Mobile Edge Computing MEC is a concept that proposes to bring the computing and storage resources in close proximity 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 are able to host mobile applications and serve web contents. The particularity of MEC is that servers are implemented directly at the base stations which enable edge caching and ensure deployment in close-proximity to mobile users. However, Edge Caching still has issues reducing latency.

Key Words： Caching, Mobile Edge Computing, LRU, LFU, ARC, QoS, Fuzzy Logic

（用英文逗号“,”分隔）

In order 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 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 popularity time varying 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 a number of popular algorithm 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 dataset with hit rates approaching 90%.

The third part describes 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 are 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 but a further investigation is needed.