Co-Operative and Hybrid Replacement Caching for Multi-Access Mobile Edge Computing

Emeka E. Ugwuanyi*, Saptarshi Ghosh*, Muddesar Iqbal*, Tasos Dagiuklas*, Shahid Mumtaz#, and Anwer Al-Dulaimi+
*Division of Computer Science and Informatics, London South Bank University, London SE1 0AA, UK
#Instituto de Telecomunicações Universidade de Aveiro, Portugal
+Center of Excellence, EXFO, Montreal, Canada

Abstract— **Multi-Access Mobile Edge Computing (MEC) is proclaimed as a key technology for reducing service processing delays in 5G networks. Caching on MEC will decrease service latency and improve data access by allowing direct content delivery through the edge without fetching content from the remote server. Caching on MEC is also deemed as an effective approach guarantying more reachability due to proximity to end-users. This paper proposes a novel hybrid content caching replacement algorithm in MEC to increase its caching efficiency where future request references are predicted using a polynomial fit algorithm along with Lagrange interpolation. Additionally, a distributed co-operative caching algorithm to improve data access within MECs. Experimental results have shown that the proposed scheme obtains more cache hits and lesser average CPU utilization due to its selective caching approach when compared with existing traditional cache replacement algorithms.**

Keywords—Multi-access Mobile Edge Computing, selective caching, OPR algorithm, co-operative caching.

I. Introduction

Multi-Access Mobile Edge Computing (MEC) is an emerging paradigm that provides computing, storage and networking resources within the edge of mobile Radio Access Network (RAN) [1]. The idea is to design mini servers known as edge nodes that can handle storage and computation resources initiated by mobile devices. These edge nodes are in close proximity to the end users providing a platform for caching and offloading with the aim of reducing bandwidth consumption and latency of the network [2].

Caching is a method of keeping the most used data in the cache memory to reduce latency and increase user QoE (Quality of Experience). The universally accepted metric of measuring the cache performance is the cache hit ratio. This is the total cache hits obtained by the system at a given point in time. Therefore, an increase in the total cache hit ratio means better cache performance of the system.

The aim of this paper is to enhance the cache performance of the MEC. To accomplish this, a caching algorithm has been designed and developed that supports cache sharing among MEC nodes, reduce latency and bandwidth and increase network robustness and reliability so that content is constantly available even if the server where the point of presence is located is down.

The proposed scheme uses co-operative caching to share cache data among the MEC nodes within the same cluster. Additionally, a hybrid scheme of the modified version of two existing replacement schemes, namely Optimal Page Replacement algorithm (OPR) [3] and Least Frequently Used algorithm (LFU) [4] is applied to each MEC to improve its cache hit ratio. The OPR algorithm, developed by Belady, is a theoretically optimal algorithm that replaces the data in the cache that will occur farthest in the future with the newly obtained request [4]. The downside of Belady's algorithm is the need for a future reference string. Reference string is used to define the incoming page requests to be cached [5]. To obtain this, historical data are kept in the pre-cache for all the requests received by an MEC and its relative frequency. The historical data are then used to predict the future occurrences of data in the cache. The enhancement of the cache hit rate performance would reduce MEC cache access time [5]. The proposed algorithm depends on the successful prediction of future requests based on past references. Therefore, a polynomial regression algorithm is employed to enable the forecast of the reference string.

The remainder of this paper is structured as follows: in section II we have reviewed related work and listed our contribution. In section III we presented the system architecture and the proposed algorithm. In section IV we presented the system model. We describe the experimental setup, tested and discussed the results in section V. Finally, we concluded the article in section VI.

II. Related Work

A. State of the Art

Many Caching policies and algorithms have been proposed for content caching. The conventional caching policies include First in First Out (FIFO), Least Frequently Used (LFU), Least Recently Used (LRU) and Optimal Page Replacement (OPR). LRU algorithm chooses its replacement victim based on which cache reference that has been unused for the longest time while LFU chooses its victim based on which of the cache references have the least frequency [4]. Content replacement policies such as the LFU and LRU have been adopted in a large number of caching policies [6] in MEC as compared to OPR which has not been applied to MEC. OPR is considered as an optimal cache replacement algorithm but labelled impractical due to the inability to precisely know the user's future requests. However, there have been some caching algorithms developed that are based on the OPR such as "Back to the Future: Leveraging Belady's Algorithm for Improved Cache Replacement," [7]. The proposed cache replacement policy learns Belady's optimal solution (OPT) for past accesses to guide future replacement decisions.

Few researchers have proposed edge caching strategies for 5G that combines both computation caching and data caching. Markakis et al [8] proposed a proactive edge caching strategy that predicts and prefetch popular contents based on big data analysis. Sungwook K. [9] leveraged a holistic caching structure for caching in small base stations using game theory. The developed hybrid algorithm uses split caching where one part caches popular content for communication and other part caches computation offloading services. In [10] a Mix-Cooperative (MixCo) caching strategy was developed for MEC servers in a Fiber-Wireless (FiWi) access network to reduces latency and increase cache performance. Huang et al in [11] proposed a CMAC (Cooperative Multicast-Aware Caching) strategy to reduce the average latency of delivering content and Yang et al in [12] explored ways of avoiding pollution attack on cooperative MEC caching. Most of the existing studies of content caching in MEC focus on energy efficiency, mobility aware caching and cache allocation [13] [14] [15].

However, to the best of our knowledge, there is no existing research that considers the optimization of MEC local cache storage by using Belady's algorithm and predicting cache reference patterns using historic cache frequency. In this article, a hybrid cache replacement policy that supports co-operative caching for MEC platforms has been designed and developed. The hybrid algorithm merges a modified Belady's algorithm with a distributed cooperative caching algorithm.

B. *Contributions*

The contributions of this article can be summarized as follows:

- A modified LFU replacement algorithm which not only compares the frequency of the data in the cache but also compares the frequency of a newly obtained request
- A modified OPR algorithm which uses polynomial regression to predict future cache data using historical data obtained from the relative frequency of the cached data
- A cooperative caching strategy for sharing cache information among MECs
- A selective caching algorithm based on where the requested data are obtained.

III. SYSTEM ARCHITECTURE

Figure 1 shows system architecture in this study, where several MECs form a cluster to provide cached content to mobile users. In this architecture, mobile devices are connected to the edge node through a wireless Access Point (e.g. IEEE 802.11) and the edge node is connected to the cloud servers which is in turn connected to the Internet. The figure shows the request-response flow for a worst case scenario when a particular user's content request is not available in the MEC cluster.

The focus of this research is on the MEC layer and how to leverage its distributed architecture to increase its cache efficiency by generating more cache hits while reducing network latency. The aim of this research is to reduce response time by storing popular data at the cache servers of the edge nodes. This will reduce access to the data repositories since most of the data are stored locally. An algorithmic approach was used in this study to create a solution for the problems identified in the earlier section. This paper proposes a new caching algorithm that optimizes caching mechanism by modification and integration of three caching schemes. These caching mechanisms are listed as follows.

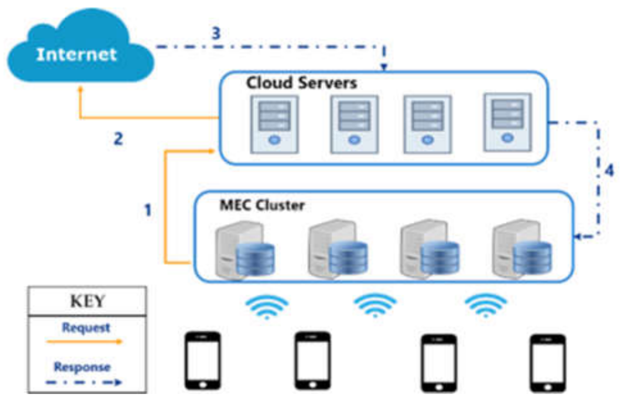

Figure 1 : System Architecture

A. *Co-operative caching*

Co-operative caching is a new way of caching in a distributed architecture that supports the sharing of cache information. One of the most popular co-operative caching methods is to create central storage for the distributed servers allowing them to access cache data from the central storage [16]. This reduces the burden of managing content only on one device. On the other hand, it introduces increased latency due to additional access time in fetching cached data from a remote location. Another problem would be where to locate the central storage device to minimize the access time of all the servers involved. In this work, distributed cooperative caching is used to reduce the impacts of one centralized cache server. The servers in the distributed architecture are allowed to keep their own cache data but they also keep a synchronized database of what data they have and what data are available in all other servers in a given cluster. We assume that the cluster size is adjusted based on the traffic load and network status to balance the content diversity and spectrum efficiency. The adjustment of the cluster is outside the scope of this paper.

If a MEC receives a request from an end-user device, it would first search if the request exists in its cache. If the request does not exist, it will search if any other MEC in its cluster has the requested data. If the requested data is still not obtained, then the MEC initiates a request to obtain the requested data from the point of presence.

B. *Modified Least Frequently Used (LFU) Cache replacement*

LFU cache replacement algorithm is a popular algorithm used to manage cache data [4]. It is a counter-based replacement algorithm. It keeps a counter of the number of requests that have been made to each reference. The algorithm requires that the reference with the least count is replaced because the actively used reference will be the one with the most count. To keep a cohesive and reliable frequency record, a window size is maintained. Since LFU only compares the frequency of the references in the cache and not the newly requested data, a problem arises when a new request with little or no historic frequency record is used to replace an existing cache data which has the least frequency in the cache but a higher historic frequency record when compared to the newly requested data. To address this problem, this paper proposes a method not only to compare the frequencies of the data in the cache but also the frequency of the newly obtained request. A decision is made not to cache the newly obtained request if the frequency of the data is less than the frequency of the least frequent data in the cache. This method is ideal for caching in MEC in a scenario where the newly requested data is obtained from the MEC cluster. Therefore, if a decision is made not to cache and the data is requested again, the data can be obtained from the MEC cluster as the link is less costly than the point of presence.

C. Modified Optimal Page Replacement (OPR) Algorithm

The Optimal Page Replacement Algorithm (Belady's Algorithm) has been considered as the optimal algorithm for cache replacement as it produces fewer page faults than any other algorithm [4]. Its working principle is to replace the reference which would not be used for the longest of time. The OPR algorithm provides the optimum result for any given cache reference and frame but it is labelled infeasible because it requires large amounts of data of future requests and time stamps of cache data.

In this research, a polynomial fit algorithm with Lagrange interpolation [17] has been leveraged to predict future cache occurrences using historic relative frequency of cache data.

IV. SYSTEM MODEL

Here we model the hierarchical content caching placement design. This consists of the cloud, the pre-cache and the MEC cache.

$$C_3 \subseteq C_2 \subseteq C_1 \tag{1}$$

Where C_1 represents the cloud data, C_2 represents the pre-cache and C_3 is the MEC local cache. The pre-cache contains a set of all the previously requested data and its relative frequency stored in the local MEC. Each content segment is uniquely identified by a hash function which is obtained from a sha256 hash of the request method, the URL and the request protocol.

$$C_2 = \{hash: freqency\}$$

$R = \{r_t | t \in [0, n-1]\}$ is a finite, non-empty reference string of reference instances r_t at time stamp t, with window size n. $C_2 = \{x_i \in R\} \subseteq R$ is a set of distinct content segments from the set R, with cardinality $|C_2| = m$. The function $f_r: x_i \in C_2 \to [0,1]$ calculates the relative frequency of occurrence of each reference x_i of pre-cache C_2 . it measures the share of occurrences of a reference instance over the total window. The function f_r is defined below.

$$f_r(x_i) = \begin{cases} 0 \;\; if\; x_i \notin R \\ \dfrac{n(x_i)}{|R|} \;\; otherwise \end{cases} \tag{2}$$

Hence the sum of relative frequency of all elements in the pre-cache is unity,

$$\sum_{i=0}^{|C_2|-1} f_r(x_i) = 1 \tag{3}$$

The proposed co-operative cache mechanism blends two different cache replacement techniques. First, OPR, using forecasting and second, selective caching using relative frequency. The mathematical foundations & modelling of these techniques are as follows. A subset of pre-cache is MEC cache ($C_2 \subseteq C_1$) with a finite size. It comprises of $|C_1|$ most frequently used references from C_2.

Let, $\tau_i = \{t_{ik}\}$ be a set of time-stamps when the reference $x_i \in C_2$ occurs in the reference string R. By nature, it is a monotonically increasing sequence which is bounded below.

$$R = \bigcup_{i=0}^{|C_2|-1} \tau_i \iff \sum_{i=0}^{|C_2|-1} |\tau_i| = |R| = n \tag{4}$$

Let λ be the Lagrange Interpolator. It takes each time series τ_i and returns a polynomial $\Phi_i(j)$ of order d that fits into the given samples of times-stamps with minimal mean squared error. Let j be the index variable of the set τ_i. For fitting a curve of order d, according to the convergence criteria of Newton's divided difference technique, it needs at least $d+1$ data samples.

$$\lambda : \tau_i \to \Phi_i(j) \mid \Phi_i(j) = \sum_{k=1}^{d} a_k j^k \tag{5}$$

Hence the forecasting of the set τ_i with a period p can be written as,

$$\tau_i' = \{\Phi_i(|\tau_i| + 1), \Phi_i(|\tau_i| + 2) \dots \Phi_i(|\tau_i| + p)\} \tag{6}$$

Hence, the forecasted reference string R_F would be,

$$R_F = \bigcup_{i=0}^{|C_2|-1} \tau_i \tag{7}$$

The function γ returns the first occurrence of a reference $x_i \in C_2$ in the forecast R_F

$$\gamma: x_i \to [0, |R_f| - 1] \tag{8}$$

Let at time instance t, a page replacement is invoked. The metric h gives the distance between each cached reference and their earliest forecasted occurrences.

$$h(x_i) = i - \gamma(x_i) | \; x_i \in C_1 \tag{9}$$

Hence, OPR then uses R_F as future data references to select via ctim (v) for page replacement.

$$v = \max(h(x_i) \mid \forall x_i \in C_1) \tag{10}$$

Selective caching: In a classical caching system, a miss in an overflowing cache yields a replacement. It may so happen, that cached data have less frequency of occurrence than the victim. Consequently, the victim is more likely to appear before the cached data hence it causes a miss. In a MEC environment, fetching data from a remote MEC node is costly, hence it raises the overall cost significantly. This problem can be tackled with the proposed selective caching where it keeps track of the relative frequency of each reference in pre-cache (C_2). The cache (C_1) only holds the most relative frequently used references from C_2. Therefore, a miss event with a reference r_i will only be served as a replacement if the following condition is met.

$$f_r(r_i) > \min(f_r(x_i)) \mid x_i \in C_1 \tag{11}$$

An extrapolation approach has been used to forecast the relative frequency is proposed using polynomial regression. Since the change in relative frequency follows the law of *temporal locality of reference*, hence it perhaps

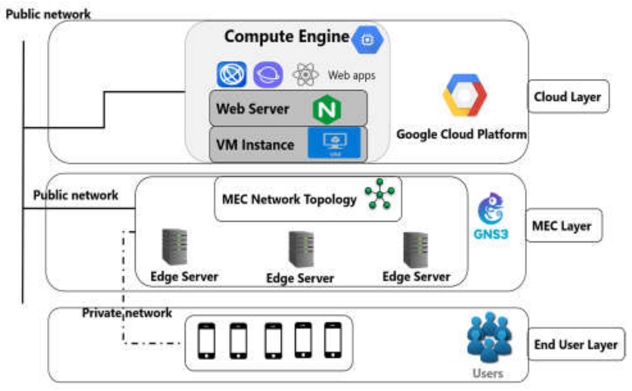

Figure 2 Experimental Set-up

fits in the time series criterion. Extrapolating the curve forecasts the probable future occurrences. In order to implement, *numpy.polyfit()*, a python based polynomial fit library is used. Depending on the origin of the fetched data, the replacement algorithm is determined, given the cache is full. The following are the two possible cases of origins.

a) Data from point of presence: Data obtained from point of presence is treated with the highest priority because the link between the MEC node and the point of presence is very costly in terms of latency. The proposed modified OPR algorithm is used if there is a need for replacement and the data obtained are cached. We try to maintain any data obtained from the point of presence in the local MEC cache units.

b) Data from MEC cluster: Data obtained from the same MEC cluster are given less priority because the link between the MEC nodes costs less. Therefore, the relative frequency of data obtained (f_d) from this link is compared with the minimum relative frequency (f_m) of the data in cache. If f_d is greater only then is the data cached else the data is sent to the user but not cached in the MEC node. By employing such approach, we redefined the key replacement algorithms policies which implies that the existing data are always replaced by the newly requested data upon request when the cache is full. Accessing data from the MEC cluster, would not incur much delay on the system since the access time to get the data from the MEC cluster is less than the time to get the data from the point of presence. This also reduces redundant caching in the MEC cluster.

B. Co-Operative and Hybrid Replacement Caching Algorithm (CHRCA)

The proposed algorithm is presented in Algorithm 1. It requires two inputs (window size and the degree of the fitted polynomial) and consists of 4 steps. Step 1 is an initialization of the MEC cache and the pre-cache which contains a set of previously requested data and its relative frequency. In step two a request is received and the relative frequency of the request rf_i is calculated. If the requested data is not in the cache, the algorithm checks if it can get the request from the MEC. The frequency of data obtained from MEC is compared with the frequency of the data in the cache and only cached when the later is higher. Data obtained from the cloud is always cached as the link between the MEC and the cloud is assumed to be costly.

V. Experiment & Results

In this section, details are provided of how the proposed algorithm was tested, which algorithms were compared and how the algorithm was deployed.

A. Experimental setup

In this subsection, the deployment set-up is explained. The components that make up the system and what tools and platform that was used are discussed. The diagram in Figure 2, is made up of three layers, this consists of the Cloud layer, MEC layer and End-user layer

a) Cloud Layer: In this article, Google cloud environment was used as the cloud platform. In the google cloud compute engine, a VM instance is deployed. A webserver is then installed which hosts the web applications.

Algorithm 1: Co-Operative and Hybrid Replacement Caching

```
Input: $W \in \mathcal{N}$   : window size
       $D \in \mathcal{N}$ : Degree of the fitted polynomial
Output: None
Data Structure: Multiple Priority Queue
Steps 1: Initialization
$C_1 \leftarrow \phi$            //Cache Memory
$C_2 \leftarrow \phi$            //Pre-Cache
Step 2: Get a Request
$C_1 \leftarrow C_1 \cup \{r_i\}$        //$r_i$ : Incoming reference
$C_2 \leftarrow C_2 \cup \{r_i\}$
Step 3: Calculate Relative Frequency
$rf_i \leftarrow \frac{r_i}{\sum r_i}$
Step 4: Indefinite Loop
$C_1 \leftarrow W\ most\ frequently\ used\ ref$
While $|C_1| \leq W$ do
{
      $\Lambda \leftarrow \{\lambda \mid list\ of\ timestamp\ of\ distinct\ reference\}$
   $f(t) \leftarrow PolyFit(\lambda, D)$
         // fit a polynomial of degree D

   $r_{t+1} \leftarrow Extrapolate(f(t+p))$
            //$p$ : period of Lagrange extrapolation
   // Page Replacement
   If Cache is full and Miss happens {
      If $rf_i \in C_3$   // Page comes from Cloud
            Replace with OPR
      Else If Page comes from MEC {
            If $\exists r_j \mid rf_j > \min(r_i \in C)$)
               $r_{depart} \leftarrow r_i \in C_1 \mid rf_i = \min(rf_i)$
               $r_{enter} \leftarrow r_j \in \{C_2 - C_1\}$
            Replace $r_{depart}$ with $r_{enter}$ using LFU
            //if the popularity by the relative frequency of a
            reference (entering reference) in pre-Cache but not
            in Cache outstands a reference in Cache (departing
            reference) then call replacement. As higher
            popularity is proportional to the probability of
            sooner reference.
      }
   }
}
```

b) MEC Layer: GNS3 was used to emulate the MEC cluster. GNS3 (Graphical Network Simulator- 3) is free software used to emulate complex networks [18]. In GNS3, a cluster of MEC network has been created. The MEC cluster nodes are interconnected using overlay networking provided by Open Daylight [19] that also acts as a Software-Defined Networking (SDN) controller for the testing platform. Each MEC node is a lightweight

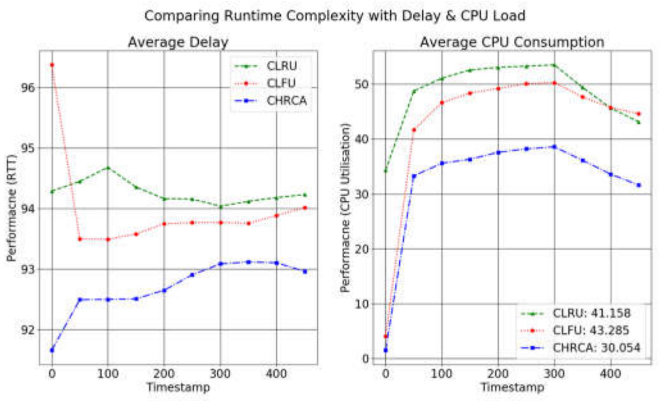

Figure 3 Lower RTT implies reduced access time due to high local hits. Less CPU consumption implies computational efficiency. CHRCA meets both criteria hence its fast and efficient

docker container. A docker container is a lightweight, standalone and executable package of software that has operating system level virtualization and includes everything needed to run an application. The MEC layer is connected to the Internet via the public network. A private network is also created within the GNS3 with which the MECs are connected. Quagga [20] is installed on the MEC docker containers and configured to serve as an access point for end-user devices. The proposed algorithm is written in python and deployed as a Docker application in the MEC Docker container.

Specification	**Amount**
Number of MECs deployed	3
Number of requests received per MEC	500
Case size of each MEC	4
Number of content items	20

Table 1 Experiment Specification

*c) **End-user layer**:* This layer is the final layer and it is made up of the end user devices. To simulate the users, we assume that the sample space of the generated requests uses Gaussian distribution [21].

B. Experiment Procedure

In this section, performance comparisons are made between the proposed algorithm and two case study algorithms. The first case study algorithm is a combination of Co-operative caching and Least Recently Used (CLRU). LRU algorithm chooses its replacement victim based on which cache reference that has been unused for the longest time [4]. While the second case study algorithm is a combination of Co-operative caching and Least Frequently Used (CLFU).The caching test specifications are summarized in Table1

Average	**CLRU**	**CLFU**	**CHRCA**
CPU utilization (%)	41.158	43.285	30.054
RTT (ms)	94.325	94.127	92.887

Table 2 Average Resource (CPU and RTT) Utilization

C. Experimental Results

In this section, we discuss the results obtained after experimentation and testing. Here a comparison of the resource utilization which includes the CPU and RTT (Round Trip Time: this is the RTT between the MEC and the web server) utilization is made for the Case study algorithms and CHRCA. A comparison is also made for the cache performance which includes local cache hits, cache misses, total cache hits.

a) Resource utilization comparison

Figure 3 shows the CPU and RTT utilization of each algorithm respectively over a period during the algorithm run time, it can be deduced be that CHRCA obtains lesser RTT with an average of 92.887ms. It can also be seen that

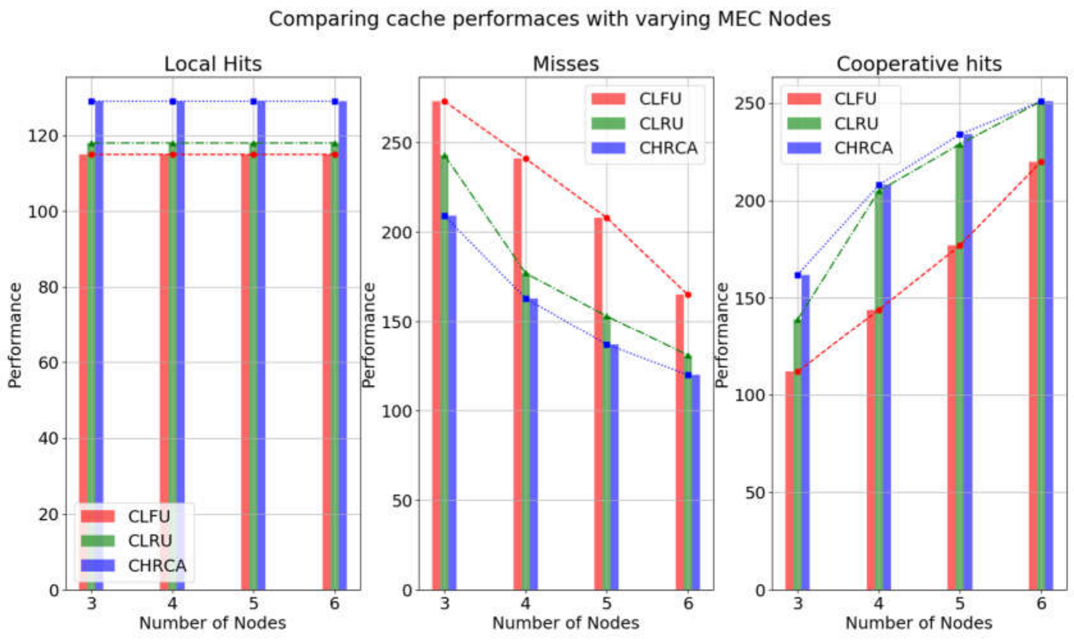

Figure 4 Number of nodes is proportional to the cache performance. However, CHRCA shows a better convergence (Higher Hits and lower misses) compared to CLFU and CLRU

CHRCA maintains a lower CPU utilization with an average of 30.054%.

Table 2 shows the average CPU and RTT utilization of each algorithm. The CPU utilization drops mainly when there are cache hits and the data is fetched locally from the device or from the MEC and not from the source.

b) Cache performance comparison

Figure 4 shows the local cache hits, cache misses, and MEC hits respectively of each algorithm over a varying number of MEC nodes. The reference string, cache size and the number of requests are kept constant while the number of MEC nodes is increased from 3 to 6 nodes. Since the reference string and cache size are the same throughout the test, the local cache hits remained the same. Increase in the number of nodes shows convergence in the misses and cooperative hits. Cooperative hits are data obtained from a remote MEC using the co-operative algorithm when a request is not cached in an MEC. Comparing the three algorithms, it can be deduced that the proposed algorithm has a better overall performance as the proposed algorithm generates more cache hits, fewer cache misses and more MEC hits compared to the case study algorithms.

VI. Conclusion & Future Scope

In this paper, a co-operative and hybrid replacement caching algorithm (CHRCA) for MEC is presented to improve the caching efficiency. The proposed algorithm is a blend of two cache replacement algorithms (OPR and LRU) and a distributed co-operative caching algorithm. Lagrange polynomial extrapolation algorithm was leveraged to predict the future occurrences of requests using historical data of relative frequency of the requests. Cache redundancy in cooperative caching is reduced by our selective caching approach. Experimental results show a better convergence in terms of CPU load and delay, compared to CLFU and CLRU.

There are two planned future extensions,

i) Recurrent Neural Networks with Long Short-Term Memory for better prediction and
ii) Incorporating IPFS for distributed caching in Information-Centric Networking environment.

Acknowledgement

This project received research funding from SONNET and the H2020-MSCA-RISE-2016 European Framework Program with contract number 734545.

References

[1] 5G PPP Architecture Working Group, "View on 5G architecture," 2, 2016.

[2] E. E. Ugwuanyi, S. Ghosh, M. Iqbal, and T. Dagiuklas, "Reliable Resource Provisioning Using Bankers' Deadlock Avoidance Algorithm in MEC for Industrial IoT," *IEEE Access*, vol. 6, pp. 43327–43335, 2018.

[3] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," *IBM Syst. J.*, vol. 5, no. 2, pp. 78–101, 1966.

[4] "Operating System Concepts, 9th Edition," *Wiley.com*, 1994. [Online]. Available: https://www.wiley.com/en-us/Operating+System+Concepts%2C+9th+Edition-p-9781118063330. [Accessed: 11-Sep-2018].

[5] P. Panda, G. Patil, and B. Raveendran, "A survey on replacement strategies in cache memory for embedded systems," in *2016 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, 2016, pp. 12–17.

[6] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[7] A. Jain and C. Lin, "Back to the Future: Leveraging Belady's Algorithm for Improved Cache Replacement," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 78–89.

[8] E. K. Markakis, K. Karras, A. Sideris, G. Alexiou, and E. Pallis, "Computing, Caching, and Communication at the Edge: The Cornerstone for Building a Versatile 5G Ecosystem," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 152–157, Nov. 2017.

[9] S. Kim, "5G Network Communication, Caching, and Computing Algorithms Based on the Two-Tier Game Model," *ETRI J.*, vol. 40, no. 1, pp. 61–71, Feb. 2018.

[10] N. Wang, W. Shao, S. K. Bose, and G. Shen, "MixCo: Optimal Cooperative Caching for Mobile Edge Computing in Fiber-Wireless Access Networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, 2018, pp. 1–3.

[11] X. Huang, Z. Zhao, and H. Zhang, "Cooperate Caching with Multicast for Mobile Edge Computing in 5G Networks," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, 2017, pp. 1–6.

[12] C. Yang, H. Li, L. Wang, and D. Tang, "Exploring the behaviors and threats of pollution attack in cooperative MEC caching," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[13] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, and M. Guizani, "Energy-Efficient Caching for Mobile Edge Computing in 5 G Networks," 2017.

[14] M. Neishaboori, "Implementation and Evaluation of Mobile-Edge Computing Cooperative Caching," Aalto University School of Science, Espoo, Finland, 2015.

[15] X. Liu, J. Zhang, X. Zhang, and W. Wang, "Mobility-Aware Coded Probabilistic Caching Scheme for MEC-Enabled Small Cell Networks," *IEEE Access*, vol. 5, pp. 17824–17833, 2017.

[16] A. Abouaomar, A. Filali, and A. Kobbane, "Caching, device-to-device and fog computing in 5 lt;sup gt;th lt;/sup gt; cellular networks generation : Survey," in *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2017, pp. 1–6.

[17] B. Li, H. Zhang, and H. Lu, "User mobility prediction based on Lagrange's interpolation in ultra-dense networks," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2016, pp. 1–6.

[18] "GNS3 | The software that empowers network professionals." [Online]. Available: https://www.gns3.com/. [Accessed: 12-Sep-2018].

[19] "Welcome to OpenDaylight Documentation — OpenDaylight Documentation Fluorine documentation." [Online]. Available: https://docs.opendaylight.org/en/stable-fluorine/. [Accessed: 14-Oct-2018].

[20] "Quagga Software Routing Suite." [Online]. Available: https://www.quagga.net/. [Accessed: 31-Oct-2018].

[21] G. J. O. 't Veld and M. C. Gastpar, "Caching Gaussians: Minimizing total correlation on the Gray-Wyner network," in *2016 Annual Conference on Information Science and Systems (CISS)*, 2016, pp. 478–483.