimedia services provided by the network. This study is divided into five chapters and sub-chapters. This chapter contains a summary of previous chapters and discusses future caching issues.

5.1 Conclusion

To cope with the transformation and upgrading of users' needs, the Internet quickly responded and updated again and again. From the current needs of users, users have become a new challenge to the development of good network functions and diversification. In this context, an always-available and confidential MEC system network was born. It completes the remote computing of mobile users through the mobile cloud, which is convenient and quick. The specificity of the mobile cloud is that it takes into account different sensors related to mobility, such as bandwidth, mobility, network connectivity, location, and context-sensitivity [119].

The first part of this thesis is the introduction. In this part, the author explains a bit about what MEC means and its importance in the world of mobile networks. From the start, the author presents the simplest definition of MEC: Mobile Edge Computing, which aims to bring users closer to their networks. On the other hand, the author briefly discusses the different caching strategies available for the MEC system and suggests that choosing the best strategy for the MEC system is better.

The second part describes the Edge Computing system still being standardized by ETSI (ISG). Edge Computing aims to optimize the use of cloud computing. Instead of sending the data emitted by devices connected to the cloud, it suffices to directly process this data at the edge of the network precisely at the same level where it is emitted. Moreover, this method is very successful because it reduces the bandwidth requirements of communications between the sensors and the data processing center by conducting analyses and knowledge at the data source level. In other words, the data is assumed automatically by the periphery which produces it or by a local server. Until today, Edge Computing has nothing more to prove. MEC (Mobile Edge Computing) is an application derived from the Edge Computing paradigm. It is a network system that offers cloud computing services

in computer services at the edge of a cellular network. The MEC system provides a computing environment and cloud equipment located at the edge of the network and in the heart of the radio access network. To provide a better user experience, MEC aims to reduce latency and ensure smooth operation and better service delivery. Going in the opposite direction of old-school mobile networks, the MEC proposes that the content is very close to the users. This is possible. Thanks to the content caching systems on the MEC servers, the volume of data transferred to the heart of the network decreases. Hence there is no longer a concern for congestion and the latency of the applications circulating on the MEC servers [120]. In other words, the current Internet must meet many challenges and therefore create problems that a user cannot bear. As an example of a problem, there is the sensitivity and security of the context. Major challenges such as security, mobility, caching, and quality of service have created new requirements. Users now focus more on the desired information; hence it is more important to refocus communication on access to content than on host-host interaction. For many researchers, the MEC system and content networks will be very useful in the universal growth of the Internet by treating content as the basis of the network architecture and storing content in several places close to the end-user.

The third part is devoted to caching management mechanisms and some available cache strategies. A cache is a place where duplicates of data are distributed that will be more easily found and quickly found by an application. The objective of the cache is to improve access to data without loss of time to gain display speed or to limit the abundance of sufficient data in memory. Caching is one of the most popular technologies and is especially considered a data recovery solution at the edges of the network. It was also mentioned as a technology that provides relevant opportunities for the execution of caching services. Specifically, MEC servers are located at the base stations, providing caching and allowing deployment onboard end users. This part introduces the general aspect of caching and mentions some caching strategies. It presents a brief description of the performance measures as well as the basic criteria that are at all times used to make the comparison of the caching mechanisms that already exist.

We continue describing the development of new communication technologies and improving the quality of experience. With the contribution of more intelligence in the decision-making at the time of the storage of content, generally speaking, and particularly in the MEC system, it is possible to offer better services in a mobile environment in full change. We present in this chapter some techniques used

successfully by some researchers: LRU, LFU, FIFO, LRU-K, LFRU, OPT, 2Q, MQ, and ARC.

LRU simply removes contents that have not been recently used, whereas LFU removes content that have are not frequently used, FIFO removes content that have been added at the earliest moment hence its name “First in First Out”, LRU-K is an attempt at giving LRU some content frequency features like LFU, it keeps references K of the most used cache items and they are replaced based on the size of K and so cold caches are discovered faster, LFRU is a combination of LRU and LFU, it keeps a recency and frequency value (CRF) for each content and the content with the least CRF is removed, OPT or the optimal algorithm also called Belady’s algorithm, this works by having some intuition on the data to be cache and so it removes contents that will not be used in the longest time in the future, 2Q is an improvement over LRU-K that aims at reducing access overheads and discovering and deleting cold caches quickly, it employs 2 LRU queues and a FIFO queue, MQ is a generalization of 2Q that has frequency based priority and supports temporal frequency for cache contents, this employs multiple LRU queues, contents have are kept stored depending on where they are in the queue, also recently evicted content is store in a FIFO history queue, MQ evict objects with the least frequency in the LRU queues, ARC was invented at IBM in 2004 it is comparable to 2Q, the main difference being that the LRU queues can change size depending on the data received, this is done if there is a cache miss in a recently evicted content in one of the queues, it then expands that queue since it was needed but removed, this is how it adapts.

Section 4 describes our approach, experiments and comparison with other algorithms, the author begins by explaining the MECs network structure where it’s composed by multiple base stations with each one connecting to 5 MECs, in the experimentation 5, 10, 15 MECs are run to monitor the latency and hit ratio improvements of such configurations, an individual MEC is a Linux virtualized Docker container containing the caching mechanisms, server applications and intra MEC communication systems (using MQTT), the set of MECs, base stations and a name server is defined as a collaborative space, we then summarize our proposed approach, this is composed of our proposed replacement algorithm, Adaptive Selective Replacement which combines LRU and LFU where the least frequent and least recent cache objects are replaced, we also show how we provide solutions to cold cache contents by selectively caching contents and also the problem of temporal frequency where we use LRU queues to organize contents based on their popularity frequency, after which we describe our cache prediction strategy “Adaptive Prefetch Caching Algorithm” here we go through how we create

association rules of contents with uncertain time varying popularity by keeping a request history and with that creating close relationship attributes that lead to the creation of a rules map that can predict how different contents are closely related, we later explain how the MECs collaborate using our Greedy Collaboration algorithm that is akin to a FIFO queue that efficiently sends and receives content and so eliminates data redundancy since it keeps a hash of the cached objects and sends content as soon as they are requested.

The second part of the fourth section describes the experiments carried out and their results, we begin by showing the network as implemented in GNS3, with base stations made with switches, a name server, and 15 MECs connected to 3 base stations/switches, we also describe in detail the environment and dataset and various variables employed in the different algorithms. After we show the results of our approach performance and resource utilization, where we look at the latency CPU usage, memory usage, impact on hit ratio, and time to make 20000 requests if a different amount of MECs is used, we end by comparing our replacement algorithm with other algorithms using different Zipf distribution parameters where we see a significant increase in performance and we even outperform ARC by 1% to 2%, in the algorithms we do not outperform in hit ratio, our approach is significantly more resource efficient than them, we observed a discrepancy with LFU where it has both high hit ratio and efficiency, this has been attributed to the Zipf distribution used but a further investigation is needed.

5.2 Future Challenges

6.2 Future challenges

In the 21st century, the Internet is bound to enter the era of big data, and the network edge, as an instant and valuable technology for storing popular content, can meet the needs of mobile users for convenience and speed, while alleviating the problem of overcrowding in the network center. Therefore, in the context of artificial intelligence, a MEC server with an internal cache can be combined with artificial intelligence to provide users with a better user experience. Not only that but the MEC has been updated to handle huge amounts of data, which is good for the growth of data traffic. In terms of network utilities, caching is special in that it speeds up data retrieval and also reduces traffic over the network. This is possible because the different requests requested by the user can be managed through nodes closer to the user. Therefore, caches are often considered a bottleneck in the evolution of MEC architectures in networks because of the limited storage capacity of caches available on the periphery. In general, its cache uses several types of placement and replacement strategies to allow outstanding and popular content to

be shared across the network. To meet the requirements of the network and users, the MEC system is designed. This is an essential part of choosing how to manage caching in the face of network challenges, and most importantly, making optimal decisions across the entire network. The biggest challenge is knowing the solution to the random-dynamic network problem while not forgetting to improve the optimal cache decision. We came up with a popular video caching solution in MEC networks, and it gave us good results. For a better future, we hope to find solutions to the following shortcomings:

(1). Insufficient input: much research only shows speed as the information base, while many other significant factors affect caching decision making

(2). Dynamic conditions: remarkably, the caching system and the dynamism of the environment are neglected they must be treated with care.

(3). Indicative Isolation: Many strategies that manage caches don't point out the hardness of the cache decision they make on the spot.

We would like to focus on designing new caching strategies based on a machine-learning system for future research. We will mainly focus more on the reinforcement learning system detailed in our thesis with its particularity of allowing agents to make decisions through different interactions with the environment.

References

1) K.-K.-T. Conditions., " In Encyclopedia of Operations Research and management science", 2013.
2) "Executive Briefing - Mobile Edge Computing (MEC) Initiative. Tech. rep.," "ETSI - European Telecommunications Standards Institute," 2018. [Online], [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/MEC%20Executive%20Brief%20v1%2028-09-14.pdf.
3) Yun Chao Hu, Milan Patel, Dario Sabella, Valerie Young. "ETSI White Paper No. 11 : Mobile Edge Computing - a key technology towards 5G," Technical report, Available: www.etsi.org, 2015.
4) Sarra Mehamel, Khaled Slimani, Samia Bouzefrane, Mehammed Daoui. "Energyefficient hardware caching decision using fuzzy logic in mobile edge computing," in Proceedings - 2018 IEEE 6th International Conference on Future Internet of Things and Cloud Workshops, W-FiCloud, 2018: 237-242.
5) Michael Till Beck, Martin Werner, Sebastian Feld, Thomas Schimper. "Mobile Edge Computing : A Taxonomy.," In 6th International Conference on Advances in Future Internet, (AFIN), 2014.
6) Arif Ahmed , Ejaz Ahmed. "A survey on mobile edge computing," In Proceedings of the 10th International Conference on Intelligent Systems and Control, ISCO 2016. Institute of Electrical and Electronics Engineers Inc, 2016.
7) G. Nan X. Jiang, J. Bi , Z. Li."A survey on Information-centric Networking: Rationales, designs and debates.," China Communications,12(7):1-12, 2015.
8) A. H. R. A. M. S. Suhaidi Hassan, "A Taxonomy of Information-Centric Networking Architectures based on Data Routing and Name Resolution Approaches".
9) Hassan Diab, Ali Kashani, Ahmad Nasri. "Cache replacement engine: A fuzzy logic approach," in In Proceedings of the 2009 International Conference on the Current Trends in Information Technology, CTIT 2009, 2009.
10) Abdelwahab Hamam ,Nicolas D. Georganas."A comparison of mamdani and sugeno fuzzy inference systems for evaluating the quality of experience of haptoaudio-visual applications," In HAVE 2008 - IEEE International Workshop on Haptic Audio Visual Environments and Games Proceedings, 2018.
11) B. D. Davison. "A Web Caching Primer," Technical Report 4, [Online].

Available: http://www.cs.rutgers.edu/~davison/.

12) E. Ba¸stug, M. Bennis , M. Debbah. "Cache-enabled small cell net-," in works: Modeling and tradeoffs, Barcelona," in 2018 11th International Symposium on Wireless Communications Systems (ISWCS), 2014, 649-653.
13) P. Blasco , D. Gündüz. "Learning-based optimization of cache content in a small cell base station," in 2017 IEEE International Conference on Communications (ICC), NSW, Sydney, 2017,1897- 1903.
14) N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch , G. Caire. "FemtoCaching: Wireless video content delivery through distributed caching helper," in in 2012 Proceedings IEEE INFOCOM, FL, Orlando, 2018, 1107-1115.
15) K. Poularakis, G. Iosifidis, V. Sourlas , L. Tassiulas. "Multicastaware caching for small cell networks," in 2018 IEEE Wireless Communications and Networking Conference (WCNC), Istanbul, 2017,2300-2305.
16) K. Poularakis, G. Iosifidis , L. Tassiulas. "Approximation caching and routing algorithms for massive mobile data delivery,"in 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, AG, 2013,3534-3539.
17) H. Ahlehagh , S. Dey."Video-Aware Scheduling and Caching in the Radio Access Network," IEEE/ACM Transactions on Networking,2018, 22(5):1444-1462.
18) B. Bai. "Caching based socially-aware D2D communications in wireless content delivery networks: a hypergraph framewor," 2018, 23(4):74-81.
19) B. Chen, C. Yang , G. Wang."Cooperative Device-to-Device Communications with Caching," in in 2018 IEEE 83rd Vehicular Technology Conference (VTC Spring, Nanjing, 2018, 1-5.
20) A. Ioannou , S. Weber."A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networking," IEEE Communications Surveys & Tutorials, 2017, 18(4):2847-2889.
21) N. Laoutaris. A closed-form method for LRU replacement under generalized power-law demand, 2017.
22) H. Ahlehagh , S. Dey."Video-Aware Scheduling and Caching in the Radio Access Network," IEEE/ACM Transactions on Networking, 2018, 22(5):1444-1462.
23) A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, T. C. Clancy. "in 2014 11th International Symposium on Wireless Communications Systems (ISWCS)," in Learning distributed caching strategies in small cell networks, Barcelona, 2018, 917-921.
24) J. Gu, W. Wang, A. Huang, H. Shan , Z. Zhang."Distributed cache

replacement for caching-enable base stations in cellular networks,," in 2018 IEEE International Conference on Communications (ICC), Sydney, NSW, , 2018, 2648-2653.

25) S. Borst, V. Gupta , A. Walid."Distributed Caching Algorithms for Content Distribution Networ," in 2017 Proceedings IEEE INFOCOM, San Diego, CA, 2017.

26) W. Jiang, G. Feng, S. Qin."Optimal Cooperative Content Caching and Delivery Policy for Heterogeneous Cellular Networks," IEEE Transactions on Mobile Computing, 2018.

27) R. Yu , al. "Enhancing software-defined RAN with collaborative caching and scalable video coding,"in 2018 IEEE International Conference on Communications (ICC), Kuala Lumpur,2016,1- 6.

28) S. Wang. "Distributed edge caching scheme considering the tradeoff between the diversity and redundancy of cached content," in 2018 IEEE/CIC International Conference on Communications in China (ICCC), Shenzhen.

29) B. Chen, C. Yang , G.Wang."Cooperative Device-to-Device Communications with Caching," in in 2018 IEEE 83rd Vehicular Technology Conference (VTC Spring), Nanjing, 2018,1-5.

30) P. Ostovari, A. Khreishah, J. Wu. "Cache content placement using triangular network coding," in in 2017 IEEE Wireless Communications and Networking Conference WCNC, Shanghai, 2013,1375-1380.

31) K. Poularakis, L. Tassiulas. "Exploiting user mobility for wireless content delivery," in in 2017 IEEE International Symposium on Information Theory, Istanbul, 2018, 1017-1021.

32) M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn , S. Moon. "Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems," IEEE/ACM Transactions on Networking, 2018,17(5): 1357-1370.

33) Yang, L. Cao, J. Liang, G. Han. "Cost aware service placement and load dispatching in mobile cloud systems," IEEE Trans. Comput, 2015, (65):1440-1452.

34) G. Paschos, E. Ba¸stug, I. Land, G. Caire , M. Debbah. " Wireless caching: technical misconceptions and business barriers," IEEE Communications Magazine, 2018, 54(8):16-22.

35) L. Breslau, Pei Cao, Li Fan, G. Phillips, S. Shenker. "Web caching and Zipf-like distributions: evidence and implications," in Proceedings of INFOCOM 99," in in Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, New York, 2016, 126-134.

36) S Traverso, al. "Temporal locality in today’s content caching: why it matters

and how to model it," ACM SIGCOMM, oct 2018, 43(5):5-12.

37) Ejder Bastug, Mehdi Bennis, Mérouane Debbah. "Living on the edge: The role of proactive caching in 5G wireless networks IEEE Communications Magazine," 2014.

38) Mathieu Leconte, Georgios Paschos, Lazaros Gkatzikis, Moez Draief, SpyridonVassilaras, Symeon Chouvardas."Placing dynamic content in caches with small population.," In Proceedings - IEEE INFOCOM, 2016.

39) Guido Urdaneta, Guillaume Pierre, Maarten van Steen. "Wikipedia workload analysis for decentralized hosting," in Computer Networks, 2009.

40) Meng Zhang, Hongbin Luo, Hongke Zhang."A survey of caching mechanisms in information-centric networking," in IEEE Communications Surveys and Tutorials, 2015.

41) Ioannis Psaras, Wei Koong Chai, George Pavlou."Probabilistic in-network caching for information-centric networks," 2012.

42) G. Rossini. "Caching performance of content centric networks under multi-path routing (and more)," Relatório técnico, Telecom ParisTech, 2011.

43) Kideok Cho, Munyoung Lee, Kunwoo Park, Ted Taekyoung Kwon, Yanghee Choi, Sangheon Pack. "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," In Proceedings - IEEE INFOCOM, 2012, 316–321.

44) W. K. C. G. P. Ioannis Psaras."Probabilistic in-network caching for information-centric networks," 2012.

45) Jason Min Wang, Jun Zhang, Brahim Bensaou. "Intra-AS cooperative caching for content-centric networks," 2013.

46) Feixiong Zhang, Chenren Xu, Yanyong Zhang, K. Ramakrishnan, Shreyasee Mukherjee, Roy Yates, Nguyen Thu."EdgeBuffer: Caching and prefetching content at the edge in the MobilityFirst future Internet architecture," in In Proceedings content at the edge in the Mobility first future internet architectur, 2015.

47) A. Mahmood, C. Casetti, C.F. Chiasserini, P. Giaccone, J. Härri. "Mobility-aware edge caching for connected cars," In 2016 12th Annual Conference on Wireless on Demand Network Systems and Services,” in WONS 2016, 2016.

48) Meng Zhang, Hongbin Luo, Hongke Zhang." A survey of caching mechanisms in information-centric networking," in EEE Communications Surveys and Tutorials, 2015.

49) Yunming Mo, Jinxing Bao, Shaobing Wang, Yaxiong Ma, Han Liang, Jiabao Huang, Ping Lu, Jincai Chen."CCPNC: A Cooperative Caching Strategy Based on Content Popularity and Node Centrality,"In 2019 IEEE

International Conference on Networking, Architecture and storage, NAS2019-Proceeding, 2019.

50) Stefan Podlipnig, Laszlo Böszörmenyi. "A survey of Web cache replacement strategies,",2004, 2:374–398.

51) Ibrahim Abdullahi, A Suki, M Arif, Suki Arif."Cache-skip approach for Information-CentricNetwork,"[Online].Available:URL https://www.researchgate.net/publication/284078557.

52) Areej M. Osman, Niemah I. Osman. " A Comparison of Cache Replacement Algorithms for Video Services," International Journal of Computer Science and Information Technology,2018, 2(5)95-111.

53) Mathieu Leconte, Georgios Paschos, Lazaros Gkatzikis, Moez Draief, Spyridon Vassilaras, Symeon Chouvardas."Placing dynamic content in caches with small population. In Proceedings - IEEE INFOCOM," 2016.

54) Dario Rossi, Giuseppe Rossini. "Caching performance of content centric networks under multi-path routing (and more," Relatório técnico, Telecom ParisTech, 2011.

55) Wei Koong Chai, Diliang He, Ioannis Psaras, George Pavlou. " Cache "less for more" in information-centric networks (extended version)," in Computer Communications, 2013.

56) M. Barthélemy. "Betweenness centrality in large complex network," In European Physical Journal B, 2004.

57) Harald Beck, Bruno Bierbaumer, Minh Dao-Tran, Thomas Eiter, Hermann Hel-lwagner, Konstantin Schekotihin. "Stream reasoning-based control of caching strategies in CCN routers," In IEEE International Conference on Communications, 2017.

58) Matha Deghel, Ejder Bastug, Mohamad Assaad, Merouane Debbah. "On the benefits of edge caching for MIMO interference alignment," In IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC, 2015.

59) Qinghua Ding, Haitian Pang, Lifeng Sun. "SAM: Cache space allocation in collab-orative edge-caching network,"In IEEE International Conference on Communications, 2017.

60) Noor Abani, Torsten Braun, Mario Gerla. "Proactive caching with mobility prediction under uncertainty in information-centric networks,"In ICN 2017 - Proceed-ings of the 4th ACM Conference on Information Centric Networking, 2017.

61) Ali Dabirmoghaddam, Maziar Mirzazad-Barijough, J. Garcia-Luna-Aceves. "Understanding optimal caching and opportunistic caching at "the edge" of

information-centric networks," In ICN 2014 - Proceedings of the 1st International Conference on Information centric networking, 2014.

62) Ravi Tandon and Osvaldo Simeone. "Cloud-aided wireless networks with edge caching: Fundamental latency trade-offs in fog Radio Access Networks," in In IEEE International Symposium on Information Theory - Proceedings, 2016.

63) Tuyen X. Tran, Dario Pompili. "Octopus: A Cooperative Hierarchical Caching Strategy for Cloud Radio Access Networks," In Proceedings - 2016 IEEE 13th In ternational Conference on Mobile Ad Hoc and Sensor Systems, MASS 2016, 2017.

64) Tuyen X. Tran, Abolfazl Hajisami, Dario Pompili. "Cooperative Hierarchical Caching in 5G Cloud Radio Access Networks," in IEEE Network, 2017.

65) [65] Tuyen X. Tran, Duc V. Le, Guosen Yue, Dario Pompili. " Cooperative Hierarchical Caching and Request Scheduling in a Cloud Radio Access Network," in IEEE Transactions on Mobile Computing, 2018.

66) Taofik Lamsub, Pichaya Tandayya. "A Dynamic Popularity Caching Policy for Dynamic Adaptive Streaming over HTTP," Ho Chi Minh City, Vietnam, 2019, 322–327.

67) Lada A Adamic, Bernardo A Huberman."Zipf's law and the Internet," in Glottometrics 3, 2002.

68) Dario Rossi, Giuseppe Rossini."Caching performance of content centric networks under multi-path routing (and more," in Relatório técnico, Telecom ParisTech, 2011.

69) Frank Hasslinger Oliver Hohlfeld Gerhard Hasslinger, Konstantinos Ntougias."Fast and Efficient Web Caching Methods Regarding the Size and Performance Measures per Data Object," in 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of communication Links and Networks (CAMAD) proceedings: 11-13 september 2019, Limassol, Cyprus, Limassol, Cyprus, 2019.

70) Ying Lu, Tarek F. Abdelzaher, Avneesh Saxena. "Design, implementation, and evaluation of differentiated caching services," in IEEE Transactions on Parallel and Distributed Systems, 2004.

71) Sedigheh Khajoueinejad, Mojtaba Sabeghi, Azam Sadeghzadeh. "A fuzzy cache replacement strategy for Named Data Networking based on Software Defined Networking," in Computers and Electrical Engineering, 2018.

72) Hiba Ali Nasir Sirour, Yahia Abdalla M. Hamad, Amir A. "Eisa An agent-based proxy cache cleanup model using fuzzy logi," In Proceedings - 2013 International Conference on Computer Electrical and Electronics

Engineering:’Research Makes a Difference ICCEEE 2013, 2013.

73) Nidhi Lal, Shishupal Kumar, Vijay Kumar Chaurasiya. "An adaptive neuro-fuzzy inference system-based caching scheme for content-centric networking,”, 2019, 6:4459-4470.

74) Alireza Sadeghi, Fatemeh Sheikholeslami, Georgios B. Giannakis. "Optimal Dynamic Proactive Caching Via Reinforcement Learning," In IEEE Workshop on Signal Processing Advances in Wireless Communications, 2018.

75) Xianzhe Xu, Meixia Tao. "Collaborative Multi-Agent Reinforcement Learning of Caching Optimization in Small-Cell Networks," In 2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings, 2018.

76) Wei Jiang, Gang Feng, Shuang Qin, Yijing Liu. "Multi-Agent Reinforcement Learning Based Cooperative Content Caching for Mobile Edge Networks," IEEE Access, 2019.

77) Jiayin Chen, Wenchao Xu, Nan Cheng, Huaqing Wu, Shan Zhang, Xuemin Sherman Shen. "Reinforcement Learning Policy for Adaptive Edge Caching in Heterogeneous Vehicular Networ,"In 2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings, 2018.

78) Wei Jiang, Gang Feng, Shuang Qin, Tak Shing Peter Yum, Guohong Cao . "Multi-Agent Reinforcement Learning for Efficient Content Caching in Mobile D2D Networks," in IEEE Transactions on Wireless Communications, 2019.

79) W. Li, E. Chan, D. Chen. "Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network," in: 2007 IEEE Wireless Communications and Networking Conference, IEEE, 2007, 3347–3352.

80) X. Li, X. Wang, S. Xiao, V.C. Leung. " Delay performance analysis of cooperative cell caching in future mobile networks," in: 2015 IEEE International Conference on Communications (ICC), IEEE, 2015, 5652–5657.

81) M.S. ElBamby, M. Bennis, W. Saad, M. Latva-Aho. " Content-aware user clustering and caching in wireless small cell networks," in: 2014 11th International Symposium on Wireless Communications Systems (ISWCS), IEEE, 2014, 945–949.

82) B. Sampathkumar, G. Yongkun. " Cloud computing simulator, uS Patent App. 14/983,765 (Jul. 6 2017).," 2017.

83) R.L. Mattson, al. "Evaluation Techniques for Storage Hierarchies," IBM Systems J,1970, 2(9).

84) L. Belady. "A Study of Replacement Algorithms for Virtual Storage Computers," IBM Systems J., 1966, 2(5):78-101.

85) F. Corbato. "A Paging Experiment with the Multics System," In Honor of P.M. Morse, MIT Press, 1969, 217-228.
86) W.R. Carr, J.L. Hennessy. "WSClock—A Simple and Effective Algorithm for Virtual Memory Management," in roc. 8th Symp. Operating System Principles, ACM press, 1981, 87-95.
87) J.E.G. Coffman, P.J. Denning. "Operating Systems Theory," Prentice Hall, 1973,282.
88) J.E.G. Coffman, P.J. "Denning, Operating Systems Theory," Prentice Hall, 1973.282
89) A.V. Aho, P.J. Denning, J.D. Ullman. "Principles of Optimal Page Replacement," J. ACM, 1971, 1(18):80-93.
90) E.J. O'Neil, P.E. O'Neil, G. Weikum."An Optimality Proof of the LRU-K Page Replacement Algorithm," J. ACM, 1999, 1(46):92-112.
91) [91] T. Johnson, D. Shash. "2Q: A Low Overhead High-Performance Buffer Management Replacement Algorithm," in Proc. VLDB Con., Morgan Kaufmann, 1994, 297-306.
92) S. Jiang, X. Zhang. "LIRS: An Efficient Low InterReference Recency Set Replacement Policy to Improve Buffer Cache Performance," in Proc. ACM Sigmetrics Conf, ACM Press, 2002, 31-42.
93) J.T. Robinson, M.V. Devarakond. "Data Cache Management Using Frequency-Based Replacement," in Proc. ACM SIGMETRICS Conf, ACM Press, 1990, 134-142.
94) D. Lee, al. "LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies," IEEE Trans. Computers, 2001, 12(50): 1352-1360.
95) Y. Zhou, J.F. Philbin. "The Multi-Queue Replacement Algorithm for Second-Level Buffer Caches," in Proc. Usenix Ann. Tech. Conf. (Usenix 2001), Usenix, 2001,91-104.
96) Le Phong Du, Tuan Anh Le, Nguyen Duc Thai, Phuong Luu Vo. "The performance of caching strategies in content centric networking," In International Conference on Information Networking, 2017.
97) Areej M. Osman, Niemah I. Osman."A Comparison of Cache Replacement Algorithms for Video Services," International Journal of Computer Science and Information Technology, 2018, 2(5):95-111.
98) Nikolaos Laoutaris, Hao Che, Ioannis Stavrakakis. "The LCD interconnection of LRU caches and its analysis," Performance Evaluation, 2006.
99) Giuseppe Rossini, Dario Rossi. "Coupling caching and forwarding: Benefits, analysis, and implementation," In ICN 2014 - Proceedings of the 1st

International Conference on Information-Centric Networking, 2014.

100) Ya ‘nan Ni Jia Shi Yuanjun He, Yi Zhu, Na Zhu. "A Cache Strategy in Content-centric Networks Based on Nodes Importance," Information Technology Journal, 2014, (13): 588–592.

101) Zheng Chen, Nikolaos Pappas, Marios Kountouris. "Probabilistic caching in wireless D2D networks: Cache hit optimal versus throughput optimal," in IEEE Communications Letter, 2017.

102) Andrea Vattani, Flavio Chierichetti, Keegan Lowenstein."Optimal probabilistic cache stampede prevention,"In Proceedings of the VLDB Endowment, 2015.

103) Jason Min Wang, Jun Zhang, Brahim Bensaou. "Intra-AS cooperative caching for content-centric networks," In ICN 2013 - Proceedings of the 3rd, 2013 ACM SIGCOMM Workshop on Information-Centric Networking, 2013.

104) Yan Zhang, Xu Zhou, Yinlong Liu, Bo Wang, Song Ci. "A novel cooperative caching algorithm for massive P2P caches," in Peer-to-Peer Networking and Applications, 2013.

105) "WEB CACHING AND RESPONSE TIME OPTIMIZATION BASED ON EVICTION METHOD," International Journal of Innovative Research in Science, Engineering and Technology, 2013.

106) Kideok Cho, Munyoung Lee, Kunwoo Park, Ted Taekyoung Kwon, Yanghee Choi, Sangheon Pack. "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in In Proceedings - IEEE INFOCOM, 2012, 316-321.

107) Ali Dabirmoghaddam, Maziar Mirzazad-Barijough, J. J. Garcia-Luna-Aceves. "Understanding optimal caching and opportunistic caching at "the edge" of information centric networks," in In ICN 2014 - Proceedings of the 1st International Conference on Information Centric Networking, 2014.

108) Arkaprava Basu, Mark D. Hill, Michael M. Swift. "Reducing memory reference energy with opportunistic virtual caching," in In Proceedings - International Symposium on Computer Architecture, 2012.

109) Yunming Mo, Jinxing Bao, Shaobing Wang, Yaxiong Ma, Han Liang, Jiabao Huang,Ping Lu, Jincai Chen. "CCPNC: A Cooperative Caching Strategy Based on Content Popularity and Node Centrality," In 2019 IEEE International Conference on Networking, Architecture and storage, NAS 2019-Proceeding, 2019.

110) Jason Min Wang, Brahim Bensaou."Improving content-centric networks performance with progressive, diversity-load driven caching,," In 2012 1st IEEE International Conference on Communications in China, ICCC 2012,

2012.

111) Muhamad Rizky Yanuar, Afwarman Manaf."Performance evaluation of progressive caching policy on NDN," in In Proceedings - 2017 International Conference on Advanced Informatics: Concepts, Theory and Applications, ICAICTA 2017, 2017.

112) Mikhail Badov, Anand Seetharam, Jim Kurose, Victor Firoiu, Soumendra Nanda. "Congestion-aware caching and search in information-centric networks," in In ICN 2014 -Proceedings of the 1st International Conference on Information-Centric Networking, 2014.

113) Jeffrey Erman, Alexandre Gerber, Mohammad T. Hajiaghayi, Dan Pei, Oliver Spatscheck. "Network-aware forward caching," In WWW'09 - Proceedings of the 18th International World Wide Web Conference, 2009.

114) B. J. Alghazo. Composer, Composer, SF-LRU cache replacement algorithm. [Sound Recording]. In Records of the IEEE International Workshop on Memory Technology, Design and Testing, 2004.

115) Anwar Kalghoum, Sonia Mettali Gammar, Leila Azouz Saidane. "Towards a novel cache replacement strategy for Named Data Networking based on Software Defined Networking," in Computers and Electrical Engineering, 2018, (66).

116) Le Jiang, Xinglin Zhang. "Cache Replacement Strategy with Limited-Service Capacity in Heterogeneous Networks," IEEE Access, 2020.

117) Svetlana Ostrovskaya, Oleg Surnin, Rasheed Hussain, Safdar Hussain Bouk, Jooyoung Lee, Narges Mehran, Syed Hassan Ahmed, Abderrahim Benslimane. " Towards Multi-metric Cache Replacement Policies in Vehicular Named Data Networks," in IEEE International Symposium on personal, indoor and Mobile Radio Communications, PIMRC,Volume 2018-september, 2018.

118) Shuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang, Wenbo Wang. , "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," IEEE Access, vol. 5, p. 6757–6779, 2017.

119) Weiwen Zhang. "Energy-optimal mobile cloud computing under stochastic wireless channel," IEEE Transactions on Wireless Communications, 2013.

120) Taofik Lamsub, Pichaya Tandayya. "A Dynamic Popularity Caching Policy for Dynamic Adaptive Streaming over HTTP," Ho Chi Minh City, Vietnam, 2019, 322–327.

121) W. Li, E. Chan, D. Chen. "Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network," in: 2007 IEEE Wireless Communications and Networking Conference, IEEE,2007, 3347-3352.

122) W. Li, E. Chan, D. Chen. "Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network," in: 2007 IEEE Wireless Communications and Networking Conference, IEEE, 2007,3347-3352.
123) X. Li, X. Wang, S. Xiao, V.C. Leung. " Delay performance analysis of cooperative cell caching in future mobile networks," in: 2015 IEEE International Conference on Communications (ICC), IEEE, 2015, 5652-5657.
124) M.S. ElBamby, M. Bennis, W. Saad, M. Latva-Aho."Content-aware user clustering and caching in wireless small cell networks," in: 2014 11th International Symposium on Wireless Communications Systems (ISWCS), IEEE, 2014, 945-949.
125) B. Sampathkumar, G. Yongkun. "Cloud computing simulator," uS Patent App. 14/ 983,765 (Jul. 6 2017).
126) R.L. Mattson et al. "Evaluation Techniques for Storage Hierarchies," IBM Systems J, 1970, 2(9):78-117.
127) D.D. Sleator and R.E. Tarjan. "Amortized Efficiency of List Update and Paging Rules," Comm. ACM, 1985, 2(28):202-208.
128) F. Corbato. "A Paging Experiment with the Multics System," In Honor of P.M. Morse, MIT Press, 1969, 217-228.
129) W.R. Carr, J.L. Henness. "WSClock—A Simple and Effective Algorithm for Virtual Memory Management," Proc. 8th Symp. Operating System Principles, ACM press, 1981, 87-95.

Appendix

MATLAB code for simulating the ARC algorithm.

```
Key= randi ([1 24], 1, 100) %100 requests
DB = 1:24; %24 videos in the database
%ARC
T1= [1,2,3] ;
T2 = [4, 5, 6] ;
B1 = [7, 8, 9] ;
B2= [1, 11, 12] ;
%storage space in T1
ARC_MAIN = [T1, T2] ;
ARC_GHOST= [B1, B2];
X =randi ([1 24], 1,50); %was 1000
N= length(X); %number iterations
%Counters
F1=0;
G1=0
G2=0
FF1= zeros (1, N);
GG1= zeros (1, N);
GG2 = Zeros (1, N);
```

作者简历及在学研究成果

一、 作者入学前简历

起止年月	学习或工作单位	备注
XXXX年XX月至XXXX年XX月	在XXXX学校XXXX专业攻读学士学位	
XXXX年XX月至XXXX年XX月	在XXXX单位从事XXXX岗位的工作	

独创性说明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得北京科技大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

签名：_____ 日期：_____

关于论文使用授权的说明

本人完全了解北京科技大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵循此规定）

签 名：_____ 导 师 签 名：_____ 日 期：_____

学位论文数据集

关键词*	**密级***	**中图分类号***	**UDC**	**论文资助**
学位授予单位名称*		**学位授予单位代码***	**学位类别***	**学位级别***
北京科技大学		10008		
论文题名*		**并列题名**		**论文语种***
作者姓名*			**学号***	
培养单位名称*		**培养单位代码***	**培养单位地址**	**邮编**
北京科技大学		10008	北京市海淀区学院路 30 号	100083
学科专业*		**研究方向***	**学制***	**学位授予年***
论文提交日期*				
导师姓名*			**职称***	
评阅人	**答辩委员会主席***		**答辩委员会成员**	
电子版论文提交格式 文本（ ） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） **推荐格式：**application/msword；application/pdf				
电子版论文出版（发布）者		**电子版论文出版（发布）地**		**权限声明**
论文总页数*				
共 33 项，其中带*为必填数据，为 22 项。				