Sudan University of Science and Technology

College of Graduate Studies

A Comparative study of A Cache Replacement Policies for Information-Centric Network

دراسة مقارنة لسياسات استبدال ذاكرة التخزين المؤقت لشبكة مركزية المعلومات

Prepared By:

Tmaheen Albdry Mohammed Albdry

Supervisor:

Dr. Mohamed Elghazali Hamza Khalil

A thesis Submitted in Partial Fulfillment of the Requirement for the Degree of M.Sc. in Information Technology (Networks)

December 2021

الإستهلال

بسم الله الرحمن الرحيم

(ومآ أوتيتم من العلم إلا قليلا)

سورة الاسراء (٨٥)

Acknowledgments

First of all, I want to express my sincere thanks and gratitude to my supervisor, Dr. Mohamed Elghazali Hamza. He provides excellent research environment. He gave me great comfort and encouragement. He is an excellent support both in academia and personality. I want to express my great gratitude to Dr. Mohammed Elghazali. Secondly, I would like to thank all my friends, who help me writing and organizing this research. Last but not least, I want to thank my mother. Without her full support, I could not finish my master study.

Abstract

Today's Internet user’s concentration on contents. Search engines, social media, play an important role (users pay an attention for contents which have a large number of like & share), ICN comes to meet the current internet user's needs. In ICN, every content is named, requesting data by name is very beneficial, especially for IOT. Now many researchers are studying ICN-based IoT systems. In this study, both router and user’s devices are able to store content. One of the key features of ICN is in-network content caching. ICN behavior is determined by a3-tuple, which are routing, content insertion and content replacement. Cache Placement strategies (e.g Caching everything-everywhere (CE2) stores all the content to be delivered on every router along the path, this approaches consumes greater amount of buffer space. As a result, a caching decision is made without regard to the content's popularity. for efficiency of caching, cache replacement policies play an important role, because the ICN router cache is limited, cache replacement is required. There are different cache replacement policies. This research compare between cache replacement algorithms (LUR, LFU, TLUR) in term of hit ratio to select the much more efficient one. Our goal is to find the optimal content replacement strategy, to face the big amount of caching (in-network caching) caused by placement algorithms, attempting to improve the network performance in terms of hit ratio and network delay, build model to compare algorithms. To calculate hit rate for every algorithm, and find the best algorithm for ICN. The application was built using web techniques (PHP, Javascript). The result of comparison explains that, the three algorithm is same when cache size is small, but the difference appears when the cache size become larger, whereas TLRU is the best one. LRU, LFU, TLRU achieves 12.5% with 4bytes cache size. While LRU, LFU achieves 25% hit, and TLRU achieves 29%hit with 8bytes cache size, using the same number of requests.

المستخلص

يركز مستخدمو الإنترنت اليوم على المحتويات ، وتلعب محركات البحث ووسائل التواصل الاجتماعي دورًا مهمًا (يهتم المستخدمون بالمحتويات التي تحتوي على عدد كبير من الإعجاب والمشاركة) ، ويأتي آي سي ان لتلبية احتياجات مستخدمي الإنترنت الحاليين. في آي سي ان ، يتم تسمية كل محتوى ، ويعد طلب البيانات بالاسم مفيدًا جدًا ، وخاصة بالنسبة لـ آي او تي. الآن العديد من الأبحاث تعمل على دراسة أنظمة إنترنت الأشياء القائمة على آي سي ان. في هذه الدراسة ، يمكن لكل من جهاز التوجيه وأجهزة المستخدمين تخزين المحتوى. إحدى الميزات الرئيسية لـ آي سي ان هي التخزين المؤقت للمحتوى داخل الشبكة. يتم تحديد سلوك ***آي سي ان*** من خلال -3جداول ، والتي تتمثل في التوجيه وإدراج المحتوى واستبدال المحتوى. تخزن استراتيجيات وضع ذاكرة التخزين المؤقت (مثل التخزين المؤقت لكل شيء في كل مكان (سي إي تو) كل المحتوى الذي سيتم تسليمه على كل جهاز توجيه على طول المسار ، وهذا النهج يستهلك قدرًا أكبر مقدار مساحة المخزن المؤقت ، ونتيجة لذلك ، يتم اتخاذ قرار التخزين المؤقت بغض النظر عن شعبية المحتوى. من أجل كفاءة التخزين المؤقت . تلعب سياسات استبدال ذاكرة التخزين المؤقت دورًا مهمًا ، نظرًا لأن ذاكرة التخزين المؤقت لجهاز التوجيه ***آي سي ان*** محدودة ، يلزم استبدال ذاكرة التخزين المؤقت. توجد سياسات مختلفة لاستبدال ذاكرة التخزين المؤقت ، يقارن هذا البحث بين خوارزميات استبدال ذاكرة التخزين المؤقت (ال آر يو، ال اف يو ،تي ال آر يو) من حيث نسبة النتائج لتحديد الخوارزميات الأكثر فاعلية. الهدف من هذه الدراسة هو العثور على استراتيجية استبدال المحتوى المثلى ، لمواجهة المقدار الكبير من التخزين المؤقت (التخزين المؤقت داخل الشبكة) الناجم عن وضع الخوارزميات ، في محاولة لتحسين أداء الشبكة من حيث نسبة الوصول وتأخير الشبكة ، وبناء نموذج لمقارنة الخوارزميات ، احسب معدل الدخول لكل خوارزمية ، واعثر على أفضل خوارزمية لـ آي سي ان. تم إنشاء التطبيق باستخدام تقنيات الويب (بي اتش بي،جاقاسكريبت) ، وتوضح نتيجة المقارنة أن الخوارزمية الثلاثة هي نفسها عندما يكون حجم ذاكرة التخزين المؤقت صغيرًا ، لكن الاختلاف يظهر عندما يصبح حجم ذاكرة التخزين المؤقت أكبر ، بينما تي ال آر يو هو الأفضل. تحقق ال آر يو و ال اف يو و تي ال آر يو *12.5٪* بحجم ذاكرة تخزين مؤقت يبلغ 4 بايت. بينما تحقق ال آر يو و ا لاف يو نسبة 25٪ من النتائج ، بينما تحقق تي ال آر يو نسبة 29٪ من النتائج بحجم 8 بايت لذاكرة التخزين المؤقت ، باستخدام نفس عدد الطلبات.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CCN	Content Centric Networking
CDN	Content Delivery Networks
CS	Content Store
DDoS	Distributed Denial-of-Service
DONA	data-oriented network architecture
DoS	Denial-of-Service
FIB	Forwarding Information Base
ICN	Information-Centric Network
ICN	Information-Centric Network
IFA	Interest Flooding Attacks
ISP	Internet Service Provider
LCD	Leave Copy Down
LCE	Leave Copy Everywhere
LFRU	Least frequent recently used
LFU	Least frequently used
LRU	Least recently used
MCD	Move Copy Down
MFU	Most frequently used
MRU	Most recently used
NDN	Named-Data Networks
NetInf	Network of Information NetInf
PIT	Pending Interest Table
PoA	point of attachement
PSIRP	Publish Subscribe Internet Routing Paradigm
SAIL	Scalable and Adaptive Internet Solutions
TLRU	Time aware least recently used
TTU	Time to Use

Chapter one
Introduction

1.1 Introduction

This chapter introduces the background of ICN. The motivation of our research in ICN Caching is given as well. Then, we analyze the existing problems in ICN placement strategies. Besides, the contributions of our research work are concluded in this chapter. At the end of this chapter, the outline of this thesis is listed.

With the ubiquity and proliferation of devices that connect us to the Inter-net and the rapid advancement in wireless technologies, the global IP traffic has exploded. Consequently, the usage of computer networking shifted from sharing hardware and processing resources, as its purpose was in the early days of its creation, to accessing and sharing content instead. However, the design of the current Internet architecture was driven by the needs at the time of its creation where the ultimate goal was end-to-end communication between a few machines. Therefore, the existing architecture is facing several challenges in adapting to a phenomenal increase in content.

Having acknowledged the growth of content and the necessity of its efficient distribution, efforts from both academia and industry have been combined in an attempt to adapt the Internet architecture to the explosive content growth experienced in the last decade. This resulted in several proposals that replace the end-to-end model of TCP/IP with a more data-centric architecture under the name of ICN.

1.2 Motivation

In Information Centric Network(ICN), content is uniquely identified and so

endpoints send packets requesting names of specific content rather than an IP address of a specific destination hosting the content. In other words, the requesting client need not know where the data resides.

With naming content uniquely and disposing of the end-to-end principle that keeps end to-end transactions oblivious to resources and content available along the path, ICN leverages in-network caching where routers in the net-work cache content items. There have been several different proposals of ICN architectures they all share the common goal of efficient content distribution using two key features, which are content-based communication and universal in-network caching. in-network caching allows for data being retrieved from intermediate nodes rather than from the server itself, thus rendering content distribution more efficient by reducing network traffic ,download time and server load.

With cache placement algorithms (e.g LCE), all such routers on the for-warding path will cache the content object. As a result, a caching decision is made without regard to the content's popularity or to the space resources available.

1.3 Problem Statement

Cache Placement strategies (like Caching everything-everywhere) stores all the content to be delivered on every router along the path, this approaches consumes greater amount of buffer space, degrade cache performance and network performance. After a cretin amount of time, the buffer is getting full and there is not enough space to store new incoming data in the cache, this result in delay and less network performance. The needs to accommodate the incoming contents in the buffer "cache" is main feature of ICN, there is no way to stop, furthermore caching all contents is impossible, the problem of what content must stay or what content should gone away, is floating on surface. random replacement result in less cache

performance, so effective cache content replacement decision and strategy should take place .The strategy takes in account the performance and full utilization of time, buffer, network, by right selection of the contents must be replace ,and the procedures to be done. the research goal is to find the optimal content replacement strategy, to face the big amount of caching (in-network caching) caused by placement. algorithms, attempting to improve the network performance in terms of hit ratio and net-work delay.

1.4 Objectives

research objectives are :

1. To build model to compare algorithms in term of hit rate. high hit rate means high network performance and less traffic and delay.

   Calculate hit rate for every algorithm.for fixed parameters,(cache size, time, number of requests).
2. To find the best algorithm for ICN networks,the algorithm with high hit rate.
3. To increase the caching replacement process performance,by taking the content popularity into account. this lead to minimize response time, network traffic .and optimal utilization for cache memory and network resources.

1.5 Research Scope

Compare Cache replacement strategies(LRU-LFU-TLRU) for achieving re-search objectives, furthermore the scope of research is to compare the algorithms mention above in term of hit ratio for fixed cache size and fixed number of requests. For efficiency of caching, cache replacement policies play an important role, because the ICN router cache is limited and cannot hold all the content in-side the cache. To devise some space for new content, cache replacement is required. There are different cache replacement policies. One of the most used and popular policy is Least Recently Used (LRU). Least Frequently Used (LFU). First-In First-Out (FIFO).

1.6 Research Outline

The remaining of this thesis is structured as follows: Chapter two: presents

the background of our research. An introduction of ICN is given. ICN architectures ,exactly NDN ,in-network caching ,cache placement and replacement strategies are discussed as well. Chapter three: proposes our model for compare the algorithms ,the simulation ,and parameters. Chapter four: implement the proposed model, the simulation of the mentioned algorithms, compare the results with each other. Based on the results , then choose the much more efficient one. Chapter five: gives a conclusion about this thesis and our future work. also some future possibilities where listed in this area.

Chapter Two
Literature Review

2.1 introduction

This chapter presents the background of our research. It gives an introduction of ICN in Section 2.1, in which it describes the architecture of ICN. Then presents the NDN architecture, in Section 2.2. After that, presents Communication model is in 2.3 and IN-Network caching and its advantages in ICN in 2.4,2.5. It classifies Cache strategies into two major categories: Placement Strategy and Replacement Strategy. In each category, a detailed description is discussed. At last, we present the related studies, another important aspect of ICN, security, naming, routing system. at last we present queuing delay.

2.2 Information-Centric Network (ICN)

The core idea behind information-centric networking (ICN) architectures is that who is communicating is less significant than what data are required. This paradigm shift has occurred due to end-users' use of today's Internet, which is more content-centric than location-centric.

Internet usage has shifted from host-centric end-to-end communication to a content-centric approach mainly used for content delivery. Although content delivery represents such a large percentage of Internet traffic, the paradigm of the current Internet has not been built for content delivery.

Unlike traditional broadcast which sends one title to millions of people across the network at one time, the Internet transmits same videos many times over. the congestion in the Internet will get out of control and new solutions will be required to maintain an acceptable quality of service.

To address the problem, Information Centric Networks (ICN) were pro-

posed.ICN is a novel Internet architecture designed for content delivery. Instead of leading the Internet protocol with an end-to-end communication protocol, ICN switches to a content-centric approach where every content is named. Researchers have proposed multiple architectures [1]. In 2006, the data-oriented network architecture (DONA) project at UC Berkeley proposed an ICN architecture, which improved the security and architecture of TRIAD. The Publish Subscribe Internet Technology (PURSUIT) project, a continuation of the Publish Subscribe Internet Routing Paradigm (PSIRP)project, both funded by the EU Framework 7 Program (FP7), have proposed a pub-lish/subscribe protocol stack that replaces the IP protocol stack. In another approach, the Network of Information (NetInf) project was initially proposed by the European FP7 4WARD project, and further development has been made by the Scalable and Adaptive Internet Solutions (SAIL) project. Simi-larly, Van Jacobson, a Research Fellow at PARC, proposed the Content Cen-tric Networking (CCN) project in 2007.

Currently work is being performed to enhance the CCN architecture called "named-data networks" (NDN). All of these approaches differ in terms of implementation, but they have the same goal to improve the performance and end-user experience of the Internet by providing access to content and ser-vices by name rather than by original location. There are researches talked about ICN and its caching strategies and use in [2] [3] [4]. This is achieved by changing the concept of link protection to content protection and by exploiting in-network storage of content.

Among all these new architectures, CCN has attracted most of the attraction of the community due to three reasons:

i. In-network caching features at every node
ii. coupled name resolution
iii. data forwarding and a unified naming scheme.

From these features, in-network caching impacts directly on the content delivery efficiency. Despite the large caching literature already existing,the premises of a CCN architecture makes its study challenging. The in-network

caching features at every node becomes CCN into a network of caches. Internet has never handled caches at such a large scale, caches were located at fixed locations and now caches are placed everywhere.CCN stores content at chunks of content at a fine-granularity, in contrast with traditional architecture were complete objects were stored. CCN routers must deal with large cache sizes and a catalog ranging for all the content from the Internet. The CCN efficiency depends drastically on the performance of its caching features [5].

2.3 Content Centric Networking CCN/ NDN

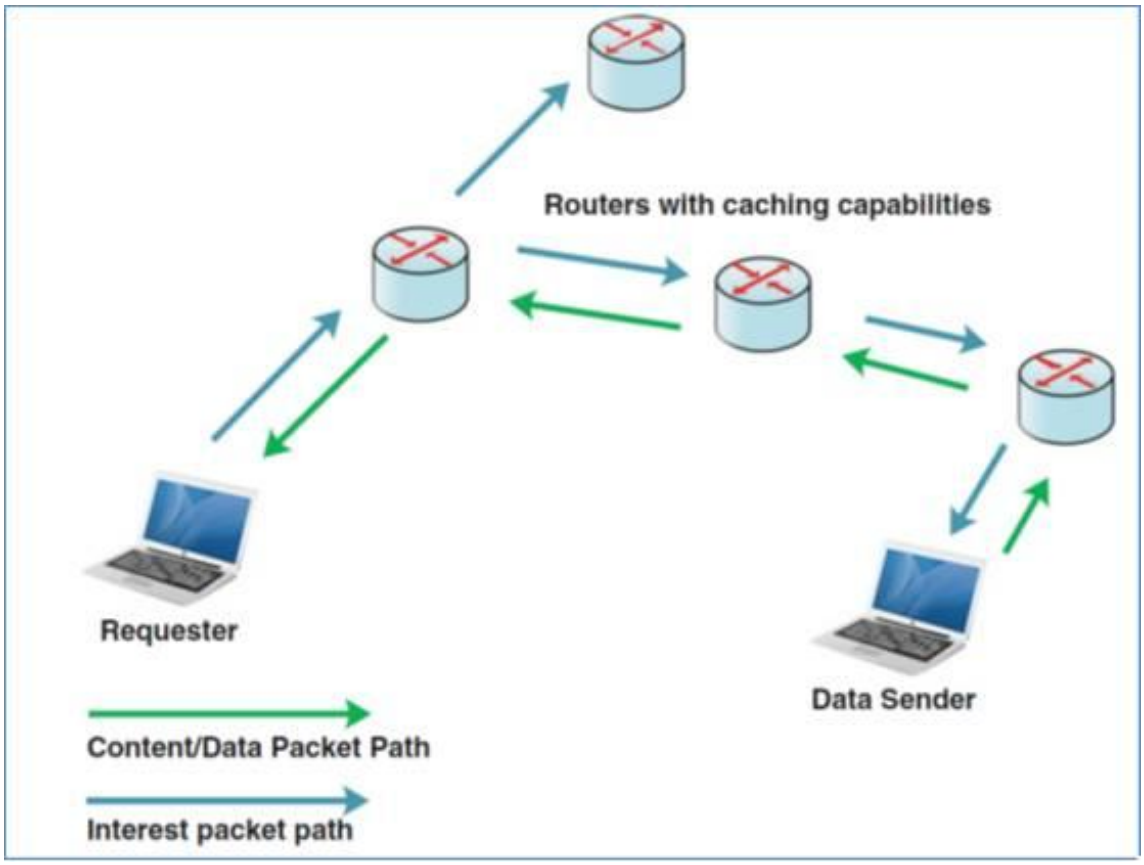

Figure 2.1: CCN/NDN overview [6]

Content Centric Networking (CCN) in one project that follows the ICN paradigm. It was originally started at the Palo Alto Research Center (PARC), a research and development company. Currently the CCN approach is being continued by for example the Named Data Networking (NDN) project and

the Community Information Centric Networking (CICN) project at Cisco.
In CCN communication is driven by the consumers of data, with publishers making that data available for access in the form of content. In CCN there are two primary packet types: Interest and Data. Consumers first use an Interest packet in order to request some content and the publisher then delivers that content, in the form of a Content Object, in a Data packet. Content Object is the CCN specific term for the generic ICN NDO. Routing is name-based and Interests are routed hop-by-hop toward publishers using longest prefix matching. Longest prefix matching is originally a forwarding algorithm used by TCP/IP routers. some papers talk about routing as in [7] [8]. When applied to CCN it means that a message will be forwarded according to the entry in the forwarding table with the name that has the longest prefix in common with the name of the message.

The namespace of CCN is hierarchical, unlike several other ICN projects which use flat namespaces. The structure is similar to the current URLs, where the hierarchy is rooted in a publisher unique prefix under which content is published. This means names are aggregately when routing in a manner reminiscent of TCP/IP route aggregation, which improves routing scalability. A CCN router has three primary data structures.

2.4 Forwarding Information Base (FIB)

It is the forwarding table. In CCN the FIB operates similarly to the FIB of a TCP/IP router, hence the identical name. It maps Content Objects, represented by their names, to network interfaces. A selected network interface leads to a next hop toward one publishing location for the content matching that particular name. In CCN terminology interfaces are simply referred to as faces.

The primary difference when compared to the FIB of TCP/IP is the fact that CCN supports multi-sourcing. In CCN each FIB entry can map a single Content Object to multiple faces, as the same content can be published at multiple network locations. How to populate the FIB is an important

problem and a common suggestion is to use a routing protocol, much like how it is done in TCP/IP. When there are multiple alternative faces to choose from for a Content Object a forwarding strategy determines to which face, or faces, the Content Object should be forwarded.

2.5 Pending Interest Table (PIT)

The PIT stores state about forwarded Interests in the form of a map, which maps Content Objects to faces from which Interests for that Content Object has been received. Similarly, to the FIB, the Content Objects are once again represented by their names.

Content Objects are not routed from the publisher to the consumer, they instead travel the same path as the initial Interest, but in the reverse direction, by consuming the state left behind by the initial Interest in the PIT at each passed hop. This is called the reverse request path. The state stored in the PIT thus serves as a breadcrumb for the Content Object to follow as it travels toward the consumer.

2.6 Content Store (CS)

The CS is the cache where each network node can store content, enabling on path caching. For example, there are papers in a caching mechanisms as in [9] caching strategies as in [10] [11], On path caching is the possibility that, as an Interest is routed toward a publisher, a cache hit occurs in the CS of one of the intermediate nodes. This reduces content download time, network traffic and the server workload. The CS operates according to some cache strategy, for instance Least Recently Used (LRU) or Least Frequently Used (LFU). There is no requirement for every node to share a single cache strategy, meaning the cache strategy can be decided on an individual node basis.

2.7 Communication model in ICN

In ICN, packets used for communication are of two types

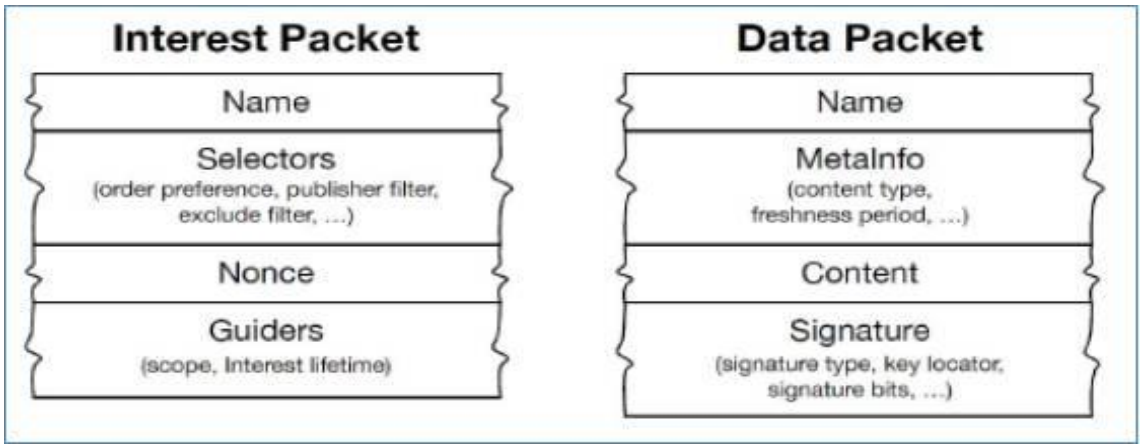

Figure 2.2: ICN packet types [12]

Interest and Data [figure 2.2]. User requests a particular data using an interest packet which includes the name prefix of the content to be fetched. Name prefix from the interest packet is used to route it forwards the requested content. The packet whose name matches with the name prefix of the interest packet or has matching data cached locally is sent to the user through the same path in reverse direction leaving cached copies at each intermediate node from source to destination. Every interest packet additionally contains a NONCE field, a random number assigned by pending interest table (PIT) to avoid forwarding loops. Data packet carries the requested data plus the signature of producer to review for inconsistencies. [12].

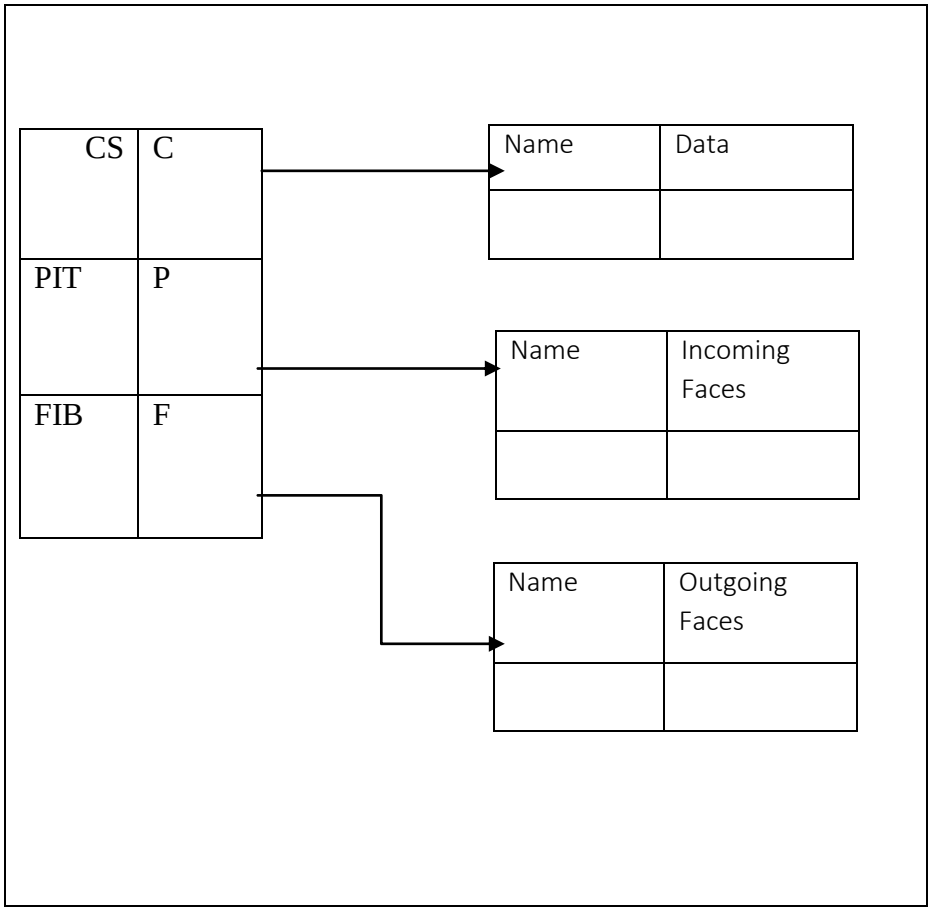

Figure 2.3: Components in ICN router

ICN routers contain three tables as shown in figure 3.Forwarding Information Base (FIB), Pending Interest Table (PIT) and a Content Store (CS). FIB acts as a routing table in an IP router. Instead of IP prefixes, ICN FIB is indexed by name prefixes and every FIB entry may have several next hops in place of one best next-hop for every name prefix. PIT keeps track of received interests. It records which interface(s) the interest is received from and has been forwarded to. CS acts as a temporary cache of data packets. If the re-quested data is in the CS, the node can immediately send the data without generating further requests to the content provider.

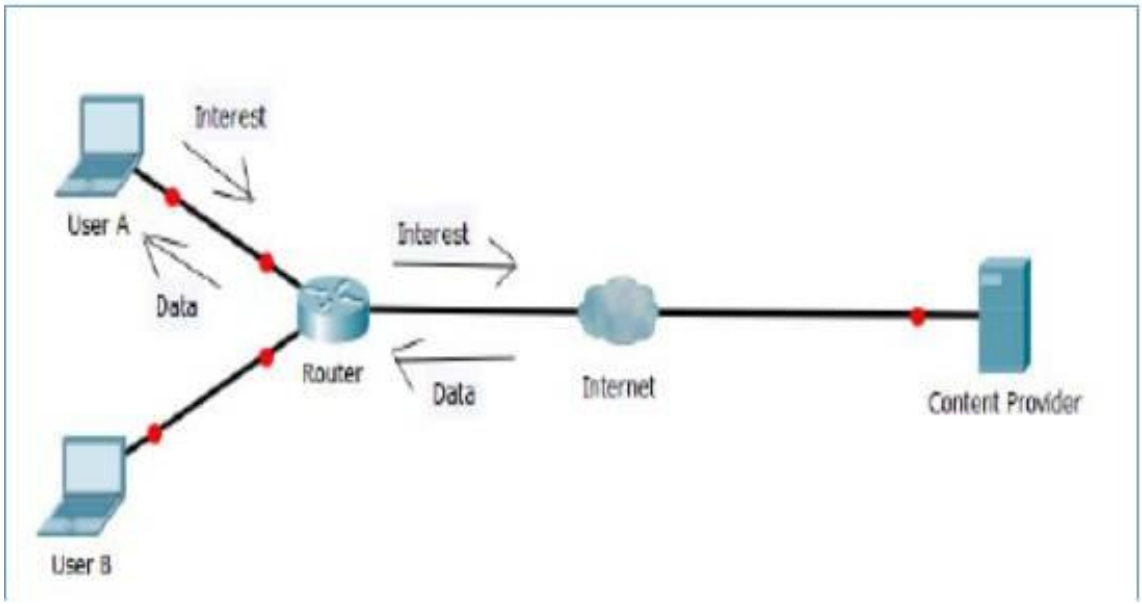

Figure 2.4: Basic operation of ICN[12]

In this example [figure2.4]: User A forwards the interest packet first. When this interest packet sent from user A reaches the router, router queries it's CS for the requested content. If there is any matching data stored in cache, data is immediately sent to the requester. In this example, as no such content is available in CS, router looks up the PIT to see if there is any request for this content. A record of the incoming interface is made in PIT if an identical entry is found. If the entry is not found in PIT, both incoming interface and outgoing interface are recorded in addition to the name of the requested content. Router will then forward interest packet depending on the information in FIB. There is no cached content for this request in the network, so the interest packet eventually hits the content provider and the data packet is sent to the requester. When this data packet arrives at router, it first looks for any pending entry in PIT for the same. If so, data packet gets forwarded towards the downstream interfaces and its corresponding entry in PIT is removed while caching the data packet in its

CS to satisfy future requests. Now, when user B sends the interest packet for the data similar to the one requested by user A, this data packet gets delivered from the nearest cache (router in this example) to user B directly [figure 2.4].

2.8 Related Work

Researchers work on several caching strategies , their algorithm , caching types , their advantages and issues . Based on the survey work, various cache methods are compared depend on different criteria. Different challenges in their works are analyzed, and a proposal based on these works is given at the end [2].

To improve the cache hit ratio, most of the existing schemes store the con-tent at maximum number of routers along the downloading path of content from source. While this helps in increased cache hits and reduction in delay and server load, the unnecessary caching significantly increases the network cost, bandwidth utilization, and storage consumption. To address the limi-tations in existing schemes, researchers in [3] propose an optimization based in-network caching policy, named as opt-Cache, which makes more efficient use of available cache resources, in order to reduce overall network utilization with reduced latency [3].

Researchers in [1] propose a caching and replacement strategies for content in Content-Centric Network (CCN). The caching strategy will choose the node that will be cached on based on the network topology. The proposed replacement strategy will take in its consideration the number of resources that the content has been consumed and if the content has been requested recently or not. To evaluate their proposed work, Researchers use a ccnSim simulator, and the simulation results show that their proposed caching strategy provides more significant result than the Leave Copy Everywhere (LCE) strategy and the replacement strategy provide more significant result than the Least Recently Used (LRU) replacement strategy.

ICN behavior is determined by a 3-tuple, which are routing, content

insertion and content replacement. Besides, Routing algorithms influence content insertion performance and, which in turn, influences in replacement policies performance. Furthermore, it is proven that content insertion policies in-fluence routing performance and there is no work regarded to analyze the impact of replacement algorithms in content insertion. Therefore researchers in [13] proposed a new caching metric called Replacement Ratio and a dynamic content insertion strategy named RatioCache to prove that content replacement, which is strongly bounded to caching system, also influence the caching process. In [4] The problem of the study is that the NDN architecture is processing several forms of online video requests simultaneously. However, limited cache and multiple buffering of requested videos result in loss of data packet as a consequence of the congestion in the cache storage network. Ad-dressing this problem is essential as congestion cause network instability. This work emphasizes on the review of cache replacement strategies to deal with the congestion issue in Named Data Networks (NDN) during the VoD delivery in order to determine the performance (strengths and weaknesses) of the cache replacement strategies. Finally, the study proposes the replacement strategies must be enhanced with a new strategy that depends on popularity and priority regarding the congestion.

Researcher in [14] propose a content replacement scheme for ICN, called Randomized LFU that is implemented with respect to content popularity tak-ing the time complexity into account. They use Abilene and Tree network topologies in their simulation models. The proposed replacement achieves encouraging results in terms of the cache hit ratio, inner hit, and hit distance and it outperforms FIFO, LRU, and Random replacement strategies.

The researcher in [15] proposed New caching policies and described: XCaching Type A, XCaching Type B, XCaching TypeC. The results of a comparative analysis of the developed caching approaches in model of information-content network are presented. Investigated the probability of

hitting the cache and the uniqueness of caching systems was made. The results of caching policy evaluations are obtained on the basis of constructed imitation model of information-content network.

Paper [16] discusses the potential of leveraging Information-Centric Net-working (ICN) principles in the 3GPP architecture for V2X communications. researcher consider Named Data Networking (NDN) as reference ICN ar-chitecture and elaborate on the specific design aspects,required changes and enhancements in the 3GPP V2X architecture to enable NDN-based data ex-change as an alternative/complementary solution to traditional IP networking, which barely matches the dynamics of vehicular environments. Results are provided to showcase the performance improvements of the NDN-based proposal in disseminating content requests over the cellular network against a traditional networking solution.

To enable a complete ICN caching solution for communication networks, Quang Ngoc Nguyen and other proposed an autonomous replacement policy to optimize the cache utilization by maximizing the utility of each CN from caching content items. By simulation, they show that PPCS, utilizing edge-computing for the joint optimization of caching decision and replacement policies, considerably outperforms relevant existing ICN caching strategies in terms of latency (number of hops), cache redundancy, and content availability (hit rate), especially when the CN’s cache size is small [7].

2.9 ICN Content Caching

Approaches to caching can be categorized into off-path caching and on-path caching based on the location of caches in relation to the forwarding path from a source to a consumer. Off-path caching, also referred as content replication or content storing, aims to replicate content within a network in order to increase availability, regardless of the relationship of

the location to the for-warding path. The actual number of replicas and the specific nodes in which replicas may be stored is a decision made by the Internet Service Provider (ISP) that supports the specific network. In on-path caching approaches, content is replicated at nodes along the forwarding paths from sources to consumers. The decision to cache a content resource at a specific node is strictly related to the content that is being requested [17].

2.10 Advantages of in-network caching in ICN

I) Reduction of content delivery delay and round- time (RTT trip): Because of in-network caching capability, contents are stored at the intermediate nodes closer to requesters and can be quickly retrieved from the server.

II) Higher content availability: In-network caching ensures higher availability of content as they are cached on all nodes back from source to requester thereby mitigating Denial of Service (DOS) to a significant level [18].

III) Network caching shows better resiliency towards packet loss by quickly retransmitting them from the nearest node which has uncorrupted copy of the content.

IV) In-network caching results in significant reduction of total traffic since data packets traverse fewer links in case of a cache hit.

V) Cache hit leads to serving one request less, thus reduced server load. [12]

2.11 Cache placement and cache replacement

Cache memory is used to store frequently referred pages to increase the throughput of the system and with minimum delay .In ICN, cache placement and cache replacement are different terms. Cache placement basically references ‘in what place the content should be placed?’ but Cache replacement defines how the event cache content eviction should take place to achieve a high hit ratio and minimum latency.

2.12 Caching Placement strategies

2.12.1 On-path Caching

On-path caching decisions applies only to the requested content(s); other content is not taken into account, while content may be cached only at the nodes lying on the delivery path.On-path caching is strictly related to the requested content and popularity rates of each item. [17].

2.12.2 Probabilistic Caching

It is a general approach, according to which each node on the delivery path decides to cache the content based on a probability p. The probability p may be a pre-determined value [19].

2.12.3 Random Caching

This model is fairly simple and results in no additional load on the network. However, it is not able to exploit the advantage of having knowledge of the optimal positions for caching each content [17] [19].

2.12.4 Unique Caching

In this approach, content is cached only in one node along the delivery path which is chosen randomly. Since only node is chosen, the probability of caching at each node equals to, one to the number of intermediary nodes.

2.12.5 Caching Everything-Everywhere

The CE2approach simply caches every content in every intermediate node in-volved in the delivery path. The CE2 approach has been criticized in a number of works for resulting into unnecessary content redundancy and resource con-sumption. As an additional drawback, CE2 does not take into account the content's popularity, providing the same probability,for both popular and un-popular content, to be cached. In contrast to its disadvantages, CE2 holds the advantage of providing fast content distribution [17].

2.12.6 ProbCache

According to this policy, each node stores a copy of the content with probability p. If the probability is 1, the LCE policy is implemented. Each node contains cache sizes and data regular changes over time, so it has to be an effective content replacement policy. If the node does not have enough space to cache a copy of the content, it selects suffer for replacement based on access time,the number of visits or access order [11].

2.12.7 Leave Copy Everywhere (LCE)

LCE leaves a copy of the content in each node along the path from producer to end user.LCE can be considered as a probability strategy with caching probability equal to one in each node. LCE designed to reduce user access time to a content and minimize the frequent download from content producer.The main disadvantage of this strategy is the redundancy of caching.To reduce cache redundancy in ICN the LCD is designed. LCD caches the content only at the direct downstream node of the node that cache hit occurs on it. [20] [21].

2.12.8 Leave Copy Down (LCD) and Move Copy Down(MCD)

(MCD) are other cache placement policies. When a user sends an Interest packet,and cache hit occurs, the content will be cached only in the neighbor downstream node. LCD pushes a copy of the content one hop closer to the client after each cache hit. Also in MCD once a cache hit occurs the content is cached only at the neighbor downstream router. MCD deletes the cached content after the hit while LCD does not [11] [21].

2.13 Replacement Strategy

Cache is filled after a certain amount of running time.Since that, a replacement strategy is needed to cache a new upcoming content. Replacement strategies can be categorized based on several characteristics

has been proposed in :

2.13.1 Random replacement strategy

When a content data are requested by requester node (client) and caching router find that content in its content store (CS) then that event is called hit event and the content data is immediately sent to the client by the caching router. When the content is not found in the content store (CS) of the caching router then the respective data request is sent to the server and in returning of that data from server, caching router selects one of the content in its CS randomly and replace that content with requests incoming content from the server .

The selection criteria of content which has to be replace is done random. [22] [19] [23].

2.13.2 Least recently used strategy

Least recently used (LRU) is one of famous and mostly used cache replacement strategy in ICN.In Least recently used strategy when a content data are requested by requester node and caching router finds that content,caching router selects one of the content in its CS on the behalf of recency of usage.

Most popular content items will be demanded more in the network so its usage will be more and recency is directly proportional to the usage. So the item which having less recency will be selected for replacement by a caching router in its CS.LRU replacement strategy gives a high hit ratio because the most popular content is accessed many times in the modern world scenario [22] [23].

2.13.3 First in first out strategy

This strategy is very simple to understand and implements .In this replacement ,when a content data is requested by requester node ,caching router selects one of the content in its CS on the behalf of oldness of usage. In this

scenario, oldness is directly proportional to the time at which the content data was stored in the cache storage. The more old data have high probability to be replaced with new arrived cached content [19] [23].

2.13.4 Least frequently used strategy

Least frequently used Strategy works with the maintenance of a counter for each content data. This counter for each content item tracks how many number of times that particular content item is requested or referred. In Least frequently used (LFU) strategy, when a content data is requested by requester node ,caching router selects one of the content in its CS on the behalf of less value of counter. This less or low value of counter indicates less number of times that particular content item is referred. The caching router will select a content item in its cache who has a low value of the counter and replaces it with newly arrived content data [22] [23].

2.13.5 Most recently used strategy

This strategy is opposite to least recently used replacement strategy, when a content data are requested by requester node (client) and caching router finds that content in its content store (CS) ,the content data is immediately sent to the client by the caching router. caching router selects one of the content in its CS on the behalf of high recency of usage. Researches show that the MRU replacement holds good results for scenarios which having accessed old content data in spite of new one [22] [23].

2.13.6 Time aware least recently used (TLRU)

The Time aware Least Recently Used (TLRU) is a variant of LRU designed for the situation where the stored contents in cache have a valid life time. The algorithm is suitable in network cache applications, such as Information-centric networking (ICN), Content Delivery Networks (CDNs) and distributed networks in general. TLRU introduces a new term: TTU (Time to Use). TTU is a time stamp of a content/page which stipulates the usability time for the content based on the locality of the content and the content publisher announcement. Owing to this locality based time stamp,

TTU provides more control to the local administrator to regulate in network storage. In the TLRU algorithm , when a piece of content arrives, a cache node calculates the local TTU value based on the TTU value assigned by the content publisher. The local TTU value is calculated by using a locally defined function. Once the local TTU value is calculated the replacement of content is performed on a subset of the total content stored in cache node. The TLRU ensures that less popular and small life content should be replaced with the incoming content [24] [23].

2.13.7 Least frequent recently used (LFRU)

The Least Frequent Recently Used (LFRU) cache replacement scheme com-bines the benefits of LFU and LRU schemes. LFRU is suitable for 'in network' cache applications, such s Information centric networking (ICN), Content De-livery Networks (CDNs) and distributed networks in general. In LFRU, the cache is divided into two partitions called privileged and unprivileged partitions. The privileged partition can be defined as a protected partition. If content is highly popular, it is pushed into the privileged partition. Replace-ment of the privileged partition is done as follows: LFRU evicts content from the unprivileged partition, pushes content from privileged partition to unpriv-ileged partition, and finally inserts new content into the privileged partition. In the above procedure the LRU is used for the privileged partition and an approximated LFU (ALFU) scheme is used for the unprivileged partition, hence the abbreviation LFRU.

The basic idea is to filter out the locally popular contents with ALFU scheme and push the popular contents to one of the privileged partition [25].

2.13.8 Most frequently used strategy

This strategy is opposite to least frequently used replacement strategy. when a content data are requested by requester node, caching router selects one of the content in its CS on the behalf of the high value of counter. This

high value of counter indicates a large number of times that particular content item is referred [22].

2.14: Security

Instead of securing connections, ICN model is based on securing data at network layer. Each data packet is digitally signed by the producer, allowing consumers to verify integrity and data-origin authenticity. A producer is thus required to have and distribute at least one public key. Existing trust models (e.g. a PKI or Web-of-Trust) can be used to validate producer identity and key ownership. Data confidentiality can be guaranteed by encrypting data payload and preventing information leakage from the name.

one commonly recognized benefit of ICN data-centric security approach is that it places trust in producers rather than in hosts that store and serve data. This enables in-network efficient data delivery operations, such as filtering, caching and multicasting, without affecting the data security properties en-forced by the data producer [28].

It is also stated that ICN can mitigate traditional Distributed Denial-of-

Service (DDoS) attacks for the certain data providers. in-network caching in ICN can greatly avoid DDoS attacks, and name-based forwarding in ICN can trace the attackers easily.Unfortunately, ICN brings a new varietal DDoS attack called Interest Flood-ing Attacks (IFA), which has become a big threat for information-centric . Typically, attackers issue a large number of fake Interest messages to request nonexisting Data, which can lead to the memory overflow for the ICN-IoT nodes. Recently, many mechanisms have been proposed to mitigate the IFA attacks [29] [30].In paper [31] the researchers showcase the existing literature in security and privacy in ICN and present open questions.

2.15 Name resolution and data routing

The name resolution is a mechanism that enables a consumer or a content

subscriber to find NDO by using a name.This mechanism provides a means of mapping a name and content locator and forward the requested data to the source. After the source of the content according the requested name has been found, the data routing process then constructs a path for transferring the data from the source to the user/client who requested the content.in [32] researchers proposed a hybrid name resolution approach, in which each content has a Home Node located in ICN routers.

2.16 Routing system in ICN

In ICN, data objects must be identified by names regardless their location or container and the names are divided into two types of schemes: hierarchical and flat namespaces. A hierarchical scheme used in CCN and NDN architec-tures has a structure similar to current URIs, where the hierarchy improves scalability of routing system. It is because the hierarchy enables aggregation of the name resulting in reducing the size of RIB or FIB as similar to IP routing system. In a flat scheme, on the other hand, name routing is not easy since names in a flat namespace cannot be aggregated anymore, which would cause more the scalability problem in routing system. In order to address such problem, a flat name can be resolved to some information which is routable through NRS,more details in [8] .

2.17 Queuing delay

Is the time spent by the packet sitting in a queue waiting to be transmitted onto the link. The amount of time it needs to wait depends on the size of the Queue. If the Queue is empty, then it transmitted immediately, but if it’s sitting behind other packets, then it needs to wait for the packets in front to be transmitted first.

researchers in [33] introduced an interest forwarding mechanism to process the requests of consumers at a CCN router, Interest packets are forwarded with respect to the priorities of addressed content while the

priority level settings are done by content publishers during an initialization phase using a collaborative mechanism of exchanging messages to agree to the priority levels of all content according to the content-nature. Interests with higher priority content are recorded in Pending Interest Table (PIT) as well as forwarded to content publishers prior to those with lower priority content.

NDN defines two basic types of packets: Data and Interest. Content items are permanently stored in the repository (provider)and partly cached in the intermediate nodes. A content item is split into a sequence of Data packets uniquely identified by names. Each consumer implements a receiver-driven transport protocol to retrieve content by sending Interest requests. A name-based routing protocol guarantees the Interests are routed toward the data repository. Every intermediate node keeps track of pending Interests, in order to deliver the requested Data packets back to the receiver through the reverse path of Interests. Each router is equipped with a local cache that stores Data packets in order to satisfy future Interests for the same Data. In addition, intermediate nodes perform Interests aggregation to avoid forwarding multiple interests for the same Data while the first one is pending.

Researchers in [34] introduce the Markovian Queuing System theory into the ICN modeling. they adopt the Queuing theory to analyze the queuing delay which is a key part of the content delivery time [20].

2.18 Summary

This chapter, presents an overview of the research background of ICN, NDN, IN-Network caching in ICN witch lead to cache placement and replacement strategies. Include that introduced to suit ICN network.at the end of this chapter we presented the related works and queuing delay.

Chapter Three
Methodology and Research Framework

3.1 Introduction

In this chapter, demonstrate our proposed model, the model involves an ICN with an NDN architecture, the “in-network” capabilities rise the need for efficient cache replacement strategy. The main goals of the proposed model are to compare algorithms and showing, analyzing the results in term of hit ratio. This chapter is organized as follows. Section 3.1 presents the hit ratio and Section 3.2 the cache load. Section 3.3 describe the Framework.

3.2 Cache hit ratio

Cache hit ratio will be evaluated by achieving the number of hits for overall request of accessing content from the requested client. If the request for content item made by the client is found in the content store of a caching rather than this phenomenon is called cache hit. The increasing number of cache hit leads to high performance of the information centric network because of less delay and content will be reached for the client before expected time. Cache hit ratio is directly proportional to the number of times content item found in the content store of caching router.

$$C(\text{hit}) = \frac{\text{number of times content found in CS}}{\text{Total no. of request in that caching router}}$$

3.3 Cache load

The term cache, load is indicated that how much request is processed by a server or a caching router at any particular time. Load is the total number of requests for accessing the content data/ items from client to the server. For better performance of the system, it is necessary that the server or caching router should not be overloaded with incoming requests for content access.

3.4 Techniques used

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.PHP is a recursive acronym for ”PHP: Hypertext Preprocessor”. It is server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

Canvas.JS: Canvas.JS is an easy to use HTML5 and Java script Charting library. This allows to create Rich reports that work across devices without compromising on Maintainability or Functionality.

- Why Canvas.JS
- Very simple and intuitive API .
- Comes with Beautiful and Elegant looking themes.
- High performance and Works on all modern devices.
- Canvas.JS is Standalone – does not depend on any other library.

3.5 Research Framework Description

3.5.1 Network environment

Our system model involves an ICN with an NDN architecture. It includes one or more servers that will have the original copies of the files. Moreover, the network includes routers that connect users to those servers. Due to the content oriented nature of the network, the features of our system are those of NDN which we reiterate as the following:

i. in-network storage capabilities at intermediate routers
ii. packet forwarding using the PIT and FIB data structures
iii. routers can serve requests for content they have cached
iv. interest packets are forwarded towards the data source.

3.5.2 Choose three algorithms to compare LRU, LFU, TLRU

3.5.2.1 Least recently used strategy(LRU)

Least recently used (LRU) is one of famous and mostly used cache replacement strategy in ICN. **here is algorithm:**

1. Start traversing the content.
2. If set holds less content than capacity.
3. Insert content into the set one by one until the size of set reaches capacity or all content requests are processed.
4. Simultaneously maintain the recent occurred index of each content in a map called indexes.
5. Increment content fault
6. Else If current content is present in set, then increment hit.
7. Else Find the content in the set that was least recently used.
   We find it using index array .We basically need to replace the content with minimum index.
8. Replace the found content with current content.
9. Increment content faults.
10. Update index of current content.
11. Return content faults.

3.5.2.2 Least frequently used strategy(LFU)

Least frequently used Strategy works with the maintenance of a counter for each content data. This counter for each content item tracks how many num-ber of times that particular content item is requested or referred.

1. Take inputs
2. Initialize cache and count array to -1
3. If (cache miss)
4. Find the least frequently used content from the contents in the cache
5. Replace content in cache by current content.
6. Create array of counts and store it in ’count’ array
7. End If
8. Increment counter

3.5.2.3 Time aware least recently used (TLRU)

The Time aware Least Recently Used (TLRU) is a variant of LRU designed for the situation where the stored contents in cache have a valid life time. A brief stepwise explanation is given below.

Step 1: Calculate (TTUij) local time stamp by (j node)value for arriving content based upon composite function . This step is optional and we argue that function should be defined by the local network administrator according to local policies and requirements.

Step 2: Proceed to save arriving content in cache if the average request time Ti j is smaller then TTU ij calculated in step-1.The reasons for this step is that if the average request time (in CCN average request time can be calculated by using information stored in PIT-Pending Interest Table) Tij > TTUij then there is a high probability that TTUij will expire before arrival of next request which means storing this content has no use. This step also endorses that relatively more popular contents should be stored.

Step 3: Store the content if there is an empty space in cache otherwise apply LRU on Ev[j] cache state Sk[j] . Subset Ev[j] is a contraction of s[j], calculated based upon the remaining TTU value and average request time. Contraction endorses that relatively less popular contents should be evicted.

3.5.3 build system to comparison

We compare between the caching replacement policies(LRU,LFU,TLRU) us-ing the same content placement policies(e.g LCE) when caching space at nodes is not enough to cache new content. The model compare all three

algorithms mentioned above for the fixed parameter value (number of requests and cache size),then calculate the hit and miss rate for each one. then changes the values and calculate again.

3.5.4 The sequence of system explained as followed: -

i. Select cache size (e.g 4 MB).
ii. Enter requests (the number of request should exceed the maximum cache size)e .g 10 requests .
iii. Execute LRU algorithm.
iv. Execute LFU algorithm.
v. Execute TLRU algorithm.
vi. Read the hit and miss for algorithm and write down to a table ,for chart.
vii. Change the number of requests by increments (e.g 20 requests) to see if changes.
viii. *Repeat steps (from 3 to 6)* .
ix. Repeat all the above steps for new cache size.
x. Use the table data in step 6 to drow a chart for each cache size.

The system flowchart explained on figure 3.1

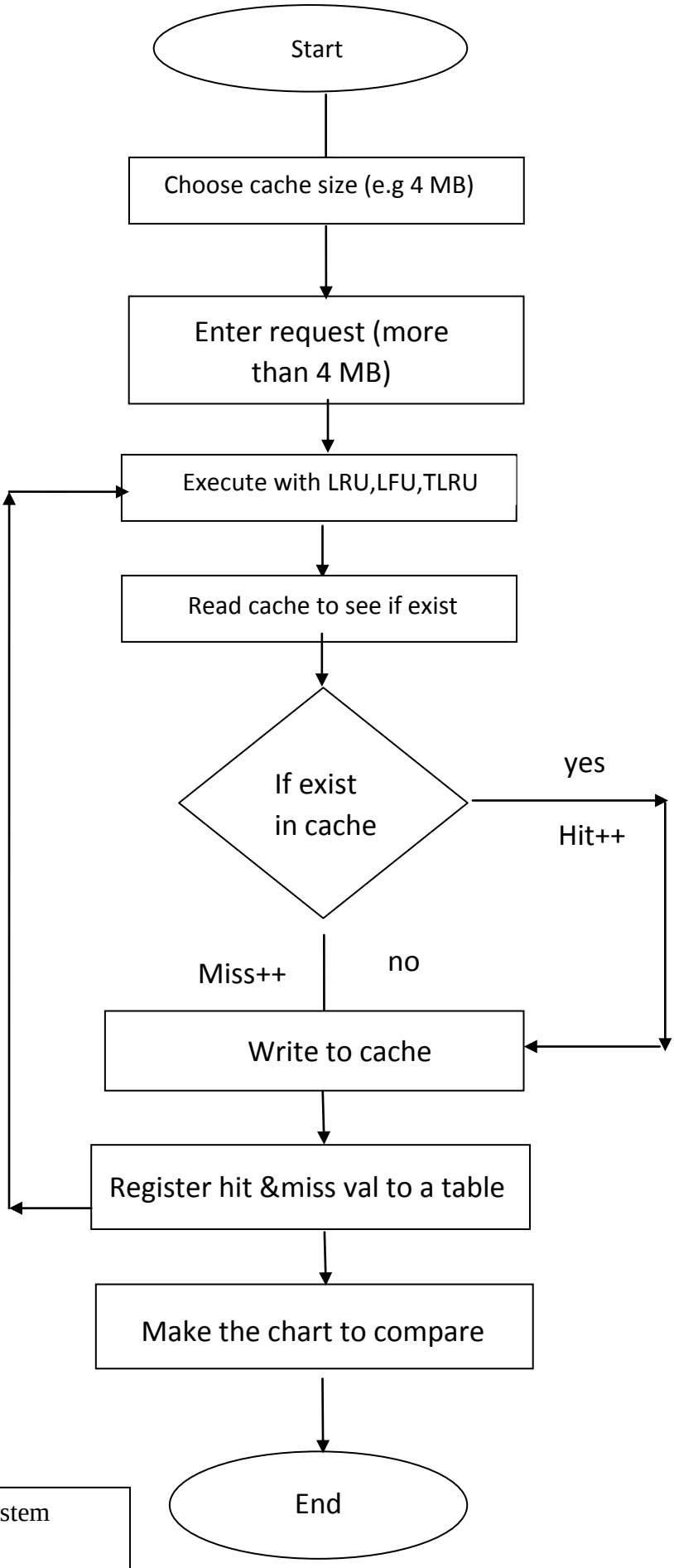

figure 3.1 The propose system flowchart

Table 3.1: Simulation parameters

Num	PARAMETERS
1.	Total number of request
2.	Total number of contents in the network
3.	Cache memory size of the router
4.	Simulation Time in seconds

3.5.5 specify parameters used in system

The proposed model parameters explaned in table 3.1

Total number of request: refer to the incoming requests to search for specific content in cache.

Total number of contents in the network: the contents saved on cache memorys on network.

Cache memory size of the router: the cache size in MB of routers.

Simulation Time in seconds: the time spent in seconds.

3.5.6 decide the best one according to the system

At the end, the system can tell us what is the best, the best will achieves the highest hit rate.

3.6 Summary

This chapter descript the framework used to solve, explains the proposed system by comparing the selected algorithms, determine comparison parameters, the techniques used to develop, and the system flowchart.

The proposed system compare three algorithms for the fixed parameter value (number of requests and cache size), then change the values.

Chapter Four
Implementation and Findings

4.1 Introduction

This chapter, present the results of execution of system which mentioned in chapter three, and then analysis the results found in the experiment. Before mention the results will demonstrate the execution of system in details.

4.2 Framework of Implementation

We design an application using web techniques. The application requires the sequence of query (input) and then show the corresponding result for the selected algorithm by clicking the button carries the algorithm name. The result screen shows the cache (last state) also the hit ratio, number of hits ,miss ratio,t he number of missed. The total number of query, cache size.

4.3 Design and implementation

The description of implementation was mentioned earlier in chapter three, here will implement the proposed solution to gain and analyze results .

Table 4.1 explain the results of hit rate for entering deferent number of requests(10,20,24) for every algorithm,with fixed cache size(4).

Table 4.1: Calculate hit rate for different number of requests with fixed Cache size=4

Parameter	10 Requests	20 Requests	24 Requests
LRU	0	5	12.5
LFU	0	5	12.5
TLRU	0	5	12.5

Table 4.2 explain the results of hit rate for entering deferent number of

requests(10,20,24) for every algorithm ,with fixed cache size(8).

Table 4.2: Calculate hit rate for different number of requests with fixed Cache size=8

Parameter	10 Requests	20 Requests	24 Requests
LRU	10	10	25
LFU	10	10	25
TLRU	10	15	29

4.3.1 Results of hit ratio

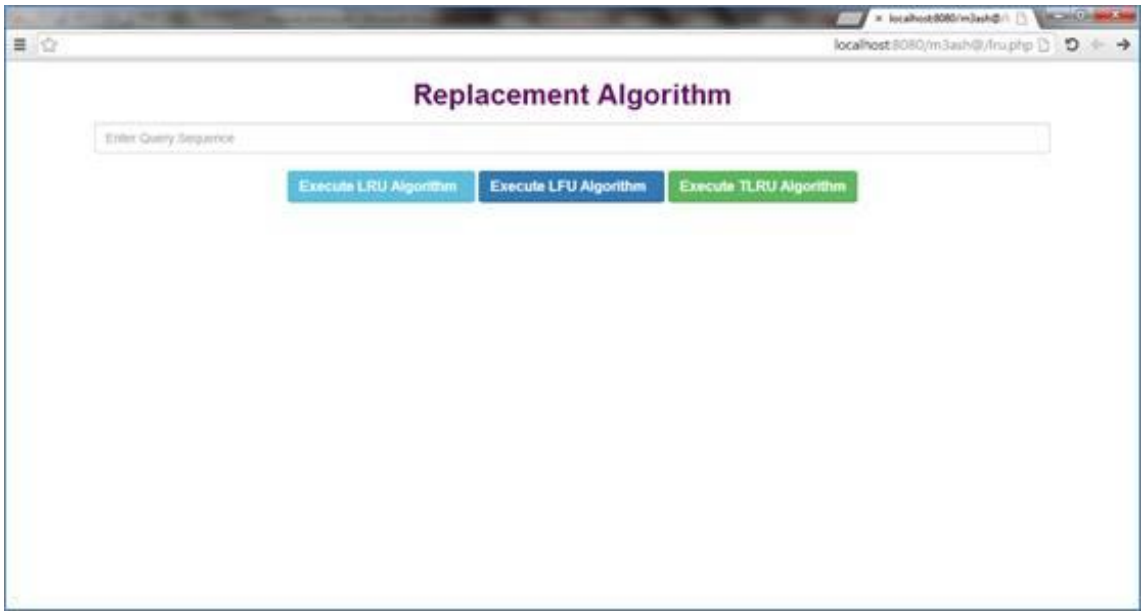

Figure 4.1: main page Allow to enter requests,choose an algorithm

4.4 Results of different number of Requests ,Cache size=4

4.4.1 Firstly: 10 Requests

I enter 10 requests with 4 cache size,the result was 0 percent hit ratio and 100 percent missed. the application achieves the same result for three algorithms.

4.4.1.1 LRU result screen 10 Requests :

Figure 4.2: result screen for10 requests and cache=4,lru algorithm

4.4.1.2 LFU result screen 10 Requests :

Figure 4.3: result screen for10 requests and cache=4,lfu algorithm

4.4.1.3 TLRU result screen 10 Requests:

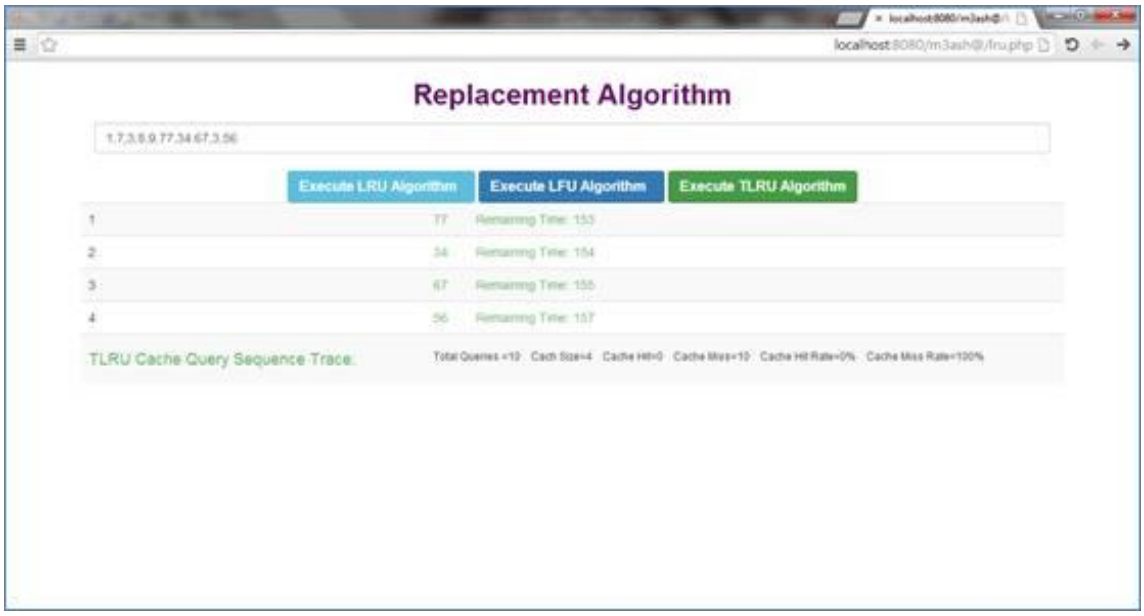

Figure 4.4: Result screen for10 requests and cache=4, TLRU algorithm

4.4.2 Secondly :20 Requests

Enter 20 request with 4 cache size , the result was 5 percent hit ratio and 95 percent miss for LRU and LFU,TLRU.

4.4.2.1 LRU result screen 20 Requests:

Figure 4.5: result screen for20 requests and cache=4, LRU algorithm

4.4.2.2 LFU result screen 20 Requests:

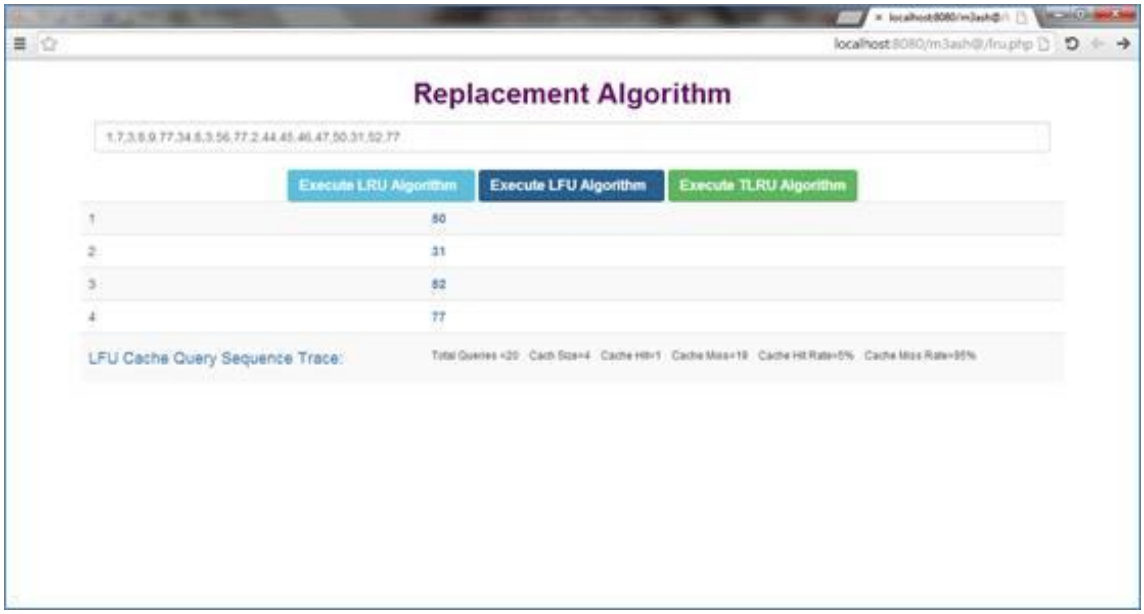

Figure 4.6: Result screen for20 requests and cache=4,LFU algorithm

4.4.2.3 TLRU result screen 20 Requests:

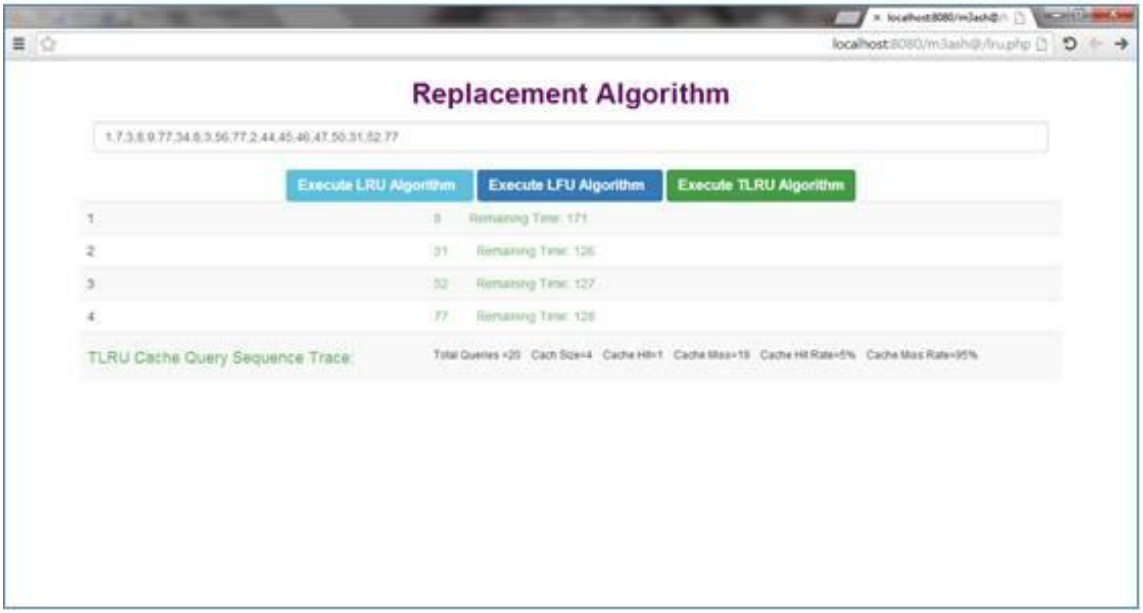

Figure 4.7: Result screen for20 requests and cache=4,TLRU algorithm

4.4.3 Thirdly: 24 Requests

24 requests with 4 cache size. LRU and LFU, TLRU achieve 12.5

percent hit ratio and 87.5 percent miss ratio. All three algorithms achieve the same results for the small cache size.

4.4.3.1 LRU result screen 24 Requests:

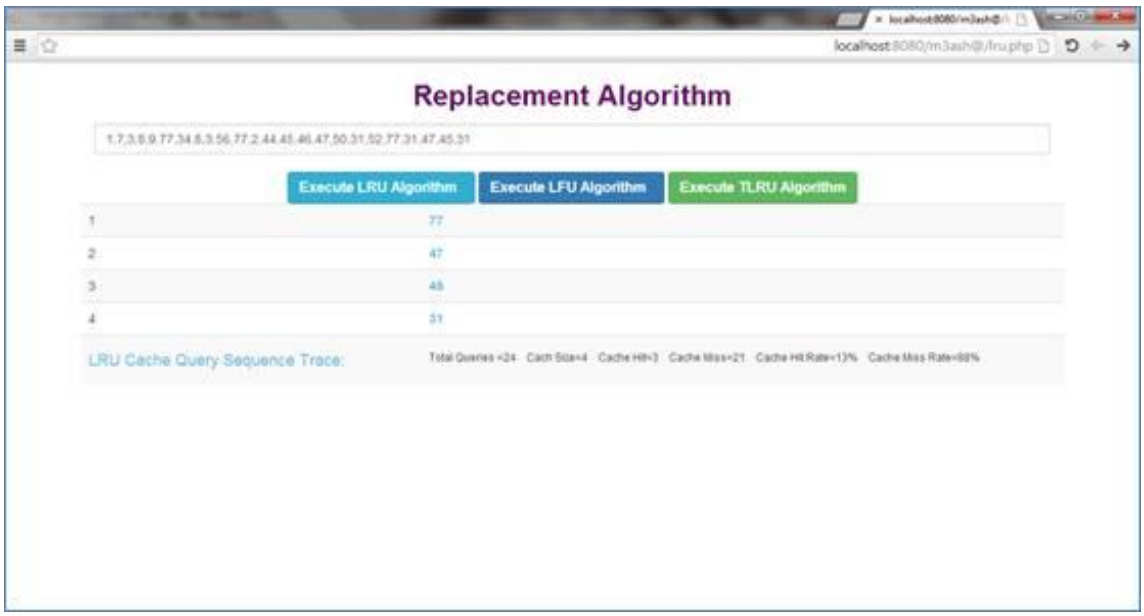

Figure 4.8: result screen for24 requests and cache=4,lru algorithm

4.4.3.2 LFU result screen 24 Requests

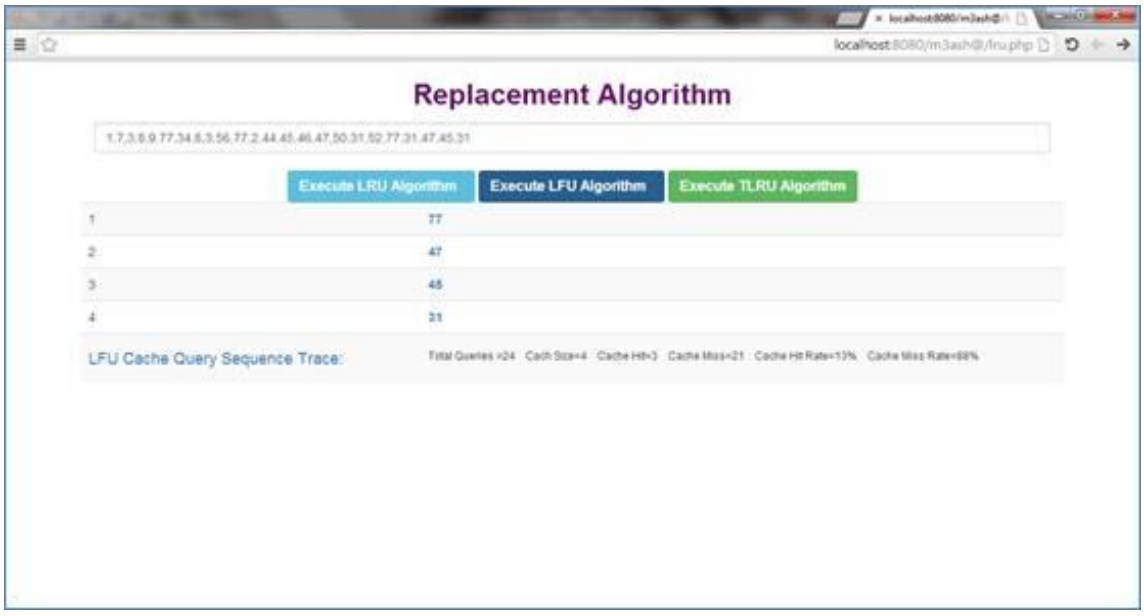

Figure 4.9: result screen for24 requests and cache=4,lfu algorithm

4.4.3.3 TLRU result screen 24 Requests

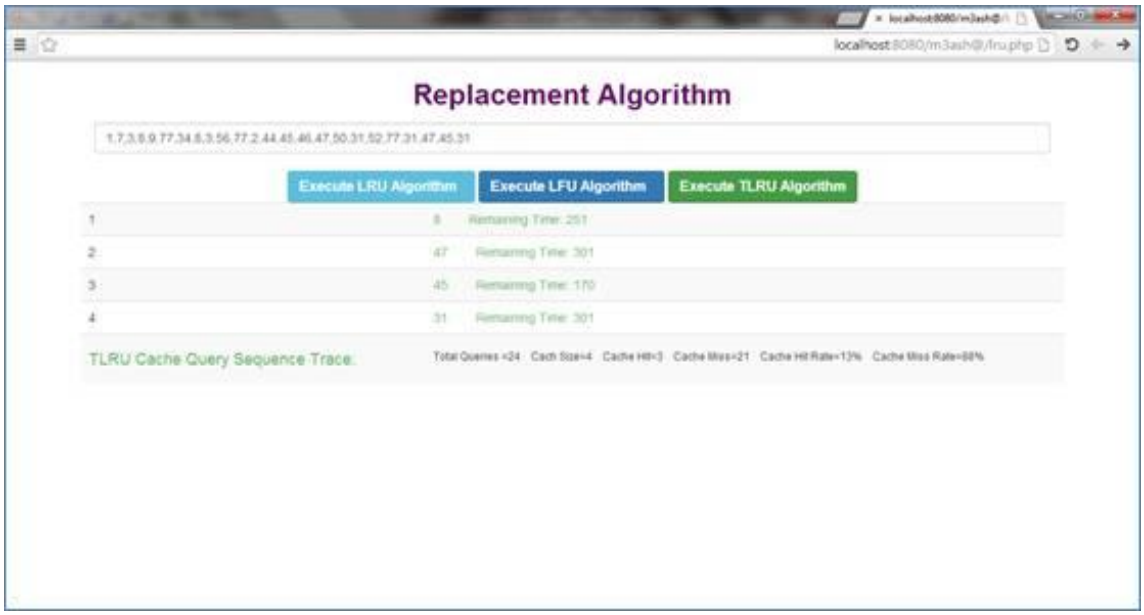

Figure 4.10: result screen for24 requests and cache=4, TLRU algorithm

4.5 Results of different number of Requests Cache size=8

4.5.1 Firstly :10 Requests

First enter 10 requests with 8 cache size, the result was 10 percent hit ratio and 90 percent missed. the application achieves the same result for three algorithms.

4.5.1.1 LRU result screen 10 Requests

Figure 4.11: Result screen for10 requests and cache=8,LRU algorithm

4.5.1.2 LFU Result screen 10 Requests

Figure 4.12: result screen for10 requests and cache=8,lfu algorithm

4.5.1.3 T LRU result screen 10 Requests

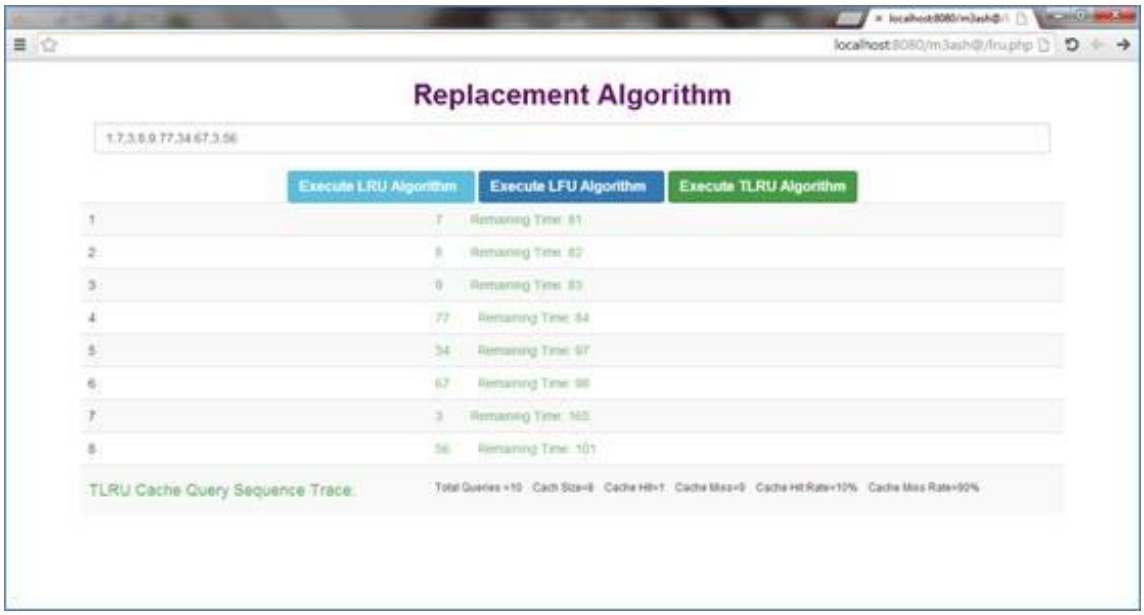

Figure 4.13: result screen for10 requests and cache=8,tlru algorithm

4.5.2 Secondly : 20 Requests

enter 20 requests with 8 cache size ,the result was 10 percent hit ratio and 90 percent missed.the application achieve the same result for three algorithms.

4.5.2.1 LRU result screen 20 Requests

Figure 4.14: result screen for 20 requests and cache=8,lru algorithm

4.5.2.2 LFU result screen 20 Requests

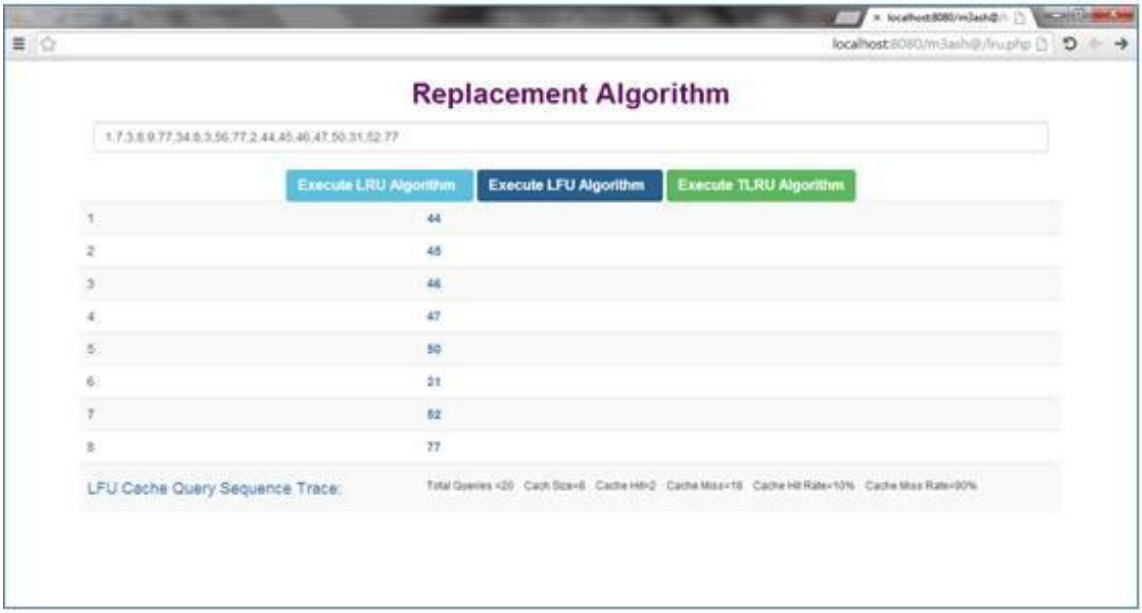

Figure 4.15: result screen for 20 requests and cache=8,lfu algorithm

4.5.2.3 TLRU result screen 20 Requests

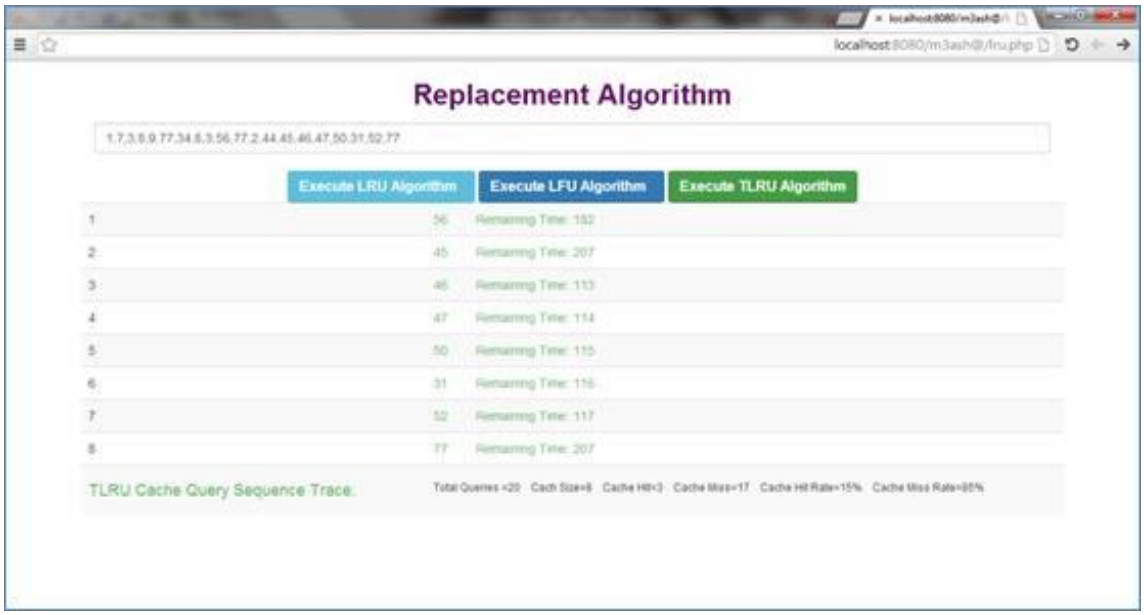

Figure 4.16: result screen for 20 requests and cache=8,tlru algorithm

4.5.3 Third: 24 Requests

Enter 24 requests with 8 cache size. LRU and LFU achieve 25 percent

hit ratio and 75 percent miss ratio. TLRU is 29 percent hit ratio and 71percent miss.

4.5.3.1 LRU result screen 24 Requests

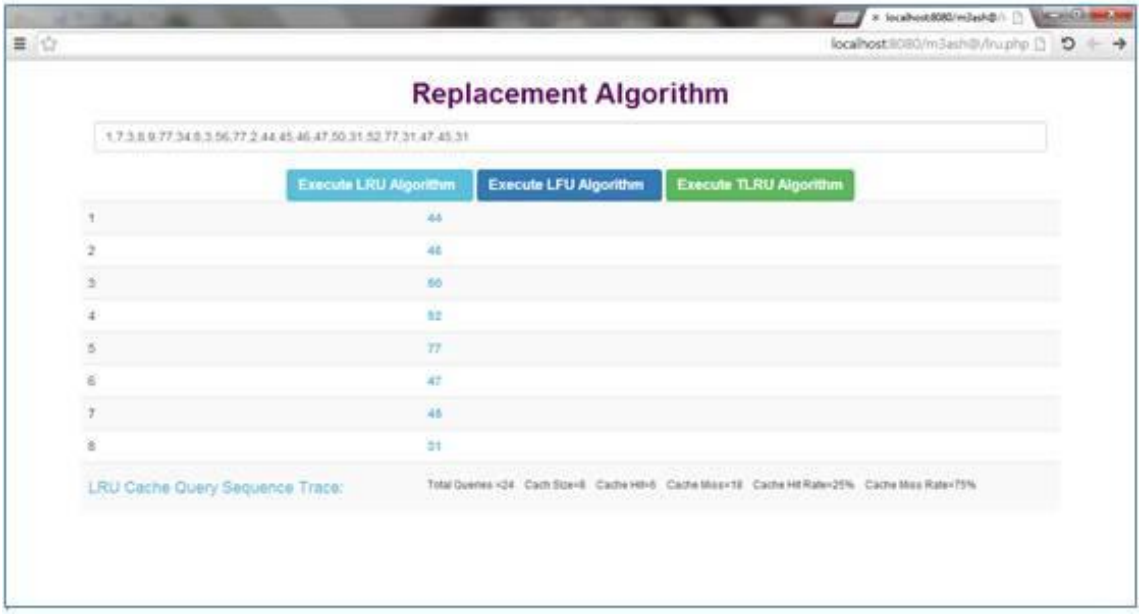

Figure 4.17: result screen for 24 requests and cache=8,lru algorithm.

4.5.3.2 LFU result screen 24 Requests

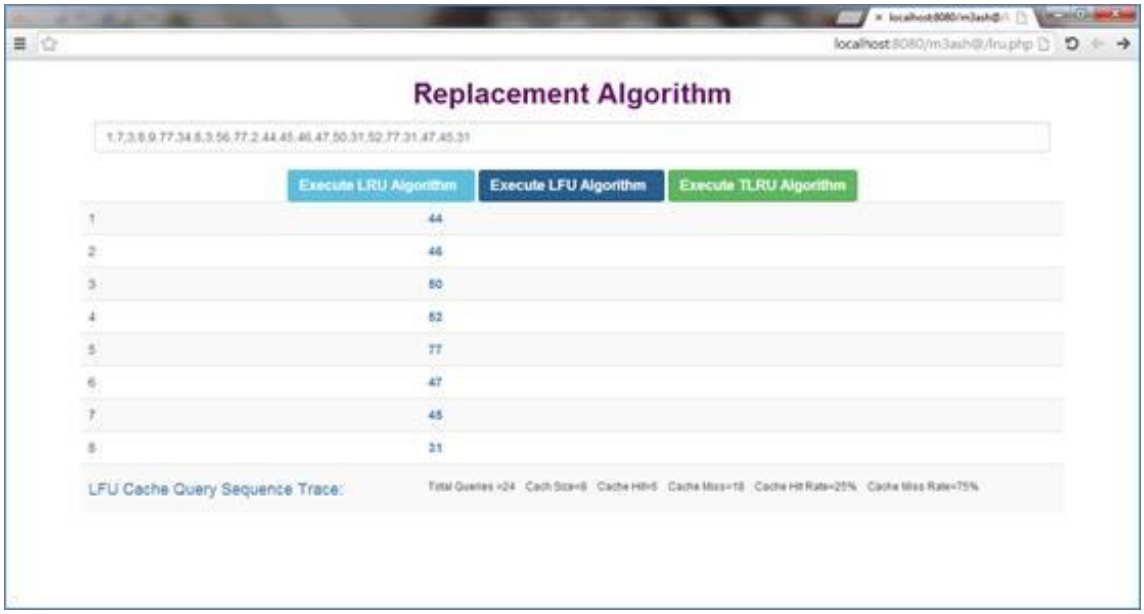

Figure 4.18: result screen for 24 requests and cache=8,lfu algorithm.

4.5.3.3 TLRU result screen 24 Requests

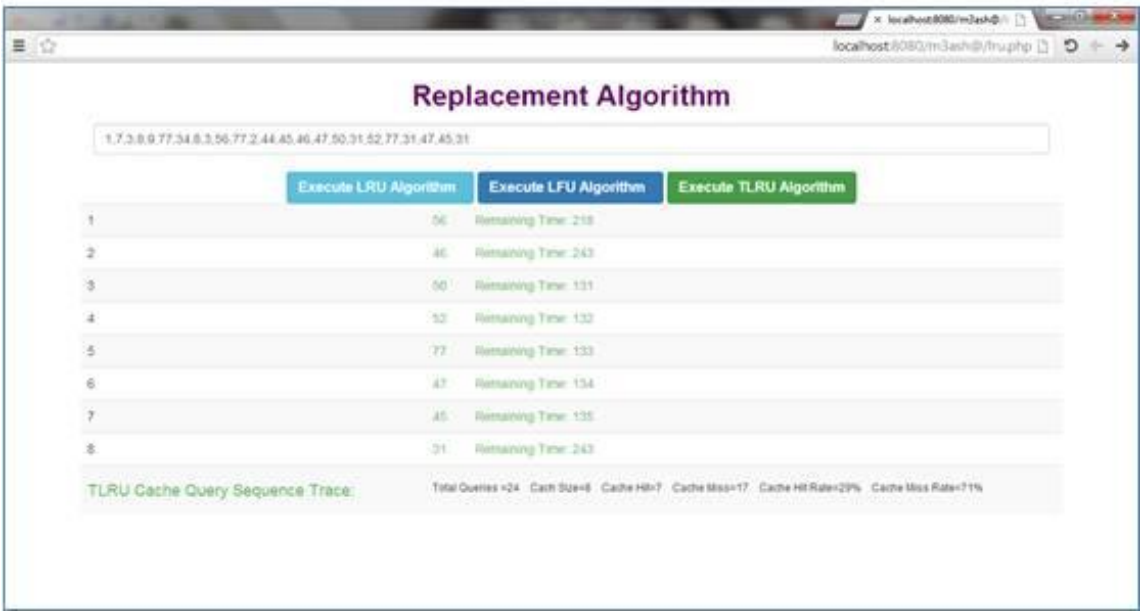

Figure 4.19: result screen for 24 requests and cache=8, TLRU algorithm.

4.6 The charts

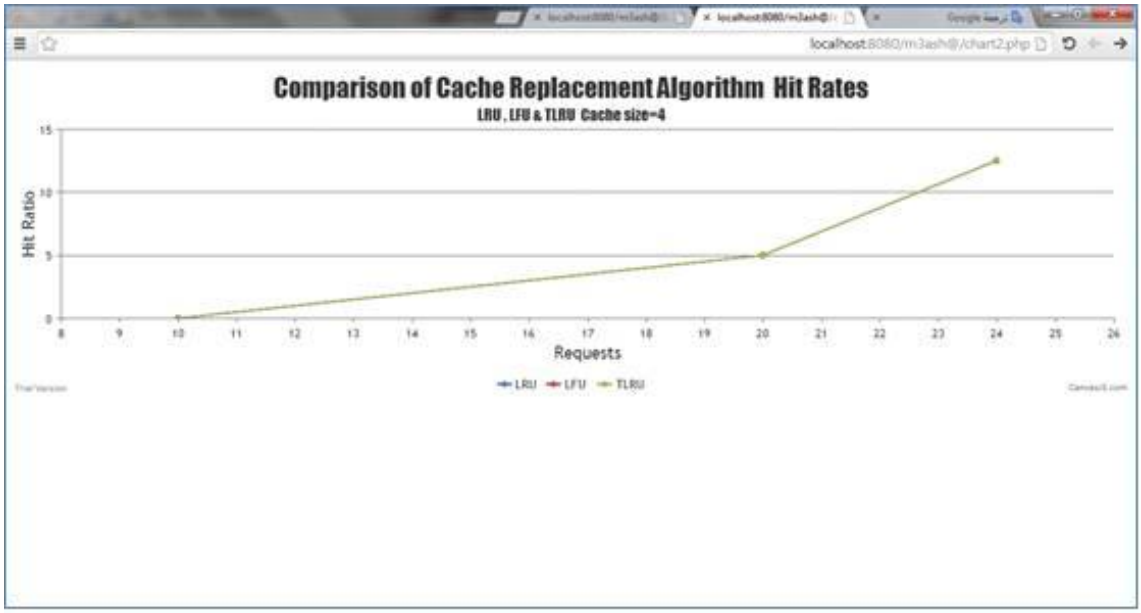

Figure 4.20: chart comparison between tlru ,lfu and lru for 4 bytes cache size

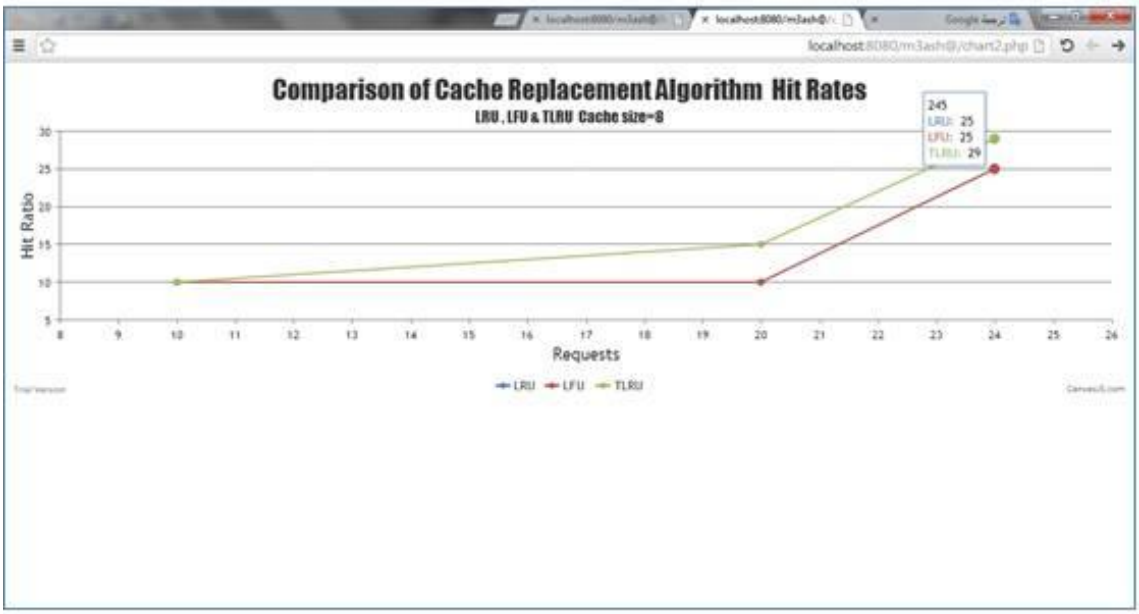

Figure 4.21: chart comparison between tlru ,lfu and lru for 8 bytes cache size

4.7 Discussion of analysis and finding

1. The hit and miss ratio results , are equal in all three algorithms, when the cache size is small.
2. TLRU achieves the highest Hit rate when the cache size become larger ,it takes the content lifetime in account .
3. Also TLRU could result the same hit and missed with two other algorithm if the cache size is small and exceed them when there is cache with large size.
4. TLRU is time award, so it increases the content lifetime if hitted to stay longer and discard the content with small lifetime.it is sortable in cases of concentric on popularity.
5. The Simulation results showed that the hit and miss ratio equal in all three algorithms, when the cache size is small, and TLRU achieves highest hit ratio for large cache size.
6. LRU, LFU, TLRU achieves 12.5 percent with 4 bytes' cache size.
7. LRU, LFU achieves 25 percent hit, and TLRU achieves 29 percent hit

with 8bytes cache size, using the same number of requests.

4.8 Summary

This chapter presented the simulation environment and the simulation results to compare the three algorithms. The results was listed based on hit ratio, number of requests .Simulation results showed that the hit and miss ratio equal in all three algorithms, when the cache size is small and TLRU achieves highest hit ratio for large cache size.

The simulation results showed that the hit and miss ratio equal in all three algorithms, when the cache size is small, and TLRU achieves highest hit ratio for large cache size. LRU, LFU, TLRU achieves 12.5 percent with 4bytes cache size. LRU, LFU achieves 25 percent hit, and TLRU achieves 29 percent hit with 8bytes cache size, using the same number of requests.

Chapter Five
Conclusions and recommendation

5.1 Conclusions

Information centric network becomes a tremendous research area nowadays. This research focus on caching replacement strategy and by inspire of this topic in networking field, I studied several researches regarding caching in ICN. Caching is just storage of content data with aiming of speedily served future request. This research presented an overview of various caching re-placement approaches in ICN with several features and regarding issues. This research gives a simple idea about ‘how the event cache content eviction should take place to achieve a high hit ratio’ problem, The proposed solution gives very high performance in terms of cache hit ratio with comparison of LRU, LFU, TLRU.

From previous mentioned the contributions of research as follow:

The model can be use to manage the cache, by selecting the right algorithm, choosing LRU in small cache, and TLRU in large cache size, and for big numbers of requests. Also Increase the network performance when the hit ratio increases. The application is a light weight and need not high ram speed and large memory size.

5.2 Recommendations

Maintaining the consistency of content data if anything updating occurs

Synchronization with various caching routers with respect to the server

Reducing redundancy of content data item in various caches Optimization of cache space to achieve high capacity to store content data item and so much else have to consider to make an effective and efficient cache mechanism in information centric network. This will lead to bright future of ICN in current internet access scenarios.

References

[1] K. Yu, S. Eum, T. Kurita, Q. Hua, T. Sato, H. Nakazato, T. Asami, and V. P. Kafle, "Information-centric networking: Research and standardization status," *IEEE Access*, vol. 7, pp. 126 164–126 176, 2019.

[2] M. Chand, "A comparative survey on different caching mechanisms in named data networking (ndn) architecture," *International Journal of Emerging Technologies and Innovative Research*, vol. 6, no. 4, pp. 264– 271, 2019.

[3] F. Qazi, O. Khalid, R. N. B. Rais, I. A. Khan *et al.*, "Optimal content caching in content-centric networks," *Wireless Communications and Mo-bile Computing*, vol. 2019, 2019.

[4] S. J. Taher, O. Ghazali, and S. Hassan, "A review on cache replacement strategies in named data network," *Journal of Telecommunication, Elec-tronic and Computer Engineering (JTEC)*, vol. 10, no. 2-4, pp. 53–57, 2018.

[5] C. Bernardini, T. Silverston, and A. Vasilakos, "Caching strategies for information centric networking: opportunities and challenges," *arXiv preprint arXiv:1606.07630*, 2016.

[6] M. A. Yaqub, S. H. Ahmed, S. H. Bouk, and D. Kim, "Information-centric networks (icn)," in *Content-Centric Networks*. Springer, 2016, pp. 19–33.

[7] Q. N. Nguyen, J. Liu, Z. Pan, I. Benkacem, T. Tsuda, T. Taleb, S. Shimamoto, and T. Sato, "Ppcs: a progressive popularity-aware caching scheme for edge-based cache redundancy avoidance in information-centric networks," *Sensors*, vol. 19, no. 3, p. 694, 2019.

[8] H. Liu, K. Azhandeh, X. de Foy, and R. Gazda, "A comparative study of name resolution and routing mechanisms in information-centric networks," *Digital Communications and Networks*, vol. 5, no. 2, pp. 69–

75, 2019.

[9] M. Zhang, H. Luo, and H. Zhang, “A survey of caching mechanisms in information-centric networking,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1473–1499, 2015.

[10] H. Khelifi, S. Luo, B. Nour, and H. Moungla, “In-network caching in icn-based vehicular networks: Effectiveness & performance evaluation,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[11] M. Alkhazaleh, S. Aljunid, and N. Sabri, “A comprehensive survey of information-centric network: Content caching strategies perspective.”

[12] R. Hegadi, A. Kammar, and S. Budihal, “Performance evaluation of in-network caching: A core functionality of information centric networking,” in *2019 International Conference on Data Science and Communication (IconDSC)*. IEEE, 2019, pp. 1–8.

[13] P. Sena, I. Carvalho, and A. Abelém, “Content placement aware cache decision: A caching policy based on the content replacement ratio for information-centric network,” in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Comput-ing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1913–1920.

[14] N. Alzakari, A. B. Dris, and S. Alahmadi, “Randomized least frequently used cache replacement strategy for named data networking,” in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2020, pp. 1–6.

[15] Y. Navrotsky and N. Patsei, “Cashing control and optimization in information-content networks,” in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. IEEE, 2019, pp. 1–5.

[16] M. Amadeo, C. Campolo, A. Molinaro, J. Harri, C. E. Rothenberg, and

A. Vinel, “Enhancing the 3gpp v2x architecture with information-centric networking,” *Future Internet*, vol. 11, no. 9, p. 199, 2019.

[17] A. Ioannou and S. Weber, “A taxonomy of caching approaches in information-centric network architectures,” *Elsevier Journal*, 2013.

[18] B. Al-Duwairi and Ö. Özkasap, “Preventing ddos attacks in path identifiers-based information centric networks,” in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–5.

[19] L. Saino, I. Psaras, and G. Pavlou, “Icarus: a caching simulator for information centric networking (icn),” in *SimuTools*, vol. 7. ICST, 2014, pp. 66–75.

[20] B. ALOTAIBI and S. ALAHMADI, “Efficient caching and replacement strategy in content centric network (ccn) based on xon-path and hop count.”

[21] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Hab-bal, “Caching in information-centric networking: Strategies, challenges, and future research directions,” *IEEE Communications Surveys & Tuto-rials*, vol. 20, no. 2, pp. 1443–1474, 2017.

[22] K. N. Lal and A. Kumar, “A cache content replacement scheme for infor-mation centric network,” *Procedia Computer Science*, vol. 89, pp. 73–81, 2016.

[23] Y. S. Rani and M. Seetha, “Enhanced poc tree-based algorithm for data item correlation and cache effective replacement in mobile ad hoc net-work,” *Int. J. Pure Appl. Math.*, vol. 118, no. 19, pp. 225–247, 2018.

[24] M. Bilal and S.-G. Kang, “Time aware least recent used (tlru) cache management policy in icn,” in *16th International Conference on Advanced Communication Technology*. IEEE, 2014, pp. 528–532.

[25] M. A. P. Putra, H. Situmorang, and N. R. Syambas, “Least recently frequently used replacement policy named data networking approach,” in *2019 International Conference on Electrical Engineering and*

Informatics (ICEEI), 2019, pp. 423–427.

[26] M. Hussaini, S. A. Nor, and A. Ahmad, "Producer mobility support for information centric networking approaches: A review," *Int. J. Appl. Eng. Res*, vol. 13, no. 6, pp. 3272–3280, 2018.

[27] M. Hussaini, M. A. Naeem, B.-S. Kim, and I. S. Maijama'a, "Efficient producer mobility management model in information-centric network-ing," *IEEE Access*, vol. 7, pp. 42 032–42 051, 2019.

[28] M. Sardara, "Towards a scalable and programmable incremental deploy-ment of icn in the real world," Ph.D. dissertation, Université Paris-Saclay, 2019.

[29] G. Liu, W. Quan, N. Cheng, B. Feng, H. Zhang, and X. S. Shen, "Blam: Lightweight bloom-filter based ddos mitigation for information-centric iot," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[30] P. Gasti and G. Tsudik, "Content-centric and named-data networking se-curity: The good, the bad and the rest," in *2018 IEEE International Sym-posium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2018, pp. 1–6.

[31] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: A survey," *IEEE com-munications surveys & tutorials*, vol. 20, no. 1, pp. 566–600, 2017.

[32] L. Dong and G. Wang, "A hybrid approach for name resolution and producer selection in information centric network," in *2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2018, pp. 574–580.

[33] M. Aamir, "Content-priority based interest forwarding in content centric networks," *arXiv preprint arXiv:1410.4987*, 2014.

[34] Y. Ren, J. Li, L. Li, S. Shi, J. Zhi, and H. Wu, "Modeling content trans-fer performance in information-centric networking," *Future Generation Computer Systems*, vol. 74, pp. 12–19, 2017.

Appendix

A.1 php code

A.2 TLRU Algorithm

Timer.php file

```php
  <?php
class Timer {

    var $classname = "Timer";
    var $start     = 0;
    var $stop      = 0;
    var $elapsed   = 0;

    # Constructor
    function Timer( $start = true ) {
        if ( $start )
            $this->start();
    }

    # Start counting time
    function start() {
        $this->start = $this->_gettime();
    }

    # Stop counting time
    function stop() {
        $this->stop    = $this->_gettime();
        $this->elapsed = $this->_compute();
    }

    # Get Elapsed Time
    function elapsed() {
        if ( !$elapsed )
            $this->stop();

        return $this->elapsed;
    }

    # Resets Timer so it can be used again
    function reset() {
        $this->start   = 0;
        $this->stop    = 0;
        $this->elapsed = 0;
    }

    #### PRIVATE METHODS ####

    # Get Current Time
    function _gettime() {
        $mtime = microtime();
        $mtime = explode( " ", $mtime );
        return $mtime[1] + $mtime[0];
    }

    # Compute elapsed time
    function _compute() {
        return $this->stop - $this->start;
    }
}
?>
```

Lfu.php file

```
<?php
error_reporting(0);
 function leastFrequent($arr, $n)
{
    // Sort the array
    sort($arr);
    sort($arr , $n);
    // find the min frequency   min heap
    // using linear traversal
    $min_count = $n + 1;
    $res = -1;
    $curr_count = 1;
    for($i = 1; $i < $n; $i++)
    {
        if ($arr[$i] == $arr[$i - 1])
            $curr_count++;
        else
        {
            if ($curr_count < $min_count)
            {
                $min_count = $curr_count;
                $res = $arr[$i - 1];
            }
            $curr_count = 1;
        }
    }

    // If last element is
    // least frequent
    if ($curr_count < $min_count)
    {
        $min_count = $curr_count;
        $res = $arr[$n - 1];
    }

    return $res;
}
////////////////////////////////////////////////////////////////////
//////////////////////////
  $n=$_REQUEST["number"];
  // echo $n;
   if(!empty($_REQUEST["number"]))
  {   $ss=$_REQUEST["number"];  }
  //*************************************************
 $arr = array();
$arr2 = array();
$arr3 = array();
$arr=$_REQUEST["number"];
$cachsize=4;
$fault=0;
$hit=0;
$c=0;
 $mystr=""; $q="";
    $sizz = strlen($ss);
  for($i=0;$i<$sizz;$i++)
  {
    if($arr[$i]<>',')
    { $mystr=$mystr.$arr[$i] ; }
    if($arr[$i]==',')
```

```
    {
    $q=$mystr;
    $mystr="";
     $c=$c+1;
    }
     $arr2[$c]=$mystr;
  }
  $myco=0;
    for($j=0;$j<=$c;$j++)
    {
       $aa=$arr2[$j];
        if($myco<$cachsize)
        {
        if(array_key_exists($aa, $arr3))
           {
             $hit=$hit+1;
             $aapos=key($arr3);
             $f=true;
             $mycount=count($arr3);
              $aapos=null;
             ////////////////////////////////////////
              for($x=0;$x<$mycount;$x++){
              if(($arr3[$x])==($aa))
              $aapos=$x;      }
             ////////////////////////////////////////
              for($bj=$aapos;$bj<$mycount;$bj++){
               if(($bj+1)<$mycount)
                 { $arr3[$bj]=$arr3[$bj+1];}
              }
              $arr3[$mycount-1]=$aa;
             }
            else
          {unset($arr3[$j]);
          $arr3[$myco]=$arr2[$j];   //insert it into arr3 the
first time
           $myco=$myco+1;
         $fault=$fault+1;}
        }else break;
    }//echo $j;
 //search for lfu value in the cache////////arr3 is cache to put
in///
    $lastpos=$myco;
    $mycount=count($arr3);
     if($myco>=$cachsize){
        $f=false;
       $as=$myco+1;
     ////////////////////////////////////in case of full cache
size////////////////////////////////////
       for($m=$j;$m<=$c;$m++){
            $f=false;
       for($k=0;$k<$cachsize;$k++)
       {
        $position=$k;
          if(($arr2[$m]==$arr3[$k])and($f==false))
          {
            $hit=$hit+1;
              $temp=$arr3[$k];
            for($b=$k;$b<$cachsize;$b++){
            if(($b+1)<$cachsize)
            $arr3[$b]=$arr3[$b+1];
           }
```

```
            $arr3[$cachsize-1]=$temp;
            $f=true;
          }
        }
         if(($f==false))
          {
            $temp2=$arr3[0];
             $nn= sizeof($arr2) / sizeof($arr2[0]);
            $lfu=leastFrequent($arr2,$nn);
            for($y=0;$y<$cachsize;$y++) {
            if($lfu==$arr3[$y])
            {$lfucount=$y;
            }
            }
            $arr3[$lfucount]=$arr2[$m];
            for($bb=$lfucount;$bb<$cachsize;$bb++){
            if(($bb+1)<$cachsize){
            $arr3[$bb]=$arr3[$bb+1];
            }
           }
            $arr3[$cachsize-1]=$arr2[$m];    //replace here//
            $fault=$fault+1;
             }  //end if
    }  //end for1
    }//////////////////////end
                $z=0;
                foreach ($arr3 as $value1) {
                 $z=$z+1;
                   if($z<=$cachsize) {echo"<tr><td
width='3'>".$z."</td><td width='30'>";}
                   if($z<=$cachsize)       echo "<font
color='337ab7'><b>".$value1."</b></font>";
                    else
                        if($z<=$cachsize) {echo "</td></tr>";}
                   }
echo"<tr><td><font color='337ab7'><h4>  LFU Cache Query Sequence
Trace:</h4></td><td><h6> Total Queries
=".($c+1)."    Cach Size=".$cachsize;
echo"    Cache
Hit=".$hit."    Cache
Miss=".$fault."    Cache Hit
Rate=".(round($hit/($c+1),2)*100)."%    Cach
e Miss
Rate=".(round($fault/($c+1),2)*100)."%</h6></font></td></tr>";
     ?>
```

Main.php file

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title></title>
 <meta name="viewport" content="width=device-width, initial-
scale=1">
  <link rel="stylesheet"
href="bootstrap3.4.1/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.m
in.js"></script>
```

```
  <script src="bootstrap33.4.1/js/bootstrap.min.js"></script>
<script src="https://code.jquery.com/jquery-
1.12.4.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
         ClickedEl = this.id;
        var numValue = $("#num").val();
        if(ClickedEl=="b1"){
            // Send the input data to the server using get
            $.get("lru1.php", {number: numValue} ,
function(data){
            // Display the returned data in browser
            $("#result").html(data);
             });
        }//if
         if(ClickedEl=="b2"){
            // Send the input data to the server using get
            $.get("mylfu.php", {number: numValue} ,
function(data){
            // Display the returned data in browser
            $("#result").html(data);
             });
        }//if
        if(ClickedEl=="b3"){
            // Send the input data to the server using get
            $.get("tlrualg.php", {number: numValue} ,
function(data){
            // Display the returned data in browser
            $("#result").html(data);
             });
        }//if
    });
});
</script>
<body>
<div class="container" align="center">
<div class="panel-body">

     <p align="center"><b><font size="6"
color="660066">Replacement Algorithm</font></b></p>
     <div class="col-xs-12">
     <input class="form-control" id="num" size="" type="text"
style="width:24 px" placeholder="Enter Query Sequence"><br>
        <button type="button" id="b1"class="btn btn-
info"><b><font size="3"> Execute LRU Algorithm  
</font></b></button>
        <button type="button" id="b2"class="btn btn-
primary"><b><font size="3"> Execute LFU Algorithm  
</font></b></button>
        <button type="button" id="b3" class="btn btn-
success"><b><font size="3"> Execute TLRU Algorithm
</font></b></button>
     </div>
        <table class="table table-striped" border="0"
width="100%" height="" align="right" id="result"> </table>
</div>
</div>
</body>
</html>
```

Chart.php file

```
<?php

 $dataPoints1 = array(
    array("x" => 0, "y" =>  0),
    array("x" => 10, "y" => 10),
    array("x" => 20, "y" => 15),
    array("x" => 24, "y" => 25)
 );

 $dataPoints2 = array(
    array("x" => 0, "y" => 0),
    array("x" => 10, "y" => 10),
    array("x" => 20, "y" => 20),
    array("x" => 24, "y" => 29)
);
 //$dataPoints3 = array(
//  array("x" => 0, "y" =>  0),
//  array("x" => 10, "y" => 10),
//  array("x" => 20, "y" => 15),
//  array("x" => 24, "y" => 25)
 //);

?>
<!DOCTYPE HTML>
<html>
<head>
<script>
window.onload = function () {

var chart = new CanvasJS.Chart("chartContainer", {
    animationEnabled: true,
    title:{
        text: "Comparison of Cache Replacement Algorithm  Hit
Rates"
    },
    subtitles: [{
        text: "LRU & TLRU  Cache size=6",
        fontSize: 18
    }],
    axisY: {
        prefix: ""
    },
    legend:{
        cursor: "pointer",
        itemclick: toggleDataSeries
    },
    toolTip: {
        shared: true
    },
    data: [
    {
        type: "line",
        name: "LRU",
        showInLegend: "true",
        //xValueType: "dateTime",
        //xValueFormatString: "MMM YYYY",
        yValueFormatString: "##0.##",
```

```
        xValueFormatString: "##5.##",
        dataPoints: <?php echo json_encode($dataPoints1); ?>
    },
    {
        type: "line",
        name: "TLRU",
        showInLegend: "true",
        //xValueType: "dateTime",
        //xValueFormatString: "MMM YYYY",
        yValueFormatString: "##0.##",
        xValueFormatString: "##5.##",
        dataPoints: <?php echo json_encode($dataPoints2); ?>
    }
    /*{
        type: "line",
        name: "LFU",
        showInLegend: "true",
        //xValueType: "dateTime",
        //xValueFormatString: "MMM YYYY",
        yValueFormatString: "##0.##",
        xValueFormatString: "##5.##",
        dataPoints: <?php echo json_encode($dataPoints3); ?>
    }*/
    ]
});

chart.render();

function toggleDataSeries(e){
    if (typeof(e.dataSeries.visible) === "undefined" ||
e.dataSeries.visible) {
        e.dataSeries.visible = false;
    }
    else{
        e.dataSeries.visible = true;
    }
    chart.render();
}

}
</script>
</head>
<body>
<div id="chartContainer" style="height: 370px; width:
100%;"></div>
<script src="canvasjs/canvasjs.min.js"></script>
</body>
</html>
```

Chart2.php file

```
<?php

 $dataPoints1 = array(
    array("x" => 0, "y" =>  0),
    array("x" => 10, "y" => 10),
    array("x" => 20, "y" => 10),
    array("x" => 24, "y" => 25)
 );
```

```
 $dataPoints2 = array(
    array("x" => 0, "y" => 0),
    array("x" => 10, "y" => 10),
    array("x" => 20, "y" => 15),
    array("x" => 24, "y" => 29)
);
 //$dataPoints3 = array(
//  array("x" => 0, "y" =>  0),
//  array("x" => 10, "y" => 10),
//  array("x" => 20, "y" => 15),
//  array("x" => 24, "y" => 25)
 //);

?>
<!DOCTYPE HTML>
<html>
<head>
<script>
window.onload = function () {

var chart = new CanvasJS.Chart("chartContainer", {
    animationEnabled: true,
    title:{
        text: "Comparison of Cache Replacement Algorithm  Hit
Rates"
    },
    subtitles: [{
        text: "LRU & TLRU  Cache size=8",
        fontSize: 18
    }],
    axisY: {
        prefix: ""
    },
    legend:{
        cursor: "pointer",
        itemclick: toggleDataSeries
    },
    toolTip: {
        shared: true
    },
    data: [
    {
        type: "line",
        name: "LRU",
        showInLegend: "true",
        //xValueType: "dateTime",
        //xValueFormatString: "MMM YYYY",
        yValueFormatString: "##0.##",
        xValueFormatString: "##5.##",
        dataPoints: <?php echo json_encode($dataPoints1); ?>
    },
    {
        type: "line",
        name: "TLRU",
        showInLegend: "true",
        //xValueType: "dateTime",
        //xValueFormatString: "MMM YYYY",
        yValueFormatString: "##0.##",
        xValueFormatString: "##5.##",
        dataPoints: <?php echo json_encode($dataPoints2); ?>
    }
```

```
    /*{
        type: "line",
        name: "LFU",
        showInLegend: "true",
        //xValueType: "dateTime",
        //xValueFormatString: "MMM YYYY",
        yValueFormatString: "##0.##",
        xValueFormatString: "##5.##",
        dataPoints: <?php echo json_encode($dataPoints3); ?>
    }*/
    ]
});

chart.render();

function toggleDataSeries(e){
    if (typeof(e.dataSeries.visible) === "undefined" ||
e.dataSeries.visible) {
        e.dataSeries.visible = false;
    }
    else{
        e.dataSeries.visible = true;
    }
    chart.render();
}

}
</script>
</head>
<body>
<div id="chartContainer" style="height: 370px; width:
100%;"></div>
<script src="canvasjs/canvasjs.min.js"></script>
</body>
</html>
```

TLRU Algorithm:

1- $f = \frac{|c_i|}{|n_j|} * TTu^i$

2- g= $\frac{p_{ij}}{\sum k \neq i \; p_{ij}} * TTu^i$

3-TTu^i=f(TTu^i)↑g(TTu^i)

4-If TTu^ij> t_{ij}

5-If s[j]≥ $|n_j|$

6-Do ʊ $c_i \in s[j]$

7-If TTu^ij $< t_{ij} \rightarrow \in Ev[j]$

8-LRU($Ev[j]$) → evict

9- s[j] → c_i U s[j]

10- Else

11- LRU($s[j]$) → evict

12- s[j] → c_i U s[j]

13- else s[j] → c_i U s[j]

14- else reject