# Cluster-based Cache Replacement in 5G Network

By

Saleh Yahia Saleh Kharbish

A master thesis presented to
Ryerson University
In partial fulfillment of the requirements
Of the degree of

Master of Applied Science

Department of Computer Network
Ryerson University
Toronto, Ontario, Canada
2018

**AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS**

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my dissertation may be made electronically available to the public.

# Abstract

In this thesis, we proposed a Cluster-Based Cache Replacement (CBR) scheme for 5G Networks to reduce the backhaul traffic. We developed our scheme based on the understanding of the degradation of the performance of the cache placement algorithm. We expect that whenever file request pattern differs from the file popularity distribution, such as unpopular files become more popular or vice versa, the caching system should experience performance degradation. We address this problem by presenting a cache replacement scheme based on the idea of Least Frequency Used (LFU) replacement policy, but we consider only the recent request to avoid cache pollution. We evaluated the performance of CBR through simulation and compared its performance with LRU that is widely used as a cache replacement technique in practice. We simulated three different configurations of LRU scheme in a cluster-based mobile network model. Our simulation results show that the CBR outperforms LRU, where it reduces the miss ratio from 86% to 76% and the backhaul traffic from $3.67 \times 10^5$ to $3.47 \times 10^5$ MB with 10% of cache size. This superior performance it achieves by fewer replacement decisions and storing more files in the cache.

# Acknowledgments

I would like to express my heartfelt appreciation and gratitude to my supervisor, Professor Muhammad Jaseemuddin, for his support, patience, guidance throughout my graduate studies and in proofreading and providing many valuable comments that improved the presentation and contents of this thesis.

I would like also to express my thanks to my program director, Professor Bobby Ma for his guidance, encouragement and continued support during my studies.

My thanks also go to my colleagues for providing many valuable comments, interesting discussions and friendly atmosphere.

I would like to thank my wife, Iman for her patience, support and love during the past three years in making this thesis possible and to my parents for giving me the courage and strength to persevere.

Last, but not least, I would like to thank the Libyan government for providing me with the financial support.

# Table of Contents

# List of Tables

# List of Figures

# List of abbreviations

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| BBU | Baseband Unit |
| CBR | Cluster-Based Cache Replacement algorithm |
| CDMA | Code Division Multiple Access |
| C-RAN | Cloud Radio Access Network |
| EDGE | Enhanced Data Rate for Global Evolution |
| EPC | Evolved Packet Core |
| FAR | Farthest Away Replacement |
| FDD | Frequency Division Duplexing |
| FDMA | Frequency Division Multiple Access |
| GDS | Greedy Dual-Size |
| GPRS | General Packet Radio Service |
| GPS | Global Positioning System |
| GSM | Global System for Mobile communications |
| HSDPA | High Speed Downlink Packet Access |
| HSPA | High Speed Packet Access |
| HSUPA | High Speed Uplink Packet Access |
| IMT-2000 | International Mobile Telecommunications 2000 |
| IoT | Internet of Things |
| ITU | International Telecommunications Union |
| LDIS | Location Dependent Information Services |
| LFU | Least Frequency Used algorithm |
| LFUA | LFU-Aging |
| LFUDA | LFU with Dynamic Aging |
| LRU | Least Recently Used algorithm |
| LTE | Long Term Evolution |
| MARS | Mobility Aware Replacement |
| MBS | Macro Base Station |

| | |
|---|---|
| mMIMO | massive multiple-input multiple-output |
| mmWave | millimeter-waves |
| NFV | Network Function Virtualization |
| OFDMA | Orthogonal Frequency Division Multiple Access |
| PPRRP | riority Predict Region Based Replacement Policy |
| PRRP | Predict Region Based Replacement Policy |
| RAN | Radio Access Network |
| RRH | Remote Radio Heads |
| SAE | System Architecture Evolution |
| SBSs | Small Base Stations |
| SDN | Software Defined Networking |
| SMS | Short Message Service |
| TDD | Time Division Duplexing |
| TDMA | Time Division Multiple Access |
| UDNs | Ultra-Dense Networks |
| UMTS | Universal Mobile Telecommunication System |
| VNF | Virtualized Network Functions |
| WPRRP | Weighted Predict Region Based Replacement Policy |

# Chapter 1

# Introduction

The evolution of the technology and specifically smart phones and tablets over the last decade has provided people with powerful devices to access Internet at any time and anywhere. In addition, people tend to share their daily life experience on social networks by posting high resolution pictures and videos. These social networks are one of the main reasons based on some studies [3][4] that contribute to the dramatic increase of internet demand. Another reason for this exponential increment to data volume is the mobile multimedia services. Ericsson reported in 2015 that the projection for mobile data usage by the year 2020 shows its increase by a factor of 250 [7]. The tremendous increase in user demand is starting to exceed the capability of 4G mobile networks. Current networks are not capable of handling huge number of devices and appliances to be connected to the internet giving rise to the Internet of Things (IoT). Further, some new applications require ultra-low latency and ultra-high reliability, such as remote surgery in medical services, self-driving cars, and public service applications. To achieve 5G network capacity goals, three key approaches are suggested [5]: (i) Ultra-Dense Networks (UDNs): this approach is already adopted by 4G wireless networks and known as Small Base Stations (SBSs), but denser networks where the inter-sites distance is just a few meters is what 5G wireless mobile networks need to meet their network capacity expectations. (ii) Large quantities of new bandwidth: to achieve higher network capacity, millimeter-waves (mmWave) with frequency range of 30-300 GHz is proposed as a carrier frequency. This range can provide higher bandwidth capacity. (iii) High spectrum efficiency: high numbers of antennas can be compact on a small chip in high frequencies range where the wave length is very short (mmWave), and this allows to exploit the technology of massive multiple-input multiple-output (mMIMO) on both ends.

Small cell networks address challenges of low-bandwidth in the access network by providing users with high-speed access link while increasing the network capacity. However, connecting a large number of small cell base stations providing high-speed access links with small number of macro-

base stations poses a new challenge for mobile operators with the demand for high capacity and expensive backhaul links to carry cumulative access traffic. Currently, high capacity backhaul links, such as fibre, are rarely available. One approach to mitigate the demand on backhaul link is to cache frequently used content within the network to avoid repeated download of those content causing redundant traffic on the backhaul link. Hence, Caching in the small cell is proposed to save backhaul bandwidth and to provide user with better experience by bringing content closer [1][2][3][6]. Caching of popular files inside the network of SBSs allows subsequent requests for those files to be served from one of those SBSs, which reduces the traffic of the backhaul link that serves these SBSs from the servers outside the network. Most of the caching schemes in the 5G literature focuses on formulating cache placement problem as an optimization problem using different parameters. In our prior work [1], we developed a cache placement scheme based on a recommender system approach considering a number of parameters such as user-file association and user-base station connectivity status etc. In one instance of running the placement algorithm, a decision is made, called *placement decision*, to populate all the caches in a cluster of SBSs by selecting files from the library of files. However, due to the variation of the file popularity, the initial cache placement may not be effective for a long time. Running these algorithms frequently will overload the system because of different cache assignment. Hence, moving files from one SBS to another because of different placement decision is not practical. It is recommended to run these placement algorithms periodically, such as once a day, and perform cache loading with files in the off-peak hours [3]. We call the time between two cache placement decisions an *epoch* as shown in figure 1.1.

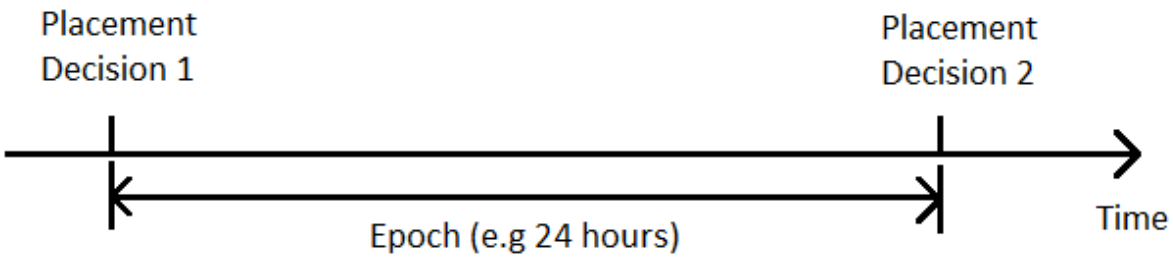


**Figure 1.1.** The time between two cache placement decision is called an epoch.

The cache placement scheme at the beginning of an epoch makes cache placement decision based on an estimate of popularity of files and an expectation of reuse of those files in the cache. However, user access pattern may deviate from initial popularity estimate because of the demand for the files that are not in the cache. More the deviation is higher will be file miss in the cache causing surge in the backhaul link. We have verified this phenomenon through simulation and discussed in Section 4. Hence, a mechanism is needed to address the problem caused by the variation of file popularity within an epoch. In this thesis, we proposed a Cluster Based Cache replacement scheme (CBR) to track the variation in file popularity within an epoch exhibited in the deviation of user access pattern from the initial popularity estimate. Our scheme ensures that the files that lose its popularity are replaced by files that gain on their popularity. It considers only a set of recent requests in making replacement decision to avoid cache pollution. It is designed for caching in a cluster of a small cell network that is connected to one Macro Base Station (MBS). It is a continuation of Cluster-Based cache placement algorithm that is based on the matching problem where user-to-file and helper-to-file associations are used to define which file to be cached and in which helper [1].

We simulated the performance of our scheme in a cluster of small cell base stations and found significant benefit in reducing the backhaul traffic. For example, we simulated the network model of [1] to calculate the miss ratio and backhaul traffic. We formed a cluster of 10 SBSs and 100 users with a library of 500 files with file size ranging from 10 to 70 MB, and total cache size is good enough to store 10% of the files in the library. We simulated 10,000 user requests following Zipf distribution with skew $= 0.4$ . These calculations consider the variation of the file popularity, and the results show that the miss ratio of the cache placement in [1] ranges from 74% to 89% and the backhaul traffic ranges from $3.44 \times 10^5$ to $3.88 \times 10^5$ MB. However, these results are improved using the CBR replacement algorithm, where the miss ratio ranges from 76% to 81% and the most important parameter, backhaul traffic, ranges from $3.41 \times 10^5$ to $3.5 \times 10^5$ MB. It is obvious that CBR algorithm can adapt with the changes in the file popularity and save the backhaul traffic. Also we found a substantial difference in the miss ratio of CBR in comparison with widely used LRU, which shows the effectives of tracking changes in popularity pattern of files for file replacement decision.

This thesis is organized in five chapters. Chapter 1 is an Introduction. Chapter 2 discusses the requirements and architecture of 5G mobile network and background work on cache replacement techniques. Chapter 3 discusses the network model and CBR algorithm. Simulation results and a performance comparison between Least Recently Used algorithm (LRU) and CBR are discussed in Chapter 4. Finally, the concluding remarks and future work are presented in Chapter 5.

# Chapter 2

# Background

## 2.1 The Evolution of Mobile networks: 1G to 4G

The first commercial mobile network was deployed in early 1980s. G1 network offered only a voice service and it was an analog network based on Frequency Division Multiple Access (FDMA). Latter in 1990s, Global System for Mobile communications (GSM) was deployed and became an international standard for 2G mobile networks. Deploying digital transmission and telephony switching offer better voice quality, more network capacity and more services such as Short Message Service (SMS) and roaming services. 2G mobile networks offer more capacity by exploiting both the FDMA and Time Division Multiple Access (TDMA) technologies. General Packet Radio Service (GPRS) was introduced as an evolution of 2G networks and was called 2.5G. It adds packet-switching data service to GSM and offers low speed data to mobile users. Enhanced Data Rate for Global Evolution (EDGE) offers higher data speed than GPRS by deploying higher order modulation scheme. GSM/EDGE has kept on advancing with their recent arrival of the 3rd Generation Partnership Project (3GPP) standard supports wider bandwidth [7].

The next generation for GSM/EDGE was deployed to achieve higher data speed and is known as the third generation of wireless mobile networks (3G). The first major 3G network that has met the requirements set by the International Mobile Telecommunications 2000 (IMT-2000) is the Universal Mobile Telecommunication System (UMTS) which is based on the technology of Code Division Multiple Access (CDMA). In 3.5G, two Radio Access Network (RAN) approaches were suggested, CDMA 2000 specified by 3GPP2 and High Speed Packet Access (HSPA). HSPA was a combination of both High Speed Downlink Packet Access (HSDPA) and High Speed Uplink Packet Access (HSUPA), where both are standards of 3GPP release 5 and 6 respectively. HSPA offers a data speed of up to 14.4 Mbps for downlink and up to 5.76 Mbps for Uplink and evolved to offer higher rates by introducing Multiple-Input and Multiple-Output (MIMO) technology. By deploying more features, HSPA later became known as HSPA+ [7].

To accommodate for the increasing demand of data, it was a necessity to go to the next step, the fourth generation of wireless mobile network 4G known as the Long Term Evolution (LTE). The first LTE standard accepted by 3GPP was known as LTE Release 8. With Orthogonal Frequency Division Multiple Access (OFDMA) as an air interface technology and new core network architecture, System Architecture Evolution/Evolved Packet Core (SAE/EPC), LTE offers high peak speed of up to 326 Mbps and low round trip latency of around 20 ms. LTE release 10 offers higher peak speeds up to 3 Gbps and 1.5 Gbps, downlink and uplink, respectively. LTE releases 12 and 13 support more features such as higher order of MIMO, carrier aggregation of Time Division Duplexing (TDD), Frequency Division Duplexing (FDD), and improved heterogeneous network support [7].

## 2.2 Requirements for 5G wireless communication systems

In September 2015, the International Telecommunications Union (ITU) characterized the prerequisites of key capabilities of 5G by specifying eight key performance indicators. Figure 2.1 [5] shows the 5G key performance indicators with comparison to 4G networks.

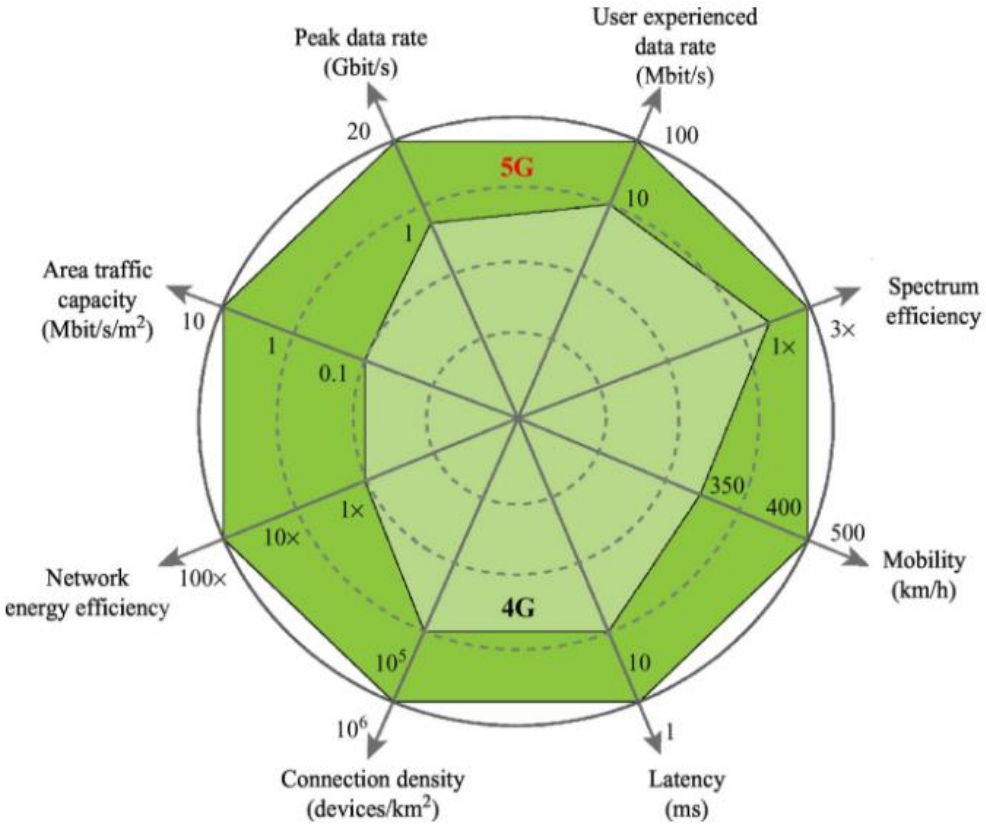


**Figure 2.1.** The 5G key performance indicators with comparison to 4G network [5]**.**

Some of the key performance indicators for 5G are described below [5]:

- **Peak data rate:** The peak data rate is supposed to be 10-20 Gbps in 5G which is at least 10 times of the peak data rate of current 4G, 1 Gbps.
- **User experienced data rate:** User experienced data rate is expected to be 100 Mbps in urban and suburban areas and might reach 1 Gbps for some scenarios.
- **Mobility:** Mobility in 5G is expected to surpass the maximum support mobility in 4G which is up to 350 km/h and 5G goal is to achieve a mobility of 500 km/h.
- **Latency:** The aim of 5G is to support the applications and services that need low latency. 5G networks are expected to have a latency of less than 1 ms, 10 times less than the latency experienced in 4G mobile networks.
- **Connection density:** 5G is expected to accommodate $10^6$ devices/km$^2$, 10 times more than the capability of current networks.
- **Area traffic capacity:** The area traffic capacity in 5G is expected to be 10 Mbit/s/m$^2$, where current networks support area traffic capacity of up to 0.1 Mbit/s/m$^2$.

## 2.3 Architecture of 5G networks

Network architecture design aims to define network components and their functionality and interaction to perfume end-to-end services under a set of standardized key requirements. Scalability, flexibility, and Multi-vendor equipment internetworking in the network architecture are essential for network development and expansion.

Deploying Network Function Virtualization (NFV) and Software Defined Networking (SDN) in the core of 5G network can provide more features and flexibility for the future mobile networks. The idea of NFV is to replace a variety of network equipment with standard powerful servers that virtualize different kinds of network functions. Instead of installing and integrating complex hardware to lunch a new service, NFV technology offers a flexible and cost-effective technology by deploying Virtualized Network Functions (VNF) which might consist of more than a virtual machine to perfume the functionality of certain hardware. SDN is meant to address the complexity of traditional distributed networks by separation of control and data. In SDN, Control plane is centralized in one programable node that controls the data plane which is represented in switches and router [7].

The ratio of traffic demand in mobile networks in peak and off-peak hours can be 10 times [9]. Base stations are designed to accommodate the service in the peak hours. As a result, many base stations are lightly utilized in the off-peak hours. Cloud Radio Access Network (C-RAN) separates baseband signal process performed by the Baseband Unit (BBU) from radio functionality which is the responsibility of Remote Radio Heads (RRH). In C-RAN, low cost RRH is connected through fronthaul links to centralized BBU pool built on cloud-based hardware. Centralized BBU pool gives a flexibility to manage the network resources where they are needed in the peak traffic demand. Figure 2.2 [9] shows the main elements of 5G mobile networks.

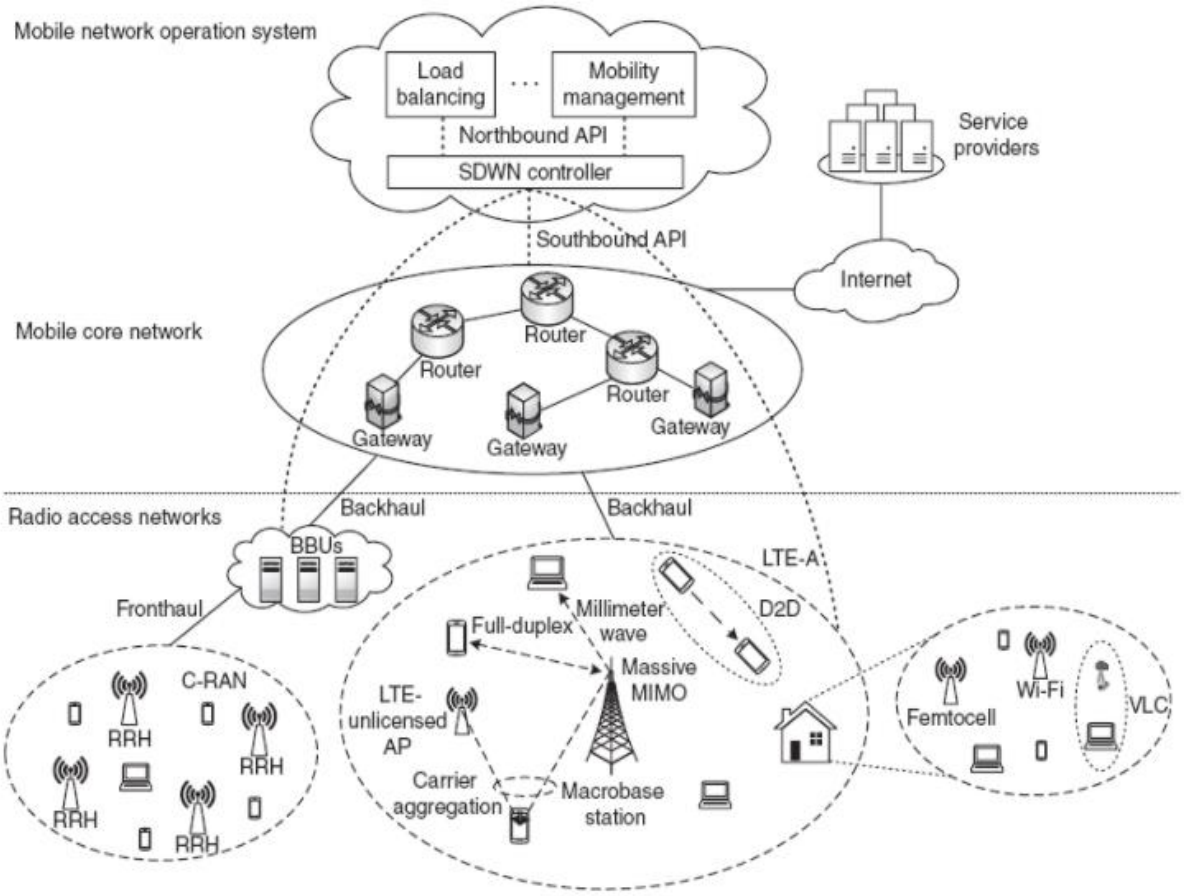


**Figure 2.2.** The main elements of 5G mobile network [9].

## 2.4 Cache Replacement Algorithm

Cache Replacement algorithms are used in any computer system with caching capabilities to evict saved content or files, making a space to cache new arrival contents. There are many policies and choosing the appropriate algorithm is crucial and it depends on the design of the system.

### 2.4.1 Cache replacement performance metrics

Performance metrics are parameters calculated to evaluate the replacement algorithm and to identify which algorithm works better for a workload or a particular system. There is no optimal cache replacement algorithm for all systems. Depending on what the algorithm is optimizing, metric measurements are performed. The main performance metrics are:

1. Hit rate: is the rate of the number of requests served from the cache to the total number of requests. This metric gives a better indication for traditional caching systems where the variation of the size and cost of object is very small. To perform higher hit rate, its preferred to cache small size data items instead of caching large once.
2. Byte hit rate: is the rate of data bytes served out of the cache to the total served bytes. In such systems where large objects are heavily used, algorithms that use the byte hit rate as a metric perform better than algorithms that use the hit rate, where the later prefer to evict large items.
3. Delay saving ratio or latency reduction: is the ratio of the reduced delay by serving out of cache to the total delay when requests served from the source. Delay saving ratio metric might not be accurate due to the variation of the download delay.

There are other performance metrics like Disk and CPU performance, but these are rarely used or studied.

## 2.5 Classification of Cache Replacement Policies

Web caching replacement algorithms can be classified into traditional web cache replacement policies and mobile web cache replacement policies [8].

### 2.5.1 Traditional Web Cache Replacement Policies

These policies can be categorised based on their strategy of selecting the evicted objects.
Following are some of the main strategies used in cache replacement algorithms:

#### 2.5.1.1 Time-Based Strategy

Least Recently Used (LRU) is the widely use cache replacement algorithm in this category. The concept behind this algorithm is that the near past is a good reflection of near future due to locality reference in time. In this algorithm, the data objects are listed in the order of their access time

where the recently used object is moved to the top of the list and the one not used for a long time remains at the bottom. When the cache is full and a new file is requested, it is cached and listed at the top and one file from the bottom is evicted. The main problem of LRU is that it does not consider the file size, and one large file size could replace a group of small files. So, in a system where there is a high file size variance, it would be difficult to maintain a high hit rate. Some LRU variants are used more in practice, for example LRU-threshold, where threshold here is the maximum size of cacheable files. Any file size greater than the threshold is not cached. This can save couple of small objects to be evicted in order to cache one large file. Another LRU variant is LRU-MIN, it minimizes the number of evicted objects by considering the size of the new cached object (S) and classifies the cached objects into two lists; smaller and larger than S where LRU replacement is applied to the later list. If cached objects' size is less than S, then listing classification threshold will be S/2 and objects greater than S/2 will be removed until needed space is created. By using this technique large files are removed first, and high hit rate can be maintained [10].

#### 2.5.1.2 Frequency-Based Strategy

Least Frequently used (LFU) policy is the main policy of frequency-based algorithm, it evicts the object with least frequency access. This algorithm tends to keep popular contents cached and replace the rarely used ones. However, some contents might build a high reference count over time and can not be replaced even if they are not used for a long time leading to a cache pollution. To reduce the effect of high referenced object which might never be used again, a variant called LFU-Aging (LFUA) and LFU with Dynamic Aging (LFUDA) are proposed [11]. LFUA calculates the average frequency of all counters and when the average reaches a certain threshold it divides the counters values by 2. Assigning the proper threshold might be a problem, so LFUDA has a different concept, a key factor $K_i$ is calculated for each object $i$ using the following equation:

$K_i = F_i + L$

Where $F_i$ is the frequency count for object $i$ and $L$ is dynamic aging. This aging is initialized to zero and updated with the key factor of evicted object, so the new cached content key factor will build on the evicted object key factor. This prevents the recently cached content to be evicted first.

#### 2.5.1.3 Function-Based Strategy

One of the examples of function-based policy is Greedy Dual-Size (GDS) algorithm. It associates a value to each element in the cache based on the cost function, which is not defined and can be a calculation of some preferences that are needed to be optimized in the network like the delay of retrieval of the objects, link utilization, number of hops between the original server and the cache server. This cost is divided by object size and the objects with lower values are replaced first. The performance of function-based algorithms depend on how well cost function is implemented [8].

### 2.5.2 Mobile Web Cache Replacement Policies

Due to the mobility of mobile clients, limited cache size and battery life, traditional web caching policies are not suitable for mobile environment. It is very crucial to find proper cache techniques to maintain high hit ratio and reduce the delay experienced by users as well as saving the mobile client's battery. Cache replacement policies in mobile network can be categorized into Location-based, cost-based and User based policies.

#### 2.5.2.1 Location based policies

An important service in mobile network is known as Location Dependent Information Services (LDIS) based on Global Positioning System (GPS). This service provides users with the ability to access information related to their geographical location such as emergency services, information of nearest facilities like gas stations, ATMs, and restaurants. Many replacement policies are location based like Manhattan Distance Based Policy, Farthest Away Replacement (FAR), Mobility Aware Replacement (MARS), Predict Region Based Replacement Policy (PRRP), Weighted Predict Region Based Replacement Policy (WPRRP) and Priority Predict Region Based Replacement Policy (PPRRP) [8].

In Manhattan Distance Based Policy, cached items with higher distance between the user terminal location and source location are evicted first. Whereas in FAR policy, cached items are classified into two categories: in-direction and out-direction items and the latter is replaced first. In [12], PPRRP also classifies the cached items to in-direction and out-direction items, but this classification is based on the prediction of the user movement in the near future. It also considers the data size and access probability.

#### 2.5.2.2 Function-based policies

In function-based policies, location and the mobile client movement are not considered in the replacement decision, SAIU cache replacement for on-demand wireless broadcast network. It considers data size ($S_i$), data retrieval delay ($L_i$), access probability ($A_i$) and update frequency ($U_i$) to calculate the gain for all objects and the one with minimum value is replaced.

$Gain(i)= (L_i * A_i) / (S_i * U_i)$

The performance evaluation [13] shows that SAIU outperforms LRU and LRU-MIN in terms of average access latency, byte hit ratio and average access latency where cache size ranges from 2% to 10 % of the database set size. However, a recent version of SAUD [14] (Min-SAUD) outperforms both mentioned cache replacement algorithms. Min-SAUD cache replacement considers the following aspects: access probability, update frequency, data size, retrieval delay and cache validation cost.

#### 2.5.2.3 User-Relationship-Based Policy

In [15], a user-relationship-based cache replacement policy is designed to combine LRU algorithm and user relationships between the requestor and generator on social networks. In this strategy, Cache Block Priority Value is calculated for each item in the cache and blocks with minimum value are evicted.

Cache Block Priority Value is a summation of Closeness Value (CV) and LRU value. When content is requested from cloud server, a CV which is calculated by cloud server based on the relation between the requestor and generator is assigned with the request. The closer connections between requestor and generator result in higher CV. LRU value is calculated based on the size and the recency access for the items. The algorithm shows maximum performance when the Cache Block Priority Value is 70% CV and 30% LRU, and it outperforms the LRU cache replacement algorithm in terms of hit ratio by almost 20%.

## 2.6 Cache Placement and Replacement in Small Cell Base Stations (SBS)

Small cell networks, also known as helpers, is one of the main architectural changes that increases the network capacity in LTE and 5G mobile networks. Many researchers studied the benefit of caching in SBS and its effect on improving users experience and reducing the traffic of the expensive backhaul links in the network. Bringing the contents closer to the users shows a

significant improvement in terms of users delay experience and backhaul traffic. In [2], mobile users are considered to have access to multiple SBSs. These SBSs should cache different content to increase the probability of backhaul offloading and in order to achieve it, file to SBS assignment is studied. In [16], collaboration between base stations and diverse content popularity distributions is considered in file to cache assignment. In [1][3], collaboration is considered only between SBSs that are connected to the same macro base station. These studies only consider the file to cache assignment but due to the variation of the file popularity, running these algorithms frequently will change the cache assignment. Hence, moving files from one SBS to another is not practical and one of the recommendations is to run these algorithms once a day and do the cache assignment in the off-peak hours [3]. To the best of our knowledge, there is no study that addresses the problem of cache replacement in duration between these cache assignment sessions in cluster based or collaborative small-cell cellular networks. In this work, a cache replacement algorithm is proposed for cluster based or collaborative small-cell cellular network that tracks the users' requests and makes the decisions to replace the unpopular files with popular ones. In this study the cache replacement algorithm was used utilizing the cache assignment model in [1]. LRU cache replacement is studied as well and three different configurations were set for LRU replacement: (i) replacement with one file: replacement happens only if the size of the least recently used file is greater than the requested file, (ii) replacement with up to two files: here algorithm allows up to two files to be evicted to provide the needed space for caching, (iii) replacement of any number of files in order to cache the requested file.

## 2.7 Deployment of LRU in Cluster of SBSs

The LRU algorithm was deployed as follows: A cluster of M small base stations, K connected users, a library of P files, and cache size of C for each SBS were employed. For each cached file in the system a counter was assigned which goes down by one on each request except for the request file, where its counter will be reset to the default value. The lower value of the file's counter, the least recently used file. A learning period, a period of time where the algorithm tracks users' requests without taking any replacement decision, was also set so that the algorithm can learn which file is least recently used and to make sure that the algorithm's decision is accurate.
After the expiration of the learning period, when a request is received, the algorithm checks the file with minimum counter value which is called a *candidate file* and defines the SBS that caches

the *candidate file* and called it *candidate SBS*. The file eviction process depends on the three different sets of thresholds mentioned above:

(i) Replacement with one file: the replacement process happens only when the size of the Nominated File is equal or greater than the requested file.

(ii) Replacement with up to two files: in this deployment, if the eviction of the *candidate file* is not enough for caching, the algorithm defines the caching SBS and finds the second least recently used file in this SBS. Cache replacement happens when the size of *candidate file* and the second candidate file is enough for replacement. Other wise, the requested file will not be cached due to the cache size constraint.

(iii) Replacement of any number of files: the algorithm goes through all the processes of (i) and (ii) and if there is no enough space, it evicts the least recently used files in the *candidate SBS* until enough free space is provided.

We used the LRU algorithm for comparison with our proposed cluster-based cache replacement algorithm which combines the recency and frequency aspects in the cache replacement decision. Also, it defines a utility value for candidate files by considering their size and hits, and files with minimum utility value are replaced.

# Chapter 3

## 3.1 System Model for Collaborative Small-Cell (Multi-cell) Cellular Networks

In our system, we consider a single Base Station known as Evolved Node B (eNodeB or eNB) connected to the backbone of cellular network that is known as Evolved Packet Core (EPC) through a backhaul link. It serves $K$ users through $M$ number of small cells base stations SBSs, also known as helpers. These SBS are distributed within the coverage of the eNB as shown in Figure 3.1, forming one cluster. Each helper is connected to eNB and equipped with a cache storage size $C$. In this thesis, we consider that these caches are already fully utilized, and cache number of files based on our previous work of Cluster-based Cache Placement in 5G network [1]. The above cache placement scheme makes the placement decision of files from the library of $F$ files with different sizes on the helpers within a single cluster that have the same cache size. Each helper within a cluster caches unique files.

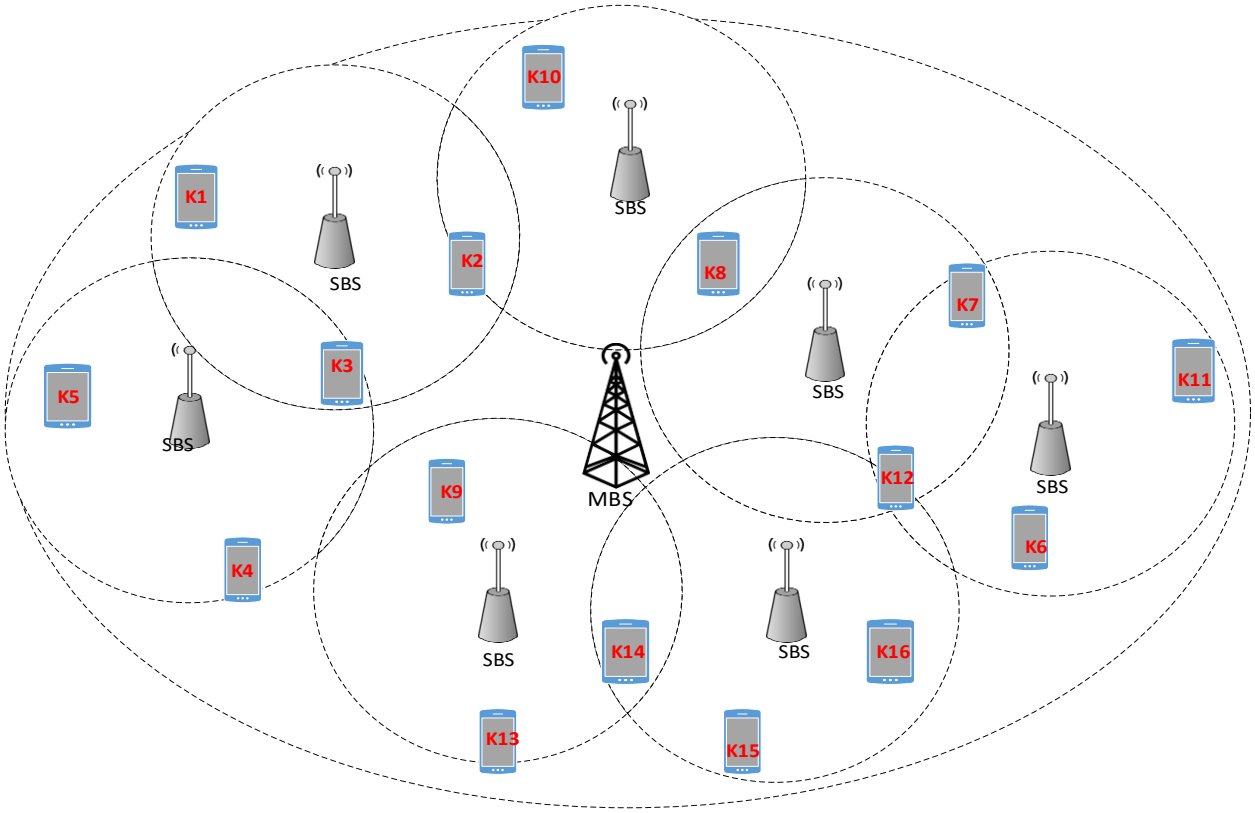


**Figure 3.1.** Network topology: cluster of SBSs connected to MBS**.**

The Macro Base Station (MBS) in this topology keeps track of all connected users and their requests in its coverage range. Each user is connected at least to one SBS. In the case of a user connected to more than one SBS, the user sends the file request to the SBS that has better link quality. The request might be served from the same helper or from other helpers within the cluster that have a connection with the user. Following are some definitions of the important concepts used in our algorithm:

1. **Serving SBS** is the SBS that receives a user's requests. It is also the SBS with the best link quality among all the other SBSs for that specific user.
2. **Caching SBS** is the SBS that caches the requested file.
3. **Local Hit** is the hit count for the requested file when the user requesting a file has a connection with the caching SBS.
4. **Neighbor Hit, Local Miss**: when a user requests a file that is cached in the SBS that is not directly connected to the user, then the request is served by another caching SBS within the cluster. In this case, the caching SBS sends the file to the serving SBS through either direct SBS-SBS link or SBS-MBS-SBS links. Neighbor hit is counted in the system as well as local miss for the serving SBS.
5. **Total Hit** is the sum of the local and neighbor hits.
6. **Cluster Miss**: Every time the request is served from the backhaul of the MBS, a cluster miss is counted.
7. **Candidate SBS** is the SBS that has minimum total hits of all the cached files among other SBSs within the same cluster.
8. **Candidate files** are files cached in the candidate SBS and might be evicted under certain conditions.

## 3.2 CBR Algorithm

Our proposed scheme is based on the idea of Least Frequency Used (LFU). Unlike general LFU, we only keep track of the requests originated within a recent time window, called $W$ , whose span can be adjusted depending upon how long a history is considered in making the replacement decision. The algorithm keeps a record of Local Hits, Neighbor Hits, Total Hits, Local Miss, and Total Miss for each file stored in every helper. For example, the replacement decision for the

request at time $t_x$ is based only on traces of the requests that are made in the period $t_x - W$ as shown in Figure 3.2.

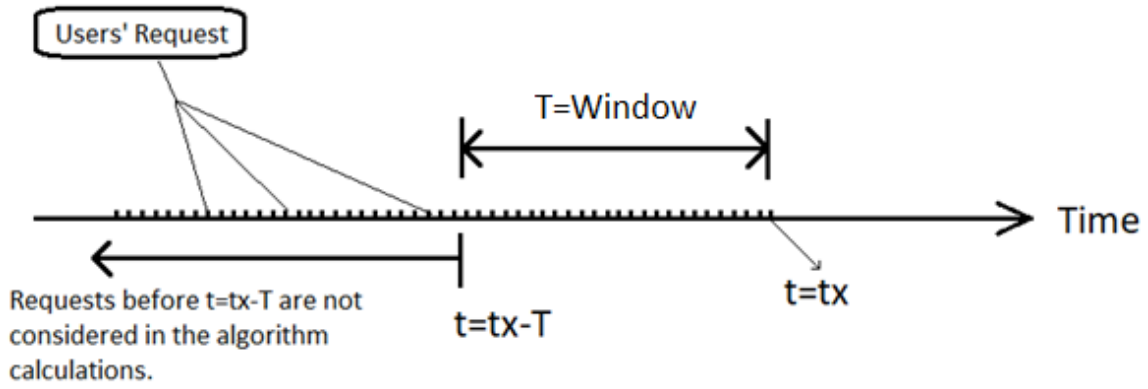


**Figure 3.2.** The Window used in algorithm calculation**.**

We define a ***learning time*** at the start during which the algorithm keeps track of the user requests and records of the decision parameters without making any replacement decision. It takes care of cold start in the learning process. We maintain learning time in our system equivalent to the arrival time of a set of user requests that do not cause cache replacement. The benefit of this learning time is to build a history of users' request and prevent replacing the important files as the initial counter values of all hits and misses are zeros. After the learning time, the algorithm builds a data set representative of users' interests that help in making one of the following decisions:

1- **Caching without replacement:** the system has enough space to cache and no need for cache replacement decision.
2- **Caching with replacement:** replace a cached file or more with the requested file
3- **File replication:** cache the same file in more than one helper.

**Caching without replacement:** When the system receives a request for a file that is not cached in any helper in the cluster, the MBS performs a check on whether one of the SBSs within the cluster has enough space to cache the requested file. It essentially finds an SBS in the cluster with largest free cache space. The file will be cached if there is enough space. In case that there is not enough space to cache the requested file, a replacement decision is made for the requested file.

**Caching with replacement:** Our proposed algorithm evicts up to two files to make a space in the cache for the requested file. Otherwise, it serves the user without making any replacement decision.

As mentioned before, the MBS keeps track of all misses and hits for all requested files in the last Window $W$. The algorithm searches for the candidate SBS, which is the SBS that has minimum total hits of all the cached files among all other SBSs within the same cluster. Let us define $H = \{h_1, h_2, h_3, \dots . h_M\}$ as a set of helpers within the cluster, and $h_i = \{f_1, f_2, f_3, \dots . f_n\}$ as a candidate SBS that caches files $f_1, f_2, f_3, \dots . f_n$, and $N = \{f_1, f_2, f_3, \dots . f_R\}$ as a set of files $f_1, f_2, f_3, \dots . f_R$ such that each $f_i \in N$ is a candidate file for replacement. Required space for caching is $S$ where:

$S = size\ of\ request\ file - free\ space\ of\ candidate\ SBS.$

The *individual candidacy conditions* for $f_i \in N$ that makes it a candidate for replacement are:

1. Number of hits for $f_i$ < number of misses for the requested file.
2. Size of $f_i$ >= $S$.

For each candidate file, a product of number of hits and file size is calculated and the file with minimum value of this product is selected for replacement.

To make sure that the file that is recently cached after replacing another file should not itself become candidate for replacement due to its low hit count, our algorithm assigns a weight to the recently cached file in the total hit field equal to the value of miss, such that the weight decays with time until it reaches zero after a period of time equal to W if there is no subsequent request.

In case that $N = \{\emptyset\}$ because there is no candidate file for replacement due to number of hits being greater than or equal to the number of misses of the requested file (violation of candidacy condition 1), then a miss for the requested file is counted and it is served to the user without being cached.

In case that $N = \{\emptyset\}$ because of cache size constraint (violation of candidacy condition 2), then a set of candidate files $N'$ is created based one the following *set candidacy conditions* on $f_i \in N'$ :

1. Number of hits on $f_i$ < number of misses on the requested file.
2. Size of $f_i$ < $S$.

After $N'$ is formed, a list of $i(i-1)/2$ possible pairs from candidate files is constructed. For example, if $N' = \{f_1, f_2, f_3, f_4\}$, the possible pair are $\{(f_1, f_2), (f_1, f_3), (f_1, f_4), (f_2, f_3), (f_2, f_4), (f_3, f_4)\}$. The candidate pairs are the pairs that their sum of hits is less than the miss value of requested file and the sum of their file size in addition to the free space of the cache is greater than the size of the requested file. For each candidate pair, a value is calculated based on multiplication of the sum of file size with the sum of file hits; for example, for a pair of $(f_y, f_z)$, this calculated value equal to $(size\ of\ f_y + size\ of\ f_z) * (no.of\ hits\ on\ f_y + no.of\ hits\ on\ f_z)$. The pair of files with minimum value is selected for replacement to cache the requested file. If

non-of the pairs meet the above conditions, the request file is not cached and only be served to the user.

The following example demonstrates our proposed algorithm. Figure 3.3 shows the network topology for the example. Let us consider receiving the requests in Table 3.1 after the learning time. Table 3.2 shows the information (traced in the last window $W$) that algorithm needs to make a decision. Each SBS has a cache with size of 40 MB:

**Table 3. 1. Users' request**.

| **User** | $k_1$ | $k_3$ | $k_6$ | $k_2$ |
|---|---|---|---|---|
| **File #** | $f_7$ | $f_5$ | $f_{20}$ | $f_{21}$ |
| **Size of file (MB)** | 18 | 13 | 9 | 16 |
| **File misses** | - | - | 2 | 4 |

**Table 3. 2. Cache status of the SBSs.**

| SBS 1 | | | SBS 2 | | | SBS 3 | | | SBS 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **File #** | Hits | Size | File # | Hits | Size | File # | Hits | Size | File # | Hits | Size |
| $f_1$ | 8 | 6 | $f_3$ | 7 | 8 | $f_9$ | 12 | 17 | $f_7$ | 16 | 18 |
| $f_4$ | 5 | 8 | $f_6$ | 5 | 9 | $f_5$ | 6 | 13 | $f_{11}$ | 3 | 9 |
| $f_8$ | 2 | 9 | $f_{12}$ | 2 | 5 | $f_{16}$ | 1 | 9 | $f_2$ | 3 | 8 |
| $f_{13}$ | 1 | 7 | $f_{14}$ | 1 | 8 | | | | | | |
| $f_{25}$ | 1 | 6 | $f_{17}$ | 1 | 7 | | | | | | |
| **Sum** | 17 | 36 MB | | 16 | 37 MB | | 19 | 39 MB | | 22 | 37 MB |

From Table 3.1, the first request made by user $k_1$ and for file $f_7$ that is cached in SBS 4 as shown in Table 3.2. User $k_1$ has no direct connection with SBS 4, since it has only one link with SBS 1 as shown in Figure 3.3.

In this case, the request is served by SBS 4 by transferring the file to user $k_1$ through SBS 1 and internal network, and a hit will be added for $f_7$, but a local miss will be counted on SBS 1 for $f_7$. Then, user $k_3$ requests file $f_5$ while it has two connections with SBS 2 and SBS 3. The user sends the request to the SBS with better link quality (main link), which is in this case SBS 2, without knowing whether the requested file is cache there. Since SBS 3 has $f_5$ in its cache, it serves the file directly to the user that experiences reduced delay. Next, user 6 requests file 20, and since it is not cached in any of the SBSs then the misses of file 20 will increase by 1 and become 3 misses. MBS

will deliver the file from the internet and will decide on whether to cache it or not and where to cache it.

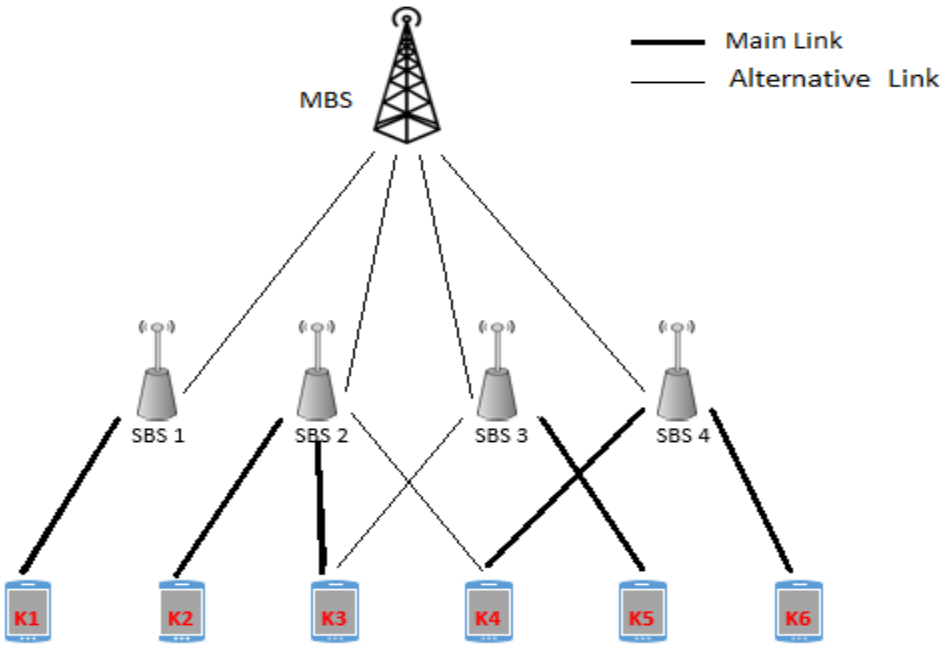


**Figure 3.3**. Network topology for the example.

Table 3.2 shows that the SBS2 is the candidate SBS because it has minimum hits. The algorithm forms the set of candidate files $N(f) = \{f_{12}, f_{14}, f_{17}\}$, as their hit counts are less than misses of $f_{20}$, as shown in Table 3.2. Since only $f_{14}$ and $f_{17}$ meet the size constraint, the set of candidate files is reduced to $N(f) = \{f_{14}, f_{17}\}$. Further, $f_{17}$ is chosen for replacement since both files have the same hit count and $f_{17}$ is the shortest file meeting the size constraint. File $f_{20}$ replaces file $f_{17}$ and its hit count is initialized to 3 as shown in Table 3.3.

**Table 3. 3. Updated state of SBSs' caches**

| SBS 1 | | | SBS 2 | | | SBS 3 | | | SBS 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **File #** | Hits | Size | File # | Hits | Size | File # | Hits | Size | File # | Hits | Size |
| $f_1$ | 10 | 6 | $f_3$ | 7 | 8 | $f_9$ | 12 | 17 | $f_7$ | 16+1 | 18 |
| $f_4$ | 5 | 8 | $f_6$ | 5 | 9 | $f_5$ | 6+1 | 13 | $f_{11}$ | 3 | 9 |
| $f_8$ | 2 | 9 | $f_{12}$ | 2 | 5 | $f_{16}$ | 1 | 9 | $f_2$ | 3 | 8 |
| $f_{13}$ | 1 | 7 | $f_{14}$ | 1 | 8 | | | | | | |
| $f_{25}$ | 1 | 6 | $f_{20}$ | 3 | 9 | | | | | | |
| **Sum** | 17 | 36 MB | | 18 | 39 MB | | 19+1 | 39 MB | | 22+1 | 37 MB |

Table 3.3 shows updated state of Table 3.2 after processing the above three requests. When user $k_2$ submits its request for file $f_{21}$, which is the last request in Table 3.1. Since $f_{21}$ is not found in any of the cache, the algorithm increases the miss count of $f_{21}$ from 4 to 5 and searches for the minimum space available to store $f_{21}$. There is no free space in any cache to store the file; hence, the algorithm looks for candidate SBS set using the conditions in equation (1) and finds SBS 1 has the minimum hit count. Among the files cached at SBS 1, $f_8$,$f_{13}$, and $f_{25}$ have hit counts less than the hit count of $f_{21}$but none of them is big enough to cache $f_{21}$; hence, there is no candidate file and $N(f) = \{\emptyset\}$ due to cache size constrain. In this case, the algorithm forms a new set of candidate files $N'(f) = \{ f_8, f_{13}, f_{25}\}$ to replace multiple files and the possible pairs for replacement are $(f_8, f_{13})$, $(f_8, f_{25})$, $(f_{13}, f_{25})$. The algorithm calculate the sum of (byte*hits) for these pairs as shown in Table 3.4, and the pair with minimum product (byte*hits) is the candidate pair for replacement.

**Table 3. 4. Pairs of candidate files with byte hits**.

| pairs | Sum of byte hits |
|---|---|
| $(f_8, f_{13})$ | 25 |
| $(f_8, f_{25})$ | 24 |
| $(f_{13}, f_{25})$ | 13 |

Table 3.5 shows the final status of SBSs' caches after four requests, where two files were served from the caches of SBSs and three files are replaced with two files.

**Table 3. 5. The final status of SBSs' caches.**

| **SBS 1** | | | **SBS 2** | | | **SBS 3** | | | **SBS 4** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **File #** | Hits | Size | File # | Hits | Size | File # | Hits | Size | File # | Hits | Size |
| $f_1$ | 10 | 6 | $f_3$ | 7 | 8 | $f_9$ | 12 | 17 | $f_7$ | 16+1 | 18 |
| $f_4$ | 5 | 8 | $f_6$ | 5 | 9 | $f_5$ | 6+1 | 13 | $f_{11}$ | 3 | 9 |
| $f_8$ | 2 | 9 | $f_{12}$ | 2 | 5 | $f_{16}$ | 1 | 9 | $f_2$ | 3 | 8 |
| $f_{21}$ | 5 | 16 | $f_{14}$ | 1 | 8 | | | | | | |
| | | | $f_{20}$ | 3 | 9 | | | | | | |
| **Sum** | 22 | 39 MB | | 18 | 39 MB | | 19+1 | 39 MB | | 22+1 | 37 MB |

**File replication:** Caching unique files within a cluster of SBSs increases the opportunity of serving more request from the cache that tends to minimize the backhaul traffic. However, when there is

a high demand for a few popular files, then the traffic on the SBS-SBS link from the SBSs that cache those files to the SBSs connected to the requesting user would be high causing users to experience increased delay. This shows the need for replicating most popular files in the caches of on more than one SBS under certain conditions to enhance the system performance. We consider a threshold user delay to decide about replicating a file in several caches within a cluster. The topology is simplified in Figure 3.4 to show different delay components: $D_b$, $D_h$ and $D_u$ are MSB backhaul link delay, SBS backhaul link delay and user-SBS link delay, respectively.

**Case1: SBS 1 and SBS 2 cache unique files.**

It is assumed that both SBSs cache only one file, SBS 1 caches file $f_1$ and SBS 2 caches file $f_2$. Consider $X_1$ users and $Y_1$ users request file $f_1$, and $X_2$ and $Y_2$ users request file $f_2$. The total delay for all these requests is $D_T$.

$$D_T = X_1 D_u + X_2 D_u + 2X_2 D_h + Y_1 D_u + Y_2 D_u + 2Y_2 D_h$$

$$D_T = (X_1 + X_2 + Y_1 + Y_2) D_u + (2X_2 + 2Y_1) D_h \qquad (1)$$

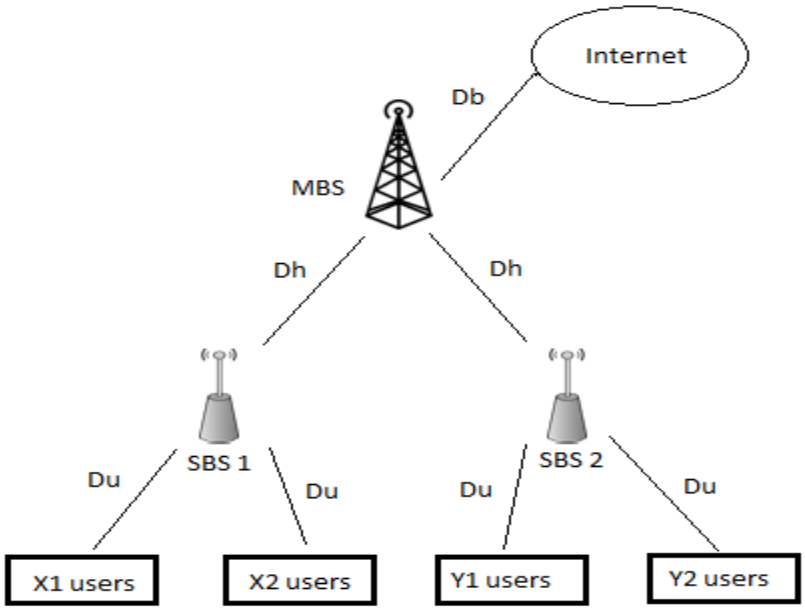


**Figure 3.4.** The delays in the cluster-based topology of small cell network.

**Case2: SBS 1 and SBS 2 cache same file.**

Both SBS 1 and 2 cache the same file say file $f_1$. As in case 1, $X_1$ and $Y_1$ users request file $f_1$ and $X_2$ and $Y_2$ users request file $f_2$. The total delay is calculated in the following equations.

$$D'_T = X_1D_u + X_2D_u + X_2D_h + X_2D_b + Y_1D_u + Y_2D_u + Y_2D_h + Y_2D_b$$

$$D'_T = (X_1 + X_2 + Y_1 + Y_2)D_u + (X_2 + Y_2)D_h + (X_2 + Y_2)D_b \qquad (2)$$

By comparing equation (1) from case1 and equation (2) from case2 and assume that $D'_T < D_T$

$$(X_2 + Y_2)D_h + (X_2 + Y_2)D_b < (2X_2 + 2Y_1)D_h$$

$$(X_2 + Y_2)D_b < (X_2 - Y_2 + 2Y_1)D_h \qquad (3)$$

We assume $X_1$ and $Y_1$ users generate less requests than $X_2$ and $Y_2$ users, then we can further assume $(X_2 - Y_2 + 2Y_1) \approx 2Y_1$ to simplify equation (3) as:

$$(X_2 + Y_2)D_b < 2Y_1 * D_h$$

$$\frac{D_b}{D_h} < \frac{2Y_1}{(X_2 + Y_2)} \qquad (4)$$

When users connected to one of the SBSs have a high demand for a file cached in the neighbor SBS, then $Y_1$ represents the local miss for that file, and $X_2 + Y_2$ represents the hits of the file with minimum hits and cached in the same SBS. For example, let us assume $D_b$=10, $D_h$=1, and SBS 1 has the following tables:

**Tables 3.6a. SBS cache status.**

| File | Hits | Size (MB) |
|---|---|---|
| f1 | 10 | 10 |
| f3 | 7 | 14 |
| f5 | 4 | 8 |
| f6 | 1 | 7 |

**Table 3.6b. SBS cache status.**

| File | Local miss | Size (MB) |
|---|---|---|
| f2 | 9 | 6 |
| f7 | 4 | 9 |
| f8 | 2 | 3 |
| f9 | 1 | 5 |

From the above tables, $X_2 + Y_2 = 1$ that are the hits of file $f_6$, and $Y_1$ = 9, which is the neighbor miss of file $f_2$. Thus, if equation (4) is TRUE then file $f_6$ can be replaced with file $f_2$ that is already cached in the other SBS. As the equation (4) is TRUE, we only have to check the available space for caching.

Size of file $f_2$ (6 MB) < Size of file $f_6$ (7 MB) + free space (1 MB).

Since both statements are True, file $f_2$ replaces file $f_6$ in SBS1.

# Chapter 4

# Simulation results

To evaluate the performance of the proposed work, we simulated our proposed Cluster-Based Cache Placement in 5G Network Model [1] and the Least Recently Used (LRU) cache replacement algorithm that we used as baseline for comparison. The simulation parameters and files request generation are discussed and then the simulation results are analyzed.

## 4.1 Network Simulation Parameters and Files Request Generation

In this simulation, using Matlab, a single cluster network with 10 helpers and 100 users are distributed within a range of 100 meter of MBS. Each helper is equipped with a cache storage, whose capacity ranges from 1% to 7% of the total size of the library of files containing 500 files. The file size ranges from 10-70 MB.

The user file requests are generated based on two different Zipf distributions. First Zipf distribution with skew parameter of $\alpha_1$ represents the files popularity of the library. The second Zipf distribution with skew parameter of $\alpha_2$ represents the users' request probability. If the user $k_1$with request probability equal to 0.1 and the file popularity of $f_1$ equal to 0.2, then it means 20% of all user generated requests will be for $f_1$ and most likely 10% of these requests are generated by user $k_1$. Thus, 2% of all requests are generated by user $k_1$ for file $f_1$.

To study the effect of users file request pattern on cache replacement algorithm, two request patterns are generated. The parameters of the first pattern based on the Zipf distribution are as follows: $\alpha_1 = 0.4$, $\alpha_2 = 0.2$, number of users $k = 100$, number of files $f = 500$, and number of generated requests $r = 10000$. Since top ranked files have high access probability, the tail of file request pattern consists of only top ranked files. For example, for a set of 6 files $\{f_1, f_2, f_3, f_4, f_5, f_6\}$ with file popularity $\{0.4, 0.25, 0.15, 0.1, 0.05, 0.05\}$, the number of requests for $f_1, f_2, \ldots\ f_6$ equals $\{8, 5, 3, 2, 1, 1\}$ in a stream of 20 requests (file requests $r = 20$). Using random selection of these files, the request pattern resembles the following order $\{f_1,\ f_3,\ f_6,\ f_2,\ f_1,\ f_5,\ f_4,\ f_3,\ f_2,\ f_1,\ f_3,\ f_4,\ f_2,\ f_1,\ f_2,\ f_1,\ f_2,\ f_1,\ f_1,\ f_1\}$. The second file request pattern uses the same values of the parameters of the first pattern except it shuffles file requests in a random order. For example, a

shuffle of the above request pattern could be. $\{f_1, f_3, f_2, f_1, f_4, f_2, f_1, f_3, f_2, f_1, f_4, f_1, f_2, f_1, f_2, f_6, f_1, f_3, f_1, f_5\}$. We calculated Euclidean distance $d$ between two probability vectors using the following formula as a measure of shift between their respective patterns.

$$d = \left|R_{f_i} - R'_{f_i}\right| \qquad (1)$$

Where $R_{f_i}$ is the rank of $f_i$ of the first vector, $R'_{f_i}$ is the file rank of the second vector, and $i = 1, 2, 3 \ldots F$.

File request probability $P_j$ for file j is the frequency of request for file $j$ in a user request pattern. Zipf probability $P_j$ of file $j$ is the probability of file $j$ in the popularity distribution that is used to assess the popularity ranking of files. Figure 4.1 shows popularity diversity for files *i, j, k.* which is represented by file rank and calculated using Euclidean distance.

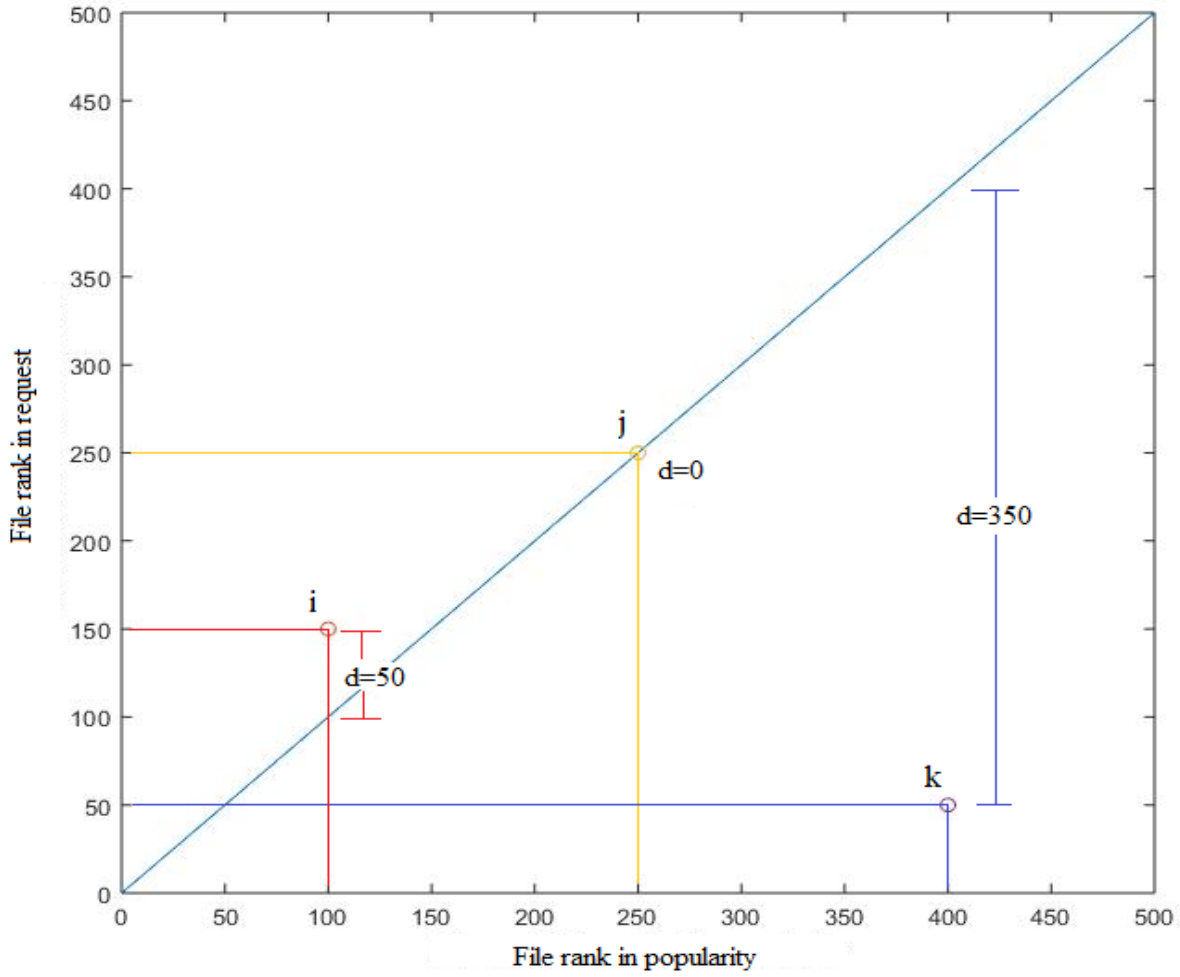


**Figure 4.1.** Diversity of file popularity and Euclidean distance.

Figure 4.1 shows that the request pattern of file *j* closely follows its popularity; hence, both its rank in file request pattern and popularity distribution are the same, which is indicated by d=0. The distance d=50 shows small divergence in request pattern from the file popularity of file *i*. The high divergence distance d=350 shows large number of requests (request rank 50) are generated for a less popular file *k* (popularity rank 400).

For all simulated algorithms, learning period is set to 1000 requests. In this period the algorithm tracks the hits and misses of the requested files without doing any replacement. For the proposed work, the algorithm keeps track of the hits and misses of the recent requests within a specific time known as a window size, and set to 1500 requests. Table 4.1 summaries the parameters used in the simulation.

**Table 4. 1. Simulation parameters.**

| Parameter | Value | Description |
|---|---|---|
| $K$ | 100 | Number of users. |
| $M$ | 10 | Number of helpers. |
| $C$ | $i * 200, i = 1,2, \ldots .7$ | Cache size of each helper. |
| $F$ | 500 | Library size. |
| $d$ | $(0, 25, 50, 125, 250)$ | Shift of file popularity distribution. |
| Learning time | 1000 | |
| Window time | 1500 | |
| SNR | 13 | Signal to Noise Ratio. |
| $\alpha_1$ | 0.4 | Zipf distribution parameter for file popularity |
| $\alpha_2$ | 0.2 | Zipf distribution parameter for user request |
| $r$ | 10000 | Number of generated requests |

## 4.2 Evaluation of Cache Placement Algorithm in Cluster-Based Cache Placement in 5G Network

Two performance metrics are used to evaluate the simulated algorithms, miss ratio and backhaul traffic. Due to difference in file sizes, backhaul traffic may not be directly proportional to miss ratio. The parameters shown in Table 4.1 were used to evaluate the Cluster-Based Cache Placement in 5G Network. We first develop understanding of the degradation of the performance of the cache

placement algorithm. For this case, the file request pattern follows probability distribution of the file Zipf popularity distribution that is used for cache placement assignment. We identify this request pattern with $d$ =0 that is the percentage of the requests generated for a file is the same as its Zipf probability. Table 4.2 shows the best-case performance of the cache placement algorithm when request pattern follows popularity distribution.

**Table 4. 2.** Evaluation of cache placement algorithm with file popularity distribution distance $\boldsymbol{d = 0}$.

| $d = 0$ | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|
| Miss ratio | 74.44% | 60% | 47.11% | 38.01% | 30.12% | 25.13% | 17.48% |
| Backhaul traffic (MB) | $3.44\times10^5$ | $2.99\times10^5$ | $2.53\times10^5$ | $2.2\times10^5$ | $1.8\times10^5$ | $1.56\times10^5$ | $1.12\times10^5$ |

It shows that as cache size increases, the miss ratio significantly decreases as well as the backhaul traffic decreases. However, we expect that whenever file request pattern differs from the file popularity distribution, such as unpopular files become more popular or vice versa, the caching system should experience performance degradation. Then, we measured the same performance metrics for the file request pattern with shift $d = 250$. Figures 4.2a and 4.2b compares the performance of the cache placement algorithm for the request pattern of shift $d = 250$ with the shift $d = 0$. The result expectedly shows 15% higher miss ratio for the cache size equal to 10%. The impact of the shift in the file popularity decreases with larger cache size. With 70% of cache size, the miss ratio increases by only 4%. The cache placement Algorithm is tested by varying the range of file popularity distances $d = \{25, 50, 125, 250\}$, and the results are listed in Appendix A. These results clearly demonstrate the need for a cache replacement scheme due to following reasons. In our system, cache placement decision is made at time $t = 0$ based on an estimate of file association with users measured as file popularity available at that time. As time progresses users generate requests for files that deviate from initial popularity estimate. This deviation results in increased miss ratio corresponding to requests for files that were not cached because they were deemed less popular at time $t = 0$. We conjecture that a cache replacement scheme that tracks this deviation in file popularity from its initial estimate and replace less popular files with more popular files may decrease the miss ratio and backhaul traffic. Notice every time a file is missed it is

downloaded through the backhaul link causing backhaul traffic. Hence, we expect that a smart cache replacement scheme will ensure that the changes in the file popularity will not affect the user experience.

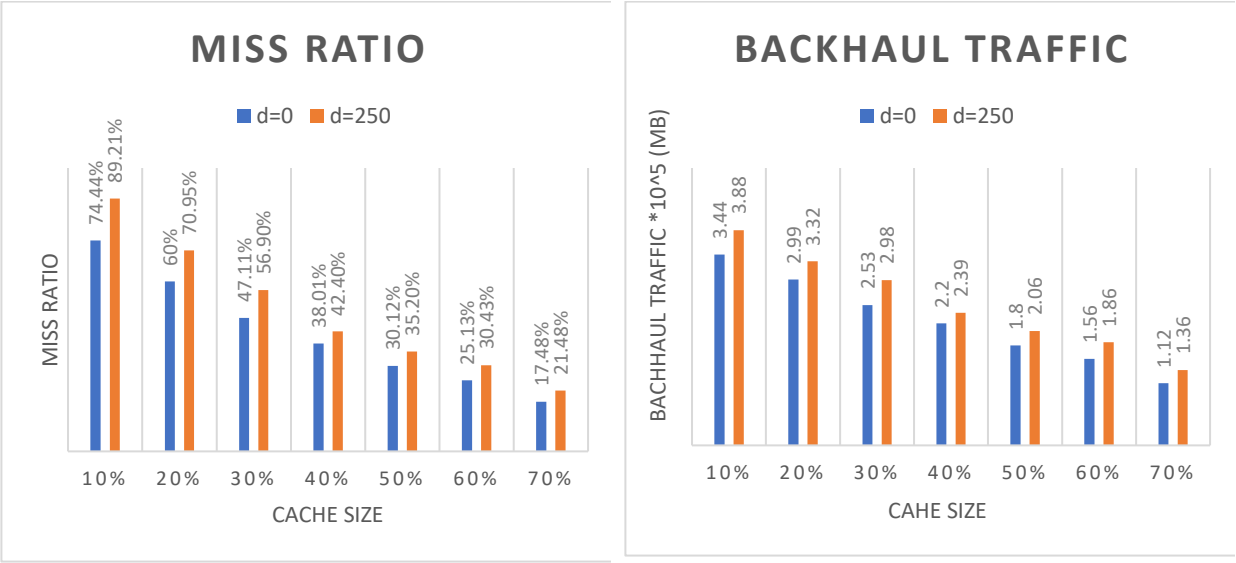


**Figure 4.2 a**, **Figure 4.2 b**. The performance metric of the cache placement with a shift of file popularity $\boldsymbol{d = 0}$, and $\boldsymbol{d = 250}$.

## 4.3 Performance Evaluation of LRU

Before we proceed to evaluate the performance of our proposed cache replacement scheme, we simulated LRU algorithm in a cluster-based mobile network model and evaluate its performance because LRU is widely used as a cache replacement technique. Three different configurations are set for LRU:

1- Replacement with one file: In this configuration, the LRU algorithm does the replacement only when evicting the least recently used file provides the needed space. This configuration is called ***LRU-1***.

2- Replacement with up to two files: In this configuration, Algorithm does the replacement by evicting up to two least recently used files if they free the needed space. This configuration is called ***LRU-2***.

3- Replacement with any number of files: LRU algorithm evicts the needed number of least recently used files. This configuration is called ***LRU-A***.

Comparing the LRU with these configurations with no shuffling of file requests and with a shift of file popularity distribution, *d* set equal to zero, ***LRU-1*** and ***LRU-2*** outperform ***LRU-A*** in terms

of miss ratio (lower miss ratio is better) in the range of cache size 30% to70%. However, all three configurations of LRU show similar performance is close for small cache sizes ranging from 10% to 20%. ***LRU-A*** and ***LRU-2*** show similar performance in terms of backhaul traffic and they outperform ***LRU-1***. Figures 4.3 and 4.4 show the performance comparison between the three LRU configurations in terms of miss ratio and backhaul traffic, respectively.

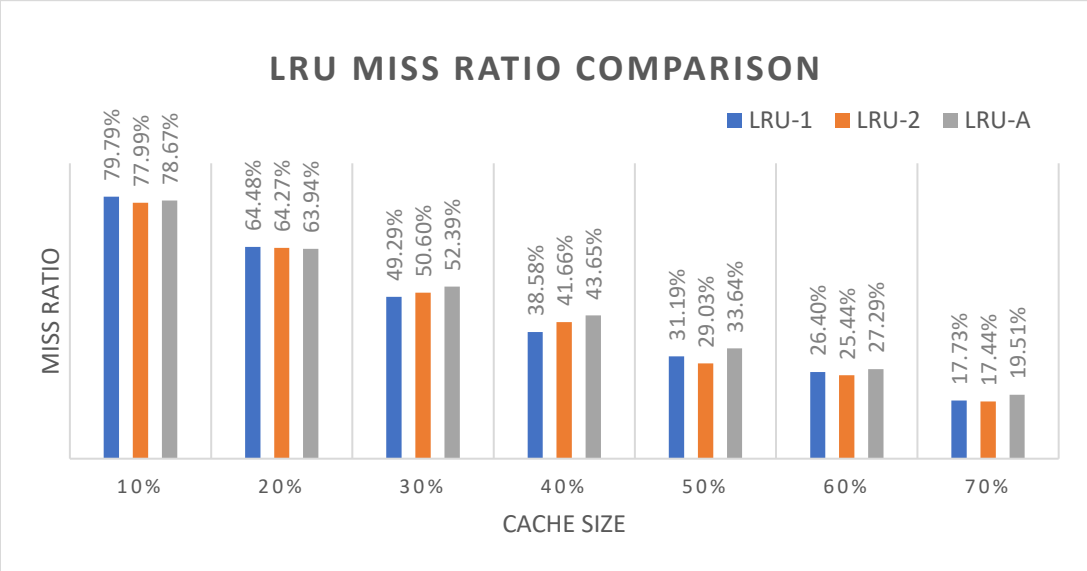


**Figure 4.3**. Miss ratio comparison between the three LRU configurations**.**

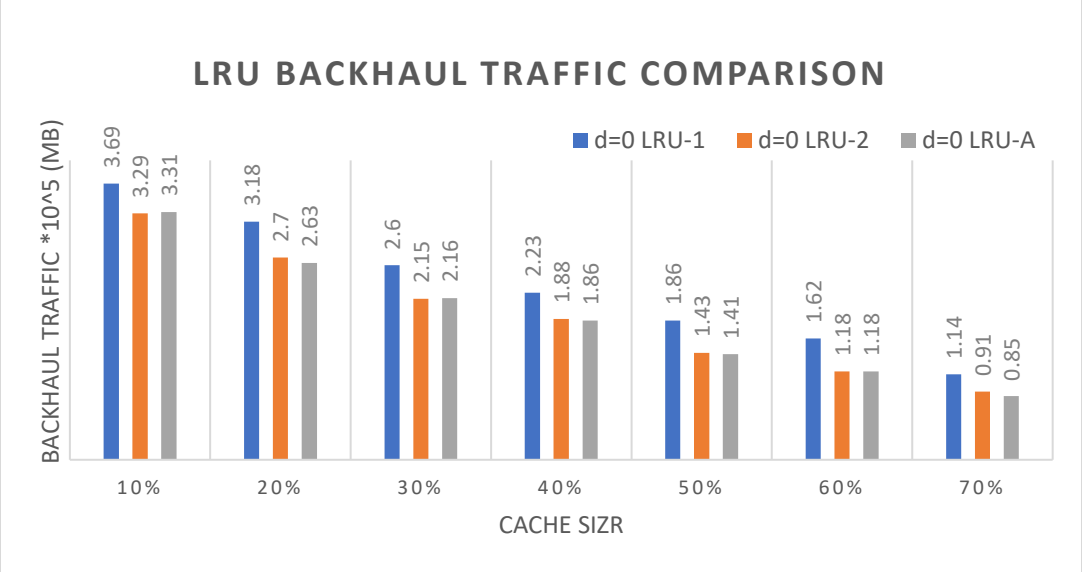


**Figure 4.4**. Backhaul traffic comparison between the three LRU configurations**.**

In addition to these key performance indicators, we also monitor the frequency of replacement decisions as a parameter that plays a role in the cache replacement design. Figure 4.5 demonstrates the number of replacements for the three LRU configurations.

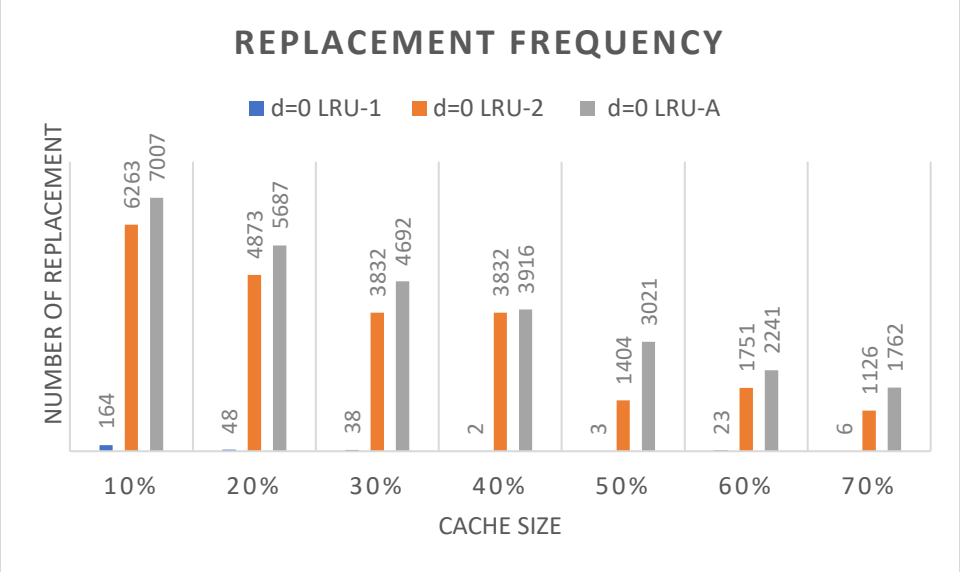


**Figure 4.5.** File replacement frequency of the three LRU configuration, $\boldsymbol{d = 0}$.

It becomes obvious from Figures 4.2-4.5 that ***LRU-1*** shows the worst performance in terms of backhaul traffic. It also shows backhaul traffic higher than the case where no replacement policy is applied after one time cache placement decision. This means that the few replacement decisions made by ***LRU-1*** causes high file misses and backhaul traffic. In contrast, ***LRU-2*** and ***LRU-A*** outperform the no cache replacement case in terms of backhaul traffic, but they process high numbers of file cache replacement, which raises practical issues. It is also observed that higher number of replacement decisions lead to the set of files in the cache that is totally different from the initial set stored after the placement decision.

## 4.4 Effect of File Request Pattern on Cache Replacement Decisions

We also studied the effect of request patterns by comparing two file request patterns: (i) Non-shuffled requests, and (ii) Shuffled requests. The result shows that file request patterns have a significant effect on the performance of the cache replacement algorithms. LRU-2 and LRU-A exhibit higher miss ratio and backhaul traffic when the request is shuffled as compared to non-shuffled request pattern. For example, there is a 10% difference in miss ratio and $3.7 \times 10\text{^}4$ (MB) in backhaul traffic between the two request patterns for the cache size of 10%. Figures 4.6a

and 4.6b show the effect of file request pattern on LRU-A in terms of miss ratio and backhaul traffic. Appendix A. shows more LRU-1, LRU-2 and LRU-A results.

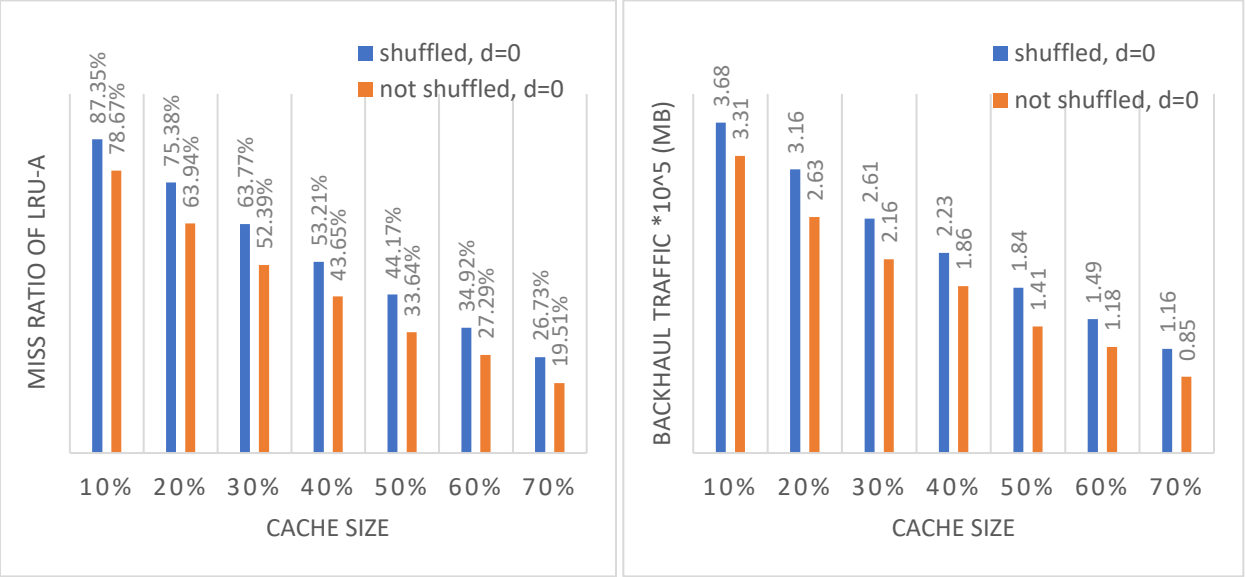


**Figure 4.6a, 4.6b**. File request pattern effect on LRU-A in terms of miss ratio and backhaul traffic.

### 4.5 Cluster-Based Cache Replacement Algorithm (CBR)

We observed from the results in section 4.4 that ***LRU-A*** shows better performance over ***LRU-1*** and ***LRU-2*** in term of saving the backhaul traffic when the request is not shuffled and the shift in distance of the file popularity distribution is set to zero. However, ***LRU-1*** can be excluded from the comparison due to the very low number of replacements compared to the number of users' requests. ***LRU-2*** slightly outperforms ***LRU-A*** when requests are shuffled and the distance *d* in file popularity distribution is set to zero. Hence, we consider ***LRU-2*** as the baseline case for performance comparison with ***CBR*** replacement algorithms for the shuffled request pattern.

***CBR*** replacement algorithm significantly outperforms ***LRU-2*** in terms of both miss ratio and backhaul traffic, as shown in Figure 4.7a and 4.7b, for small cache sizes ranging from 10% to 30%. Further, CBR achieves this superior performance with less number of file replacements as compared to ***LRU-2*** as shown in Figure 4.8.

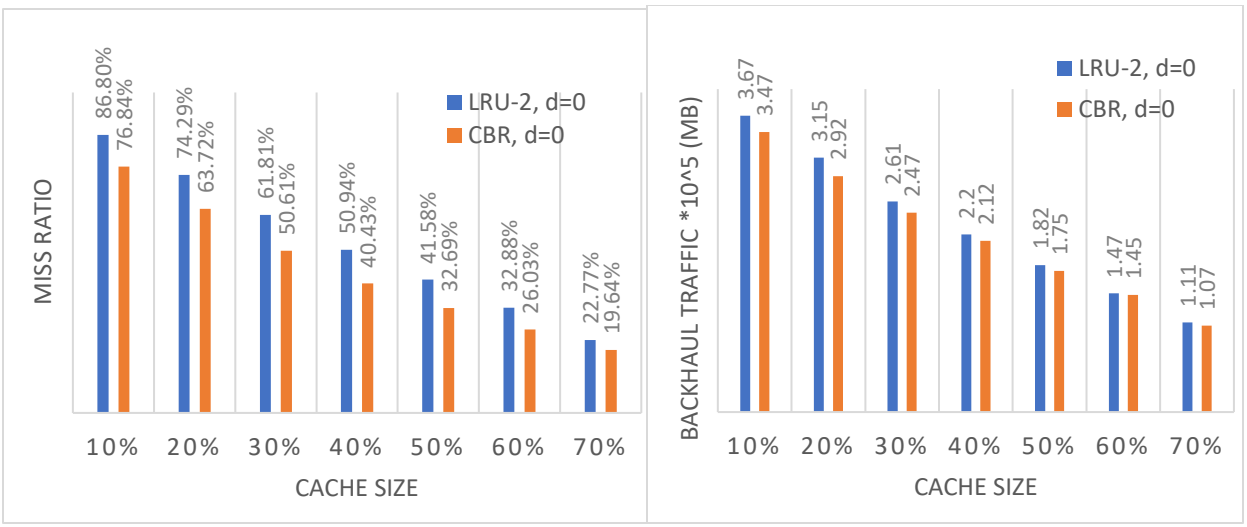


**Figure 4.7a, 4.7b.** Performance comparison between ***LRU-2*** and ***CBR***.

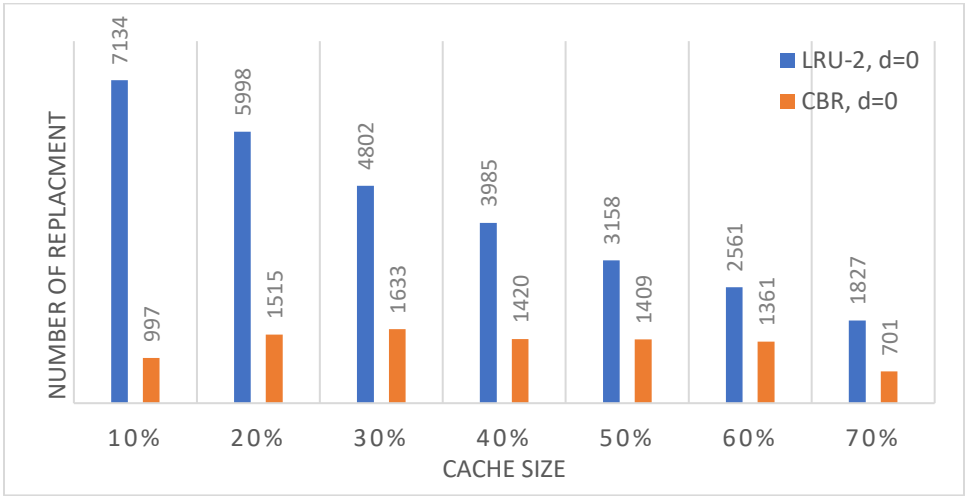


**Figure 4.8.** Number of replacement for ***LRU-2*** and ***CBR***.

In terms of number of cache files, a comparison between the initial and final state is made between ***CBR*** and ***LRU-2*** as shown in Table 4.3.

**Table 4. 3. Number of cached file in *CBR* and *LRU-2*.**

| Cache size | Number of files in the initial state | Number of file after running ***CBR*** (% no. of cached files to initial state) | Number of file after running ***LRU-2*** (% no. of cached files to initial state) |
|---|---|---|---|
| 10% | 74 | 76 (102.7%) | 44 (59.4%) |
| 20% | 158 | 133 (84.1%) | 99 (62.6%) |
| 30% | 230 | 195 (84.7%) | 149 (64.7%) |
| 40% | 282 | 254 (90%) | 208 (73.7%) |

Table 4.3 shows that **CBR** caches almost 20% more files than ***LRU-2*** when cache size is 40% and this percentage increases for smaller cache sizes. In summary, our proposed CBR scheme outperforms LRU in terms of miss ratio and backhaul traffic. This superior performance it achieves by fewer replacement decisions and storing more files in the cache.

## 4.6 CBR and File Replication.

It is obvious that caching the same file in multiple helpers increases the miss ratio and backhaul traffic of the cluster network. However, if there is a high demand for a few popular files, then the traffic on the SBS-SBS link from the SBSs that cache those files to the SBSs connected to the requesting user would be high causing users to experience increased delay. File replication in this case might decrease the delay experienced by users. Three parameters have an affect of file replication decision, the ratio of backhaul link delay $D_b$ to SBS-MBS link delay $D_h$, number of user's link to helpers, and file popularity parameter $\alpha$.

The higher value of $D_b/D_h$, the more chance for file replication. Users chance to be served from neighbor helper is higher when fewer links connect them to the network, this increase the local miss of popular files and make replication possible. We control the number of user's serving links by changing the SNR threshold, where user's serving links have SNR above the threshold.

We assume $D_b = 10$, $D_h = 1$, $\alpha = 0.4, 0.6, 0.9$, and we set $SNR\ threshold = 15, 19$.

1) $\alpha = 0.4$, $SNR\ threshold = 15, 19$.

Results show no file replication in the SBS.

2) $\alpha = 0.6$, $SNR\ threshold = 15, 19$.

Results show only one file replication when $threshold = 19$.

3) $\alpha = 0.9$, $SNR\ threshold = 15, 19$.

Results show that SNR threshold affects the number of replicated file in the network, Figure 4.9 shows a comparison of file replication under the two SNR threshold values.

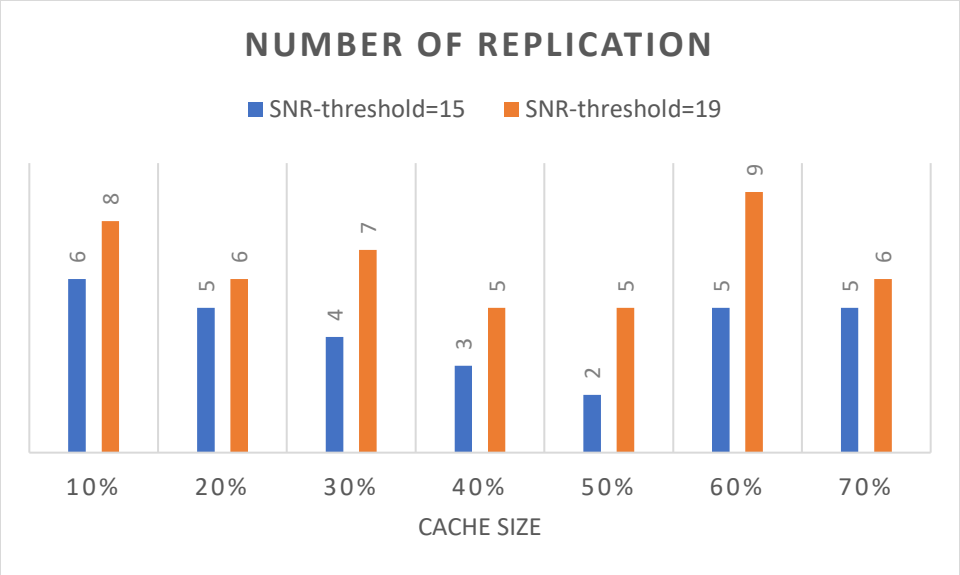


**Figure 4.9**. File replication comparison.

To study the benefit of file replication, we calculate the total delay experienced by users using the following parameters: $D_b = 5$, $D_h = 1$, $\alpha = 0.9$ and $SNR\ threshold\ =\ 15$.

We found that the total delay experienced by user in case of file replication is slightly less than the case of no file replication as shown in Figure 4.10. However, file replication reduces up to 25% of traffic on SBS-MBS links that experiences the highest traffic among all other SBS-MBS links.

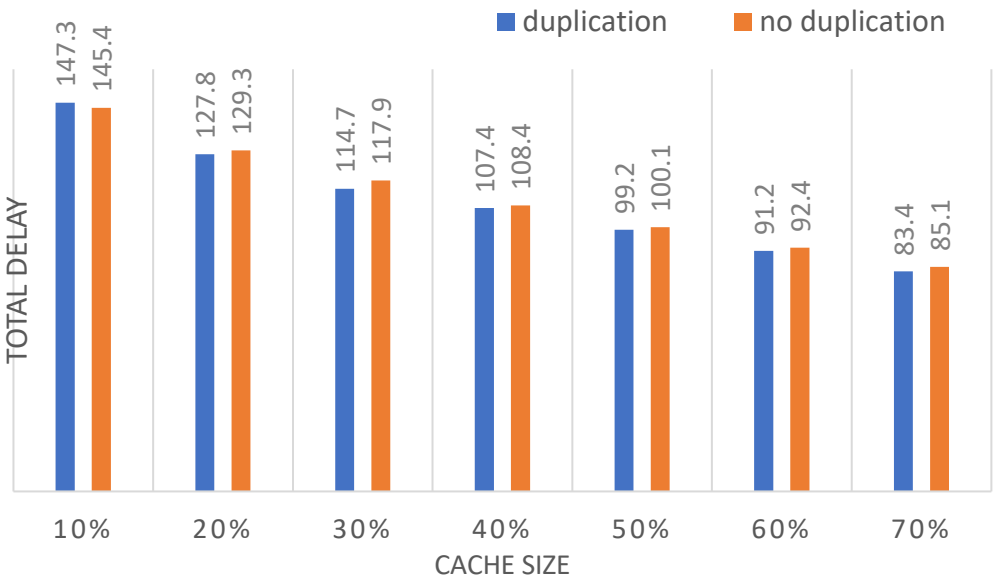


**Figure 4.10.** Total delay experienced by users.

# Chapter 5

# Conclusion

In this study, we simulated the proposed Cluster-Based Cache Placement in 5G Network Model and study performance degradation cause by the change in file request pattern, when it differs from the file popularity distribution, such as unpopular files become more popular or vice versa. The result shows 15% higher miss ratio for the cache size equal to 10%.

We also simulate LRU algorithm in a cluster-based mobile network model and evaluate its performance because LRU is widely used as a cache replacement technique. Three different configurations are set for LRU: (i) Replacement with one file (***LRU-1***). (ii) Replacement with up to two files (***LRU-2***). (iii) Replacement with any number of files (***LRU-A***). Then, we study the file request order on the three LRU configuration and we set two file request patterns: (i) not shuffled requests, (ii) shuffled requests. Result shows that file request pattern has a significant effect on the cache replacement algorithms and ***LRU-2*** outperforms the others LRU configurations when requests are shuffled and the distance $d$ in file popularity distribution is set to zero.

We proposed a Cluster-Based Cache Replacement algorithm ***CBR***. We consider ***LRU-2*** as the baseline case for performance comparison with ***CBR*** replacement algorithms for the shuffled request pattern. ***CBR*** replacement algorithm significantly outperforms ***LRU-2*** in terms of both miss ratio and backhaul traffic for small cache sizes ranging from 10% to 30%. Result shows that **CBR** caches almost 20% more files than ***LRU-2*** when cache size is 40% and this percentage increases for smaller cache sizes.

We presented in this thesis a heuristic algorithm that establishes the benefit of replacement. In future work, there is a need to develop more intelligent replacement decisions based on file utility in the network. Further, a more thorough simulation study can be done by varying the size of epoch and its impact on the traffic considering placement traffic in the network

# Appendix A

| Cluster-Based Cache Placement | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 74.44% | 60% | 47.11% | 38.01% | 30.12% | 25.13% | 17.48% |
| | Backhaul traffic | 3.44 | 2.99 * 10^5 | 2.53 * 10^5 | 2.2 * 10^5 | 1.8 * 10^5 | 1.56 * 10^5 | 1.12 * 10^5 |
| d=25 | Miss ratio | 77.83% | 60.25% | 48.44% | 38.86% | 32.25% | 26.2% | 17.34% |
| | Backhaul traffic | 3.63 * 10^5 | 3 * 10^5 | 2.61 * 10^5 | 2.25 * 10^5 | 1.94 * 10^5 | 1.63 * 10^5 | 1.11 * 10^5 |
| d=50 | Miss ratio | 78.14% | 62.26% | 48.99% | 40.63% | 31.56% | 26.6% | 16.33% |
| | Backhaul traffic | 3.62 * 10^5 | 3.14 * 10^5 | 2.618 * 10^5 | 2.37 * 10^5 | 1.88 * 10^5 | 1.65 * 10^5 | 1.05 * 10^5 |
| d=125 | Miss ratio | 83.7% | 64.73% | 51.64% | 43.84% | 31.64% | 25.81% | 19.1% |
| | Backhaul traffic | 3.77 * 10^5 | 3.15 * 10^5 | 2.73 * 10^5 | 2.51 * 10^5 | 1.89 * 10^5 | 1.59 * 10^5 | 1.23 * 10^5 |
| d=250 | Miss ratio | 89.21% | 70.95% | 56.9% | 42.4% | 35.2% | 30.43% | 21.48% |
| | Backhaul traffic | 3.88 * 10^5 | 3.32 * 10^5 | 2.98 * 10^5 | 2.39 * 10^5 | 2.06 * 10^5 | 1.86 * 10^5 | 1.36 * 10^5 |

| LRU-1, File requests are not shuffled | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 79.79% | 64.48% | 49.29% | 38.58% | 31.19% | 26.40% | 17.73% |
| | Backhaul traffic | 3.69 * 10^5 | 3.18 * 10^5 | 2.6 * 10^5 | 2.23 * 10^5 | 1.86 * 10^5 | 1.62 * 10^5 | 1.14 * 10^5 |
| d=25 | Miss ratio | 83.09% | 66.79% | 49.56% | 39.51% | 32.76% | 26.08% | 17.55% |
| | Backhaul traffic | 3.87 * 10^5 | 3.27 * 10^5 | 2.64 * 10^5 | 2.28 * 10^5 | 1.96 * 10^5 | 1.62 * 10^5 | 1.12 * 10^5 |
| d=50 | Miss ratio | 81.02% | 64.99% | 51.04% | 41.58% | 31.75% | 27.35% | 17.02% |
| | Backhaul traffic | 3.75 * 10^5 | 3.2 * 10^5 | 2.68 * 10^5 | 2.4 * 10^5 | 1.89 * 10^5 | 1.68 * 10^5 | 1.08 * 10^5 |
| d=125 | Miss ratio | 79.73% | 67.17% | 52.52% | 44.6% | 32.44% | 26.99% | 21.43% |
| | Backhaul traffic | 3.7 * 10^5 | 3.24 * 10^5 | 2.76 * 10^5 | 2.54 * 10^5 | 1.92 * 10^5 | 1.63 * 10^5 | 1.35 * 10^5 |
| d=250 | Miss ratio | 80.23% | 68.54% | 55.51% | 42.12% | 35.42% | 30.19% | 21.43% |
| | Backhaul traffic | 3.74 * 10^5 | 3.26 * 10^5 | 2.92 * 10^5 | 2.37 * 10^5 | 2.06 * 10^5 | 1.83 * 10^5 | 1.35 * 10^5 |

| LRU-2, File requests are not shuffled | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 77.99% | 64.27% | 50.6% | 41.66% | 29.03% | 25.44% | 17.44% |
| | Backhaul traffic | 3.29 ∗ 10^5 | 2.7 ∗ 10^5 | 2.15 ∗ 10^5 | 1.88 ∗ 10^5 | 1.43 ∗ 10^5 | 1.18 ∗ 10^5 | 9.14 ∗ 10^4 |
| d=25 | Miss ratio | 78.0% | 64.34% | 51.79% | 41.73 | 33.58% | 26.66% | 18.78% |
| | Backhaul traffic | 3.29 ∗ 10^5 | 2.71 ∗ 10^5 | 2.2 ∗ 10^5 | 1.86 ∗ 10^5 | 1.47 ∗ 10^5 | 1.21 ∗ 10^5 | 8.97 ∗ 10^4 |
| d=50 | Miss ratio | 76.92% | 63.83% | 50.42% | 41.49% | 32.11% | 24.76% | 17.14% |
| | Backhaul traffic | 3.25 ∗ 10^5 | 2.69 ∗ 10^5 | 2.13 ∗ 10^5 | 1.85 ∗ 10^5 | 1.43 ∗ 10^5 | 1.15 ∗ 10^5 | 8.53 ∗ 10^4 |
| d=125 | Miss ratio | 76.66% | 62.36% | 50.21% | 39.81% | 32.01% | 24.65 | 17.07% |
| | Backhaul traffic | 3.26 ∗ 10^5 | 2.6 ∗ 10^5 | 2.13 ∗ 10^5 | 1.82 ∗ 10^5 | 1.41 ∗ 10^5 | 1.12 ∗ 10^5 | 8.5 ∗ 10^4 |
| d=250 | Miss ratio | 78.38% | 62.6% | 49.42% | 40.55% | 31.67% | 23.79% | 16.69% |
| | Backhaul traffic | 3.33 ∗ 10^5 | 2.6 ∗ 10^5 | 2.1 ∗ 10^5 | 1.80 ∗ 10^5 | 1.39 ∗ 10^5 | 1.14 ∗ 10^5 | 8.71 ∗ 10^4 |

| LRU-A, File requests are not shuffled | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 78.67% | 63.94%% | 52.39% | 43.65% | 33.64% | 27.29% | 19.51% |
| | Backhaul traffic | 3.31 ∗ 10^5 | 2.63 ∗ 10^5 | 2.16 ∗ 10^5 | 1.86 ∗ 10^5 | 1.41 ∗ 10^5 | 1.18 ∗ 10^5 | 8.53 ∗ 10^4 |
| d=25 | Miss ratio | 78.54% | 65.51% | 53..23% | 43.7% | 34.93% | 27.95% | 20.26% |
| | Backhaul traffic | 3.29 ∗ 10^5 | 2.72 ∗ 10^5 | 2.2 ∗ 10^5 | 1.85 ∗ 10^5 | 1.47 ∗ 10^5 | 1.21 ∗ 10^5 | 8.84 ∗ 10^4 |
| d=50 | Miss ratio | 77.52% | 63.89% | 52% | 42.43% | 33.48% | 26.08% | 18.43% |
| | Backhaul traffic | 3.26 ∗ 10^5 | 2.64 ∗ 10^5 | 2.13 ∗ 10^5 | 1.78 ∗ 10^5 | 1.42 ∗ 10^5 | 1.13 ∗ 10^5 | 8.03 ∗ 10^5 |
| d=125 | Miss ratio | 77.5% | 63.57% | 51.56% | 42.71% | 33.24% | 26.1% | 19.17% |
| | Backhaul traffic | 3.28 ∗ 10^5 | 2.61 ∗ 10^5 | 2.12 ∗ 10^5 | 1.82 ∗ 10^5 | 1.45 ∗ 10^5 | 1.12 ∗ 10^5 | 8.33 ∗ 10^4 |
| d=250 | Miss ratio | 77.84% | 63.46% | 51.32% | 41.98% | 33.24% | 26.7% | 19.53% |
| | Backhaul traffic | 3.28 ∗ 10^5 | 2.6 ∗ 10^5 | 2.1 ∗ 10^5 | 1.77 ∗ 10^5 | 1.39 ∗ 10^5 | 1.15 ∗ 10^5 | 8.62 ∗ 10^4 |

| LRU-1, File requests are shuffled | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 75.36% | 60.21% | 47.17% | 38.45% | 30.51% | 24.64% | 17.69% |
| | Backhaul traffic | 3.52 ∗ 10^5 | 3.0 ∗ 10^5 | 2.53 ∗ 10^5 | 2.21 ∗ 10^5 | 1.8 ∗ 10^5 | 1.52 ∗ 10^5 | 1.13 ∗ 10^5 |
| d=25 | Miss ratio | 79.29% | 61.03% | 48.84% | 38.81% | 32.35% | 24.82% | 17.52% |
| | Backhaul traffic | 3.7 ∗ 10^5 | 3.03 ∗ 10^5 | 2.61 ∗ 10^5 | 2.25 ∗ 10^5 | 1.92 ∗ 10^5 | 1.53 ∗ 10^5 | 1.12 ∗ 10^5 |
| d=50 | Miss ratio | 77.43% | 63.01% | 48.91% | 40.9% | 31.89% | 26.23% | 16.28% |
| | Backhaul traffic | 3.62 ∗ 10^5 | 3.13 ∗ 10^5 | 2.61 ∗ 10^5 | 2.36 ∗ 10^5 | 1.89 ∗ 10^5 | 1.62 ∗ 10^5 | 1.04 ∗ 10^5 |
| d=125 | Miss ratio | 79.03% | 64.53% | 51.43% | 43.61% | 31.71% | 25.31% | 18.99% |
| | Backhaul traffic | 3.67 ∗ 10^5 | 3.15 ∗ 10^5 | 2.71 ∗ 10^5 | 2.49 ∗ 10^5 | 1.88 ∗ 10^5 | 1.56 ∗ 10^5 | 1.22 ∗ 10^5 |
| d=250 | Miss ratio | 81.56% | 66.29% | 56.07% | 41.83% | 35.06% | 29.7% | 21.38% |
| | Backhaul traffic | 3.76 ∗ 10^5 | 3.2 ∗ 10^5 | 2.95 ∗ 10^5 | 2.36 ∗ 10^5 | 2.05 ∗ 10^5 | 1.8 ∗ 10^5 | 1.35 ∗ 10^5 |

| LRU-2, File requests are shuffled | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 86.8% | 74.29% | 61.81% | 50.94% | 41.58% | 32.88% | 22.77% |
| | Backhaul traffic | 3.67 ∗ 10^5 | 3.15 ∗ 10^5 | 2.61 ∗ 10^5 | 2.2 ∗ 10^5 | 1.82 ∗ 10^5 | 1.47 ∗ 10^5 | 1.11 ∗ 10^5 |
| d=25 | Miss ratio | 86.97% | 74.47% | 61.75% | 50.67% | 41.38% | 33.72% | 22.82% |
| | Backhaul traffic | 3.71 ∗ 10^5 | 3.15 ∗ 10^5 | 2.61 ∗ 10^5 | 2.25 ∗ 10^5 | 1.85 ∗ 10^5 | 1.51 ∗ 10^5 | 1.11 ∗ 10^5 |
| d=50 | Miss ratio | 87.17% | 74.99% | 62.63% | 52.72% | 41.76% | 32.78% | 21.75% |
| | Backhaul traffic | 3.69 ∗ 10^5 | 3.16 ∗ 10^5 | 2.62 ∗ 10^5 | 2.3 ∗ 10^5 | 1.81 ∗ 10^5 | 1.5 ∗ 10^5 | 1.09 ∗ 10^5 |
| d=125 | Miss ratio | 87.17% | 73.81% | 61.35% | 52.28% | 40.84% | 32.48% | 23.49% |
| | Backhaul traffic | 3.68 ∗ 10^5 | 3.06 ∗ 10^5 | 2.58 ∗ 10^5 | 2.3 ∗ 10^5 | 1.79 ∗ 10^5 | 1.48 ∗ 10^5 | 1.12 ∗ 10^5 |
| d=250 | Miss ratio | 88.66% | 74.58% | 63.12% | 51.9% | 42.42% | 34.46% | 25.42% |
| | Backhaul traffic | 3.73 ∗ 10^5 | 3.1 ∗ 10^5 | 2.67 ∗ 10^5 | 2.23 ∗ 10^5 | 1.84 ∗ 10^5 | 1.57 ∗ 10^5 | 1.19 ∗ 10^5 |

| LRU-A, File requests are shuffled | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 87.35% | 75.38% | 63.77% | 53.21% | 44.17% | 34.92% | 25.75% |
| | Backhaul traffic | 3.68 ∗ 10^5 | 3.16 ∗ 10^5 | 2.61 ∗ 10^5 | 2.23 ∗ 10^5 | 1.84 ∗ 10^5 | 1.49 ∗ 10^5 | 1.10 ∗ 10^5 |
| d=25 | Miss ratio | 86.69% | 75.77% | 63.83% | 54.05% | 44.44% | 35.61% | 26.17% |
| | Backhaul traffic | 3.73 ∗ 10^5 | 3.16 ∗ 10^5 | 2.64 ∗ 10^5 | 2.27 ∗ 10^5 | 1.85 ∗ 10^5 | 1.52 ∗ 10^5 | 1.11 ∗ 10^5 |
| d=50 | Miss ratio | 87.74% | 75.82% | 64.72% | 55.15 | 44.02% | 36.19% | 26.47% |
| | Backhaul traffic | 3.69 ∗ 10^5 | 3.15 ∗ 10^5 | 2.66 ∗ 10^5 | 2.31 ∗ 10^5 | 1.82 ∗ 10^5 | 1.52 ∗ 10^5 | 1.13 ∗ 10^5 |
| d=125 | Miss ratio | 87.86% | 75.08% | 63.47% | 54.58% | 43.07% | 34.39% | 25.43% |
| | Backhaul traffic | 3.7 ∗ 10^5 | 3.08 ∗ 10^5 | 2.6 ∗ 10^5 | 2.31 ∗ 10^5 | 1.79 ∗ 10^5 | 1.46 ∗ 10^5 | 1.09 ∗ 10^5 |
| d=250 | Miss ratio | 89.43% | 76.06% | 65.28% | 54.53% | 44.44% | 36.23% | 27.02% |
| | Backhaul traffic | 3.75 ∗ 10^5 | 3.12 ∗ 10^5 | 2.68 ∗ 10^5 | 2.26 ∗ 10^5 | 1.85 ∗ 10^5 | 1.56 ∗ 10^5 | 1.18 ∗ 10^5 |

| CRB, File requests are shuffled | | Cache size 10% | Cache size 20% | Cache size 30% | Cache size 40% | Cache size 50% | Cache size 60% | Cache size 70% |
|---|---|---|---|---|---|---|---|---|
| d=0 | Miss ratio | 76.84% | 63.72% | 50.61% | 40.43% | 32.69% | 26.03% | 19.64% |
| | Backhaul traffic | 3.47 ∗ 10^5 | 2.92 ∗ 10^5 | 2.47 ∗ 10^5 | 2.12 ∗ 10^5 | 1.75 ∗ 10^5 | 1.45 ∗ 10^5 | 1.07 ∗ 10^5 |
| d=25 | Miss ratio | 77.78% | 65.17% | 51.58% | 42.08% | 33.79% | 27.33% | 19.09% |
| | Backhaul traffic | 3.42 ∗ 10^5 | 2.91 ∗ 10^5 | 2.44 ∗ 10^5 | 2.13 ∗ 10^5 | 1.73 ∗ 10^5 | 1.44 ∗ 10^5 | 1.08 ∗ 10^5 |
| d=50 | Miss ratio | 76.26% | 63.99% | 52.16% | 41.44% | 34.28% | 26.57% | 18.47% |
| | Backhaul traffic | 3.41 ∗ 10^5 | 2.89 ∗ 10^5 | 2.45 ∗ 10^5 | 2.1 ∗ 10^5 | 1.69 ∗ 10^5 | 1.41 ∗ 10^5 | 1.04 ∗ 10^5 |
| d=125 | Miss ratio | 78.37% | 65.29% | 52.48% | 42.29% | 33.45% | 26.58% | 19.64% |
| | Backhaul traffic | 3.41 ∗ 10^5 | 2.87 ∗ 10^5 | 2.41 ∗ 10^5 | 2.14 ∗ 10^5 | 1.69 ∗ 10^5 | 1.45 ∗ 10^5 | 1.07 ∗ 10^5 |
| d=250 | Miss ratio | 80.92% | 67.61% | 56.9% | 44.03% | 33.11% | 29.26% | 20.97% |
| | Backhaul traffic | 3.5 ∗ 10^5 | 2.91 ∗ 10^5 | 2.48 ∗ 10^5 | 2.13 ∗ 10^5 | 1.79 ∗ 10^5 | 1.50 ∗ 10^5 | 1.15 ∗ 10^5 |

# References


[1] Shadi Sadeghpour, Muhammad Jaseemuddin, Saleh Kharbish. “Cluster-based Cache Placement in 5G network”, accepted to appear in the proc. of the 20th IEEE International Conference on High Performance Computing and Communications, 2018.

[2] Golrezaei, Negin, Karthikeyan Shanmugam, Alexandros G. Dimakis, Andreas F. Molisch, and Giuseppe Caire. "FemtoCaching: Wireless video content delivery through distributed caching helpers." *2012 Proceedings IEEE INFOCOM*, 2012. doi:10.1109/infcom.2012.6195469

[3] Khreishah, Abdallah, Jacob Chakareski, and Ammar Gharaibeh. "Joint Caching, Routing, and Channel Assignment for Collaborative Small-Cell Cellular Networks." *IEEE Journal on Selected Areas in Communications* 34, no. 8 (2016), 2275-2284. doi:10.1109/jsac.2016.2577199.

[4] Bastug, Ejder, Mehdi Bennis, and Merouane Debbah. "Social and spatial proactive caching for mobile data offloading." *2014 IEEE International Conference on Communications Workshops (ICC)*, 2014. doi:10.1109/iccw.2014.6881261.

[5] Mumtaz, S., Rodriguez, J., & Dai, L. (2017). *MmWave massive MIMO: A paradigm for 5G*. London, United Kingdom: Academic Press is an imprint of Elsevier.

[6] Baştuğ, Ejder, Mehdi Bennis, and Mérouane Debbah. "Proactive Caching in 5G Small Cell Networks." *Towards 5G*, 2016, 78-98. doi:10.1002/9781118979846.ch6.

[7] Osseiran, A., Monserrat, J. F., & Marsch, P. (2016). *5G mobile and wireless communications technology*. Cambridge: Cambridge University Press.

[8] Ezz Hattab, Sami Qawasmeh. “A Survey of Replacement Policies for Mobile Web Caching.” *2015 International Conference on Developments of E-Systems Engineering*, 2015. Doi:10.1109/DeSE.2015.13

[9] Wong, V. W., Schober, R., Ng, D. W., & Wang, L. (2017). Key Technologies for 5G Wireless Systems. Cambridge University Press.

[10] PODLIPNIG, S., & BOSZ ¨ ORMENY, L. (2003). A Survey of Web Cache Replacement Strategies. Retrieved from http://citeseer.ist.psu.edu/showciting?cid=261528

[11] Arlitt, M., Cherkasova, L., Dilley, J., Friedrich, R., & Jin, T. (2000). Evaluating content management techniques for Web proxy caches. Retrieved from https://dl.acm.org/citation.cfm?doid=346000.346003

[12] A. Kumar, M. Manoj and A.K. Sarje, "A Predicted Region based Cache Replacement Policy for Location Dependent Data in Mobile Environment," I. J. Communications, Network and System Sciences, 1: 1-103, 2008.

[13] J. Xu, Q. Hu, D. Lee, and W. Lee, "SAIU: An Efficient Cache Replacement Policy for Wireless On-demand Broadcasts," CIKM 2000 Proceedings of the ninth international conference on Information and knowledge management, pp 46-53,McLean, VA,USA, Nov 2000.

[14] Xu, J., Lee, Q. H., & Lee, D. L. (2004, January 01). Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination. Retrieved from http://doi.ieeecomputersociety.org/10.1109/TKDE.2004.1264827

[15] Xing, Q., Wang, J., Li, Y., & Han, Y. (2015, August 26-28). A User-Relationship-Based Cache Replacement Strategy for Mobile Social Network. Retrieved from https://ieeexplore.ieee.org/document/7314709/

[16] Khreishah, Abdallah, Jacob Chakareski. "Collaborative caching for multicell-coordinated systems". *IEEE Conference on Computer Communications Workshops*, 2015.